CDP SUBMIX Functions

(with Command Line Usage)

Functions to MIX soundfiles

(Names in brackets mean that these are separate programs. The others are sub-modules of SUBMIX.)

ADDTOMIX
Add soundfiles (at maximum level and time zero) to an existing mixfile

ATSTEP
Convert a list of soundfiles to a mixfile (fixed time-step)

ATTENUATE
Alter the overall level of a mixfile

BALANCE
Mix between two soundfiles, using a balance function

CROSSFADE
Quick crossfade between 2 soundfiles (with same number of channels)

DUMMY
Convert list of sound names to a basic mixfile (for editing)

FADERS
Mix several mono or stereo files using a time-changing balance function

FILEFORMAT
Display format of a mixfile

GETLEVEL
Test maximum level of a mix, defined in a mixfile

INBETWEEN
Generate a set of sounds inbetween the 2 input sounds, through weighted mixes of the input sounds, from mostly sound1 to mostly sound2

INBETWEEN2
Generate a set of sounds in-between the 2 input sounds by interpolation pegged to zero-crossings

INTERLEAVE
Interleave mono files to make a single multichannel outfile

MERGE
Quick mix of 2 soundfiles (with the same number of channels)

MERGEMANY
Quick mix of several soundfiles (with the same number of channels)

MIX
Mix sounds as instructed in a mixfile

MODEL
Replace soundfiles in an existing mixfile

ONGRID
Convert listed soundfiles to a basic mixfile on a timed grid (for editing)

PAN
Pan sound positions in a mixfile

SHUFFLE
Shuffle the data in a mixfile

SPACEWARP
Alter the spatial distribution of a mixfile
SYNC
Synchronise soundfiles in a mixfile, or generate such a mixfile from a list of soundfiles
SYNCATTACK
Synchronise the attacks of soundfiles in a mixfile, or generate such a mixfile from a list of soundfiles
TEST
Test the syntax of a mixfile
TIMEWARP
Timewarp the data in a mixfile

ALSO SEE:
MULTIMIX CREATE
Create a multi-channel mixfile
NEWMIX
Mix from a multi-channel mixfile to give a multi-channel soundfile output

SUBMIX ADDTOMIX – Add soundfiles (at maximum level and time zero) to an existing mixfile

Usage

submix addtomix *inmixfile sndfile1 sndfile2 [sndfile3 ...] outmixfile*

Parameters

inmixfile – input *mixfile* to which to add sounds
sndfile1 – first sound to add
sndfile2 – second sound to add
[sndfile3 ...] – optional third sound to add, etc.
outmixfile – resultant output *mixfile*

Understanding the SUBMIX ADDTOMIX Function

Further sounds can be added to a *mixfile* simply by listing the original *mixfile* and the new sounds as inputs to the process. A new *mixfile* is created with the new sounds added at the foot of the file but all at start-time zero, and with maximum loudness. This new file can then be edited to re-position the start times of the new sounds in the new mix, and at the level you require.

The standard extension for a CDP *mixfile* is **.mix**, but is not required. *Sound Loom* uses **.txt** for all its text files.

Musical Applications

This is a simple utility designed to facilitate expanding the contents of a *mixfile*.

End of SUBMIX ADDTOMIX

SUBMIX ATSTEP – Convert a list of soundfiles to a mixfile (fixed time-step)

Usage

submix atstep *infile1 infile2 [infile3...] outmixfile step*

Parameters

infile1 – first soundfile in list
infile2 – second soundfile in list
[infile3...] – optional third soundfile in list, etc.
outmixfile – output mixfile produced by the program
step – time, in seconds, between the start times of each sound

Understanding the SUBMIX ATSTEP Function

Similar to **SUBMIX ADDTOMIX**, but here you specify a fixed time-interval between the entry of each sound in the mix.

Musical Applications

SUBMIX ATSTEP is useful if you want to use *mixfiles* for generating strictly rhythmic sequences of sounds. (You can 'naturalise' the rhythmic feel of the output by using existing CDP processes to slightly randomise the entry times of the sounds). See **SUBMIX TIMEWARP**.

End of SUBMIX ATSTEP

SUBMIX ATTENUATE – Alter the overall level of a mixfile

Usage

submix attenuate *inmixfile outmixfile gainval* [**-s***startline*] [**-e***endline*]

Parameters

inmixfile the input is a mixfile
outmixfile the output is the adjusted mixfile
gainval gain factor to be applied (Must be > 0.0)
-s*startline* line at which attenuation begins (Default: 1st line in mixfile
-e*endline* line at which attenuation begins (Default: last line in mixfile)

Understanding the SUBMIX ATTENUATE Function

The *gainval* parameter is a multiplier and is applied to the levels of all the sounds listed in the mixfile. The levels in the mixfile may be notated in dB (e.g., 0dB) or as floating point values (e.g., 1.0). Values > 1 will increase the level.

The gain may be applied to selected sounds in the mixfile by specifying 'start' and 'end' lines in the mixfile. Only one line will be affected if 'start' and 'end' are given the same line number. (The line count starts at 1.)

Musical Applications

CDP mixfiles may list a large number of sounds – no limit is encoded in the software. It may sometimes, therefore, be useful to adjust the attenuation of whole groups of sounds in the mixfile. This can be done easily with SUBMIX ATTENUATE.

End of SUBMIX ATTENUATE

SUBMIX BALANCE – Mix between 2 soundfiles, using a balance function

Usage

submix balance *sndfile1 sndfile2 outfile* [**-k***balance*] [**-b***start*] [**-e***end*]

Parameters

sndfile1 first soundfile to be mixed
sndfile2 second soundfile to be mixed

> The input soundfiles may or may not have a different number of channels.

outfile resultant mix
-k*balance* – describes the relative level of the two sounds (Range: 0 to 1)

- The level of *sndfile1* is multiplied by the *balance* function.
- The level of *sndfile2* is multiplied by the **inverse** of the *balance* function.
- *Balance* may vary over time.

-b*start* – start the mix at the time specified
-e*end* – stop the mix at the time specified

Understanding the SUBMIX BALANCE Process

This is a 'quick mix' function like **SUBMIX MERGE** except that one can easily vary the loudness of the sounds relative to one another by means of the *balance* parameter.

The 'inverse of the balance' means that the second soundfile will become as much softer as the *balance* multiplier makes the first soundfile louder, and *v.vs.*. The first file is multiplied by *balance*, and the second file by 1-*balance*.

For example, if two soundfiles are mixed without any reduction of their levels, their amplitudes will be summed (and probably go over). With SUBMIX BALANCE, their amplitudes relative to one another are adjusted such that they never exceed unity (maximum amplitude).

- If *balance* is 0.7 and the amplitude of both files is 10000, the first file will become 10000 * 0.7 = 7000, and the second file will become 10000 * (1-0.7, i.e., 0.3) = 3000.
- If *balance* is 0.3 and the amplitude of both files is 10000, the first file will become 10000 * 0.3 = 3000, and the second file will become 10000 * (1-0.3, i.e., 0.7) = 7000.

In other words, as the level of *sndfile1* goes up, the level of *sndfile2* goes down, etc., **but the some of both of them is always 1**, so they don't overload: the **total** amplitude remains the same.

If either file is less than maximum amplitude, the output will also be less than maximum amplitude. The output will always remain within an acceptable range.

Musical Applications

For very simple mixing jobs, where two input files need to be mixed, BALANCE offers another easier alternative to MIX. The files may be of any length, and the restriction on having similar sampling rates and numbers of channels is removed.

You are assured that overload is avoided by adjusting the relative volumes of the two soundfiles, matching them equally at *balance* = 0.5, making the first soundfile softer and the second louder when *balance* is < 0.5, and making the first soundfile louder and the second softer when *balance* is > 0.5. The relative volumes of the two sounds in the output, however, also depends on their respective levels in the first place.

End of SUBMIX BALANCE

SUBMIX CROSSFADE – Quick crossfade between 2 soundfiles (with same number of channels)

Usage

submix crossfade 1 *sndfile1 sndfile2 outfile* [**-s***stagger*] [**-b***begin*] [**-e***end*]
OR
submix crossfade 2 *sndfile1 sndfile2 outfile* [**-s***stagger*] [**-b***begin*] [**-e***end*] [**-p***powfac*]

Modes

1 Linear crossfade
2 Cosinusoidal crossfade

Parameters

sndfile1 – first input soundfile, from which crossfade will emerge
sndfile2 – second input soundfile, into which crossfade will move
outfile – output soundfile, which will start in the first soundfile and end in the second
-s*stagger* – 2nd file starts *stagger* seconds after 1st (Default: 0)
-b*begin* – crossfade starts at *begin* seconds; must be > *stagger* (Default: 0)
-e*end* – crossfade ends at *end* seconds; must be > *begin* (Default: end of shortest file)

If crossfade ends before end of *sndfile2*, the remainder of *sndfile2* plays on.

-p*powfac* – Crossfade skew:

If *powfac* = 1, cosinusoidal crossfade is normal
In range 0.13 to 1, cosinusoidal crossfade begins rapidly and then slows.
In range 1 to 8, cosinusoidal crossfade begins slowly and then speeds up.

Understanding the SUBMIX CROSSFADE Process

SUBMIX CROSSFADE is a mixing process applied directly to soundfiles. It does not entail the use of a mixfile. It mixes the two sounds, making a gradual transition from the first to the second.

The *stagger* parameter operates in steps of 100^{ths} of a second. If a more precise gap is required, the author suggests splicing a silence of a specific duration onto the beginning of the second soundfile, and then using a *stagger* of 0. Alternatively, SUBMIX MIX can be used, in which the start time of the second soundfile can be precisely defined.

Musical Applications

Note that only spectral morphing will create a true morph between two sounds. The CROSSFADE transition can, however, be useful to prepare sounds for morphing, as well as to create transitions in which the smooth merging of spectra is not the key issue.

Transition is one of the key principles of effective musical composition, as it creates a movement over time. In this case, the movement is between sounds, a process well known in the orchestration of instrumental music, but one uniquely suited to further development in an electroacoustic composing environment. The types of sounds and nature of the transition are determined by the compositional logic.

CDP has several functions with which to approach the transition process. **ALSO SEE:**

STRANGE GLIS – glissando effects within a sound
MORPH BRIDGE – an interpolation between 2 sounds, like CROSSFADE, but in the spectral dimension
MORPH GLIDE – a movement between 2 windows 'grabbed' from the same or different analysis files (with **SPEC GRAB**)
MORPH MORPH – full spectral morphing between 2 sounds

End of SUBMIX CROSSFADE

SUBMIX DUMMY – Convert list of soundfiles into a basic mixfile (for editing)

Usage

submix dummy mode *infile1 infile2* [*infile3..*] mixfile

Modes

1 All files start at time zero
2 Each file starts where previous file ends

Parameters

infile1 infile2 [*infile3..*] – list of soundfile names;

- **NB:** Entering the name of a text file containing a list of soundfiles on the command line is no longer supported.
- As there may be various CDP extensions for files with the same root name, **it is necessary to include the soundfile extensions**.

mixfile – output mixfile

Understanding the SUBMIX DUMMY Function

The list of soundfile names should not include the extension if the CDP_SOUND_EXT environment variable has been set – it will normally be set! The output is a mixfile with default settings, which can then be edited to suit.

SUBMIX DUMMY then turns this list of sounds into a prototype mixfile. It is not as much of a 'dummy' as it makes out: it interrogates the headers of the soundfiles to find out the number of channels and check the sample rates (they all need to be the same in a mixfile).

Musical Applications

This is a simple utility which may speed up the process of creating a mixfile. The graphic user interfaces have their own way of doing this.

End of SUBMIX DUMMY

SUBMIX FADERS – Mix several mono or stereo files using a time-changing balance function

Usage

submix faders *insndfile1 insndfile2 [insndfile3...] outsndfile balance-data envelope-data*

Parameters

insndfile1 – first input soundfile
insndfile2 – second input soundfile
[insndfile3...] – optional third and subsequent input soundfiles
outsndfile – output soundfile produced by the program
balance-data – text file containing 'value sets' to specify levels. Each value set consists of a *time* followed by the *relative-level* of each soundfile in the mix at that time. Level values entered on a given line are scaled (by the program) to add up to 1.0, so as to avoid overload. E.g., for a mixfile with 3 soundfiles:

```
[time relative levels for 3 soundfiles in the mix]
0.0     0.4 0.7 0.9
4.0     0.5 0.5 0.5
8.0     0.9 0.7 0.4
```

envelope-data – the loudness envelope to apply to the overall sound.

Balance-data by definition varies over time.
Envelope-data may be a single value or vary over time.

Understanding the SUBMIX FADERS Process

This is the equivalent of mixing several sounds using movable faders, or with imposed envelopes on each sound. One file specifies the relative balance of each input file, from moment to moment. A second value or breakpoint datafile specifies the overall level of the mix, which may be time-varying.

Musical Applications

This is a remarkably straightforward way to achieve time-varying relative level changes in a mix. Previously, each soundfile in the mix would have to be enveloped separately. It is therefore a controlled way to have level 'automation' in a mix. Also, the *mixfile* stage is by-passed, going directly to a soundfile output.

End of SUBMIX FADERS

SUBMIX FILEFORMAT – Display format of a mixfile

Usage

submix fileformat

Parameters

None

Understanding the SUBMIX FILEFORMAT Function

This function displays for your information the required format options for a mixfile. The following repeats this information for your convenience.

Mixfiles consist of lines with one of the following formats:

sndname starttime_in_mix chans level
sndname starttime_in_mix 1 level pan
sndname starttime_in_mix 2 left_level left_pan right_level right_pan

SNDNAME is the name of a Mono or Stereo soundfile. All soundfiles must have the same sample rate
CHANS is the number of channels in this/these soundfile(s) (1 or 2 only)
LEVEL is loudness expressed as a number (1 = unity gain) or as dB (0dB = unity gain)

Mono AND Stereo soundfiles may have a single level and no Pan data. In this case, Mono soundfiles in stereo mixes are panned centrally. Otherwise:

- Mono soundfiles must have 1 level and 1 pan parameter (only)
- Stereo soundfiles must have 2 level and 2 pan parameters (one for each channel)

PAN is spatial positioning of file (or file channel) in output mix

- **-1** = Hard left : **0** = Centre : **1** = Hard right
- **< -1** = Hard left & attenuated : **> 1** = Hard right & attenuated

Also note the following:

1. The *mixfile* list need not be in starttime order
2. Silence at the start of the mix is ignored (splice on afterwards, if it's needed)
3. With exclusively mono inputs and with no pan information, OR, when all soundfiles are panned hard left, or all soundfiles are panned hard right, **the output will be Mono**. All other situations produce a Stereo output.
4. **Take care when panning both channels of a Stereo soundfile**; the channel contributions sum, so, for example, if both channels are panned to the same position without attenuation, overload is possible.
5. You may test for maximum level in your mix output with the function SUBMIX LEVEL. If it necessary to reduce the mixfile level, you can do so with SUBMIX ATTENUATE.
6. You may put comment lines in a mixfile: they start with a semi-colon (';'). Blank lines are ignored.

Musical Applications

The mixfile is submitted as data to **SUBMIX MIX**.

End of SUBMIX FILEFORMAT

SUBMIX GETLEVEL – Test maximum level of a mix, defined in a mixfile

Usage

submix getlevel 1 *mixfile* [**-s***start*] [**-e***end*]
submix getlevel 2–3 *mixfile outtextfile* [**-s***start*] [**-e***end*]

Parameters

mixfile – mixfile submitted to SUBMIX GETLEVEL for level test
outtextfile – report is written to a text file
-s*start* – start position in seconds at which to begin testing the output level of the mix
-e*end* – end position in seconds at which to finish testing the output level of the mix

Modes

1 Find the maximum level in the mix
2 Find the time-locations where clipping occurs in the mix
3 Find both the maximum level in the mix and the time-locations where clipping occurs

Understanding the SUBMIX GETLEVEL Function

The mix process sums amplitudes, and this can sometimes cause levels to go above the maximum amplitude handled by the system. If these go over the maximum permissible level, they will be 'clipped', producing audible clicks in the output soundfile. SUBMIX GETLEVEL checks if this will happen by performing a pre-run through the mix operation prescribed in the *mixfile*, calculating what will happen with the amplitudes.

Besides actually finding the maximum level of a mix, SUBMIX GETLEVEL also recommends a gain factor which should eliminate the clipping. It can also report on the locations where the clipping occurred, enabling you to tell at just what point(s) in the mixfile there was a problem.

The *start end* parameters enable you to focus on particular time sections of the mix.

A useful trick to maintain high signal levels without clipping is to try inverting the phase of one or other of your source files. To do this, use a *gain* of -1 in **MODIFY LOUDNESS**.

Musical Applications

This function is a utility which can save you some time. It also helps you to pinpoint where the problem(s) occur and advises you on what attenuation is needed to remedy the problem. Furthermore, it avoids creating a possibly very large output soundfile unnecessarily.

End of SUBMIX GETLEVEL

SUBMIX INBETWEEN – Generate a set of sounds inbetween the 2 input sounds, through weighted mixes of the input sounds, from mostly sound1 to mostly sound2

Usage

submix inbetween 1 *infile1 infile2 outname count*
OR:
submix inbetween 2 *infile1 infile2 outname ratios*

Parameters

infile1 – soundfile which predominates at the beginning
infile2 – soundfile which predominates at the end
outname – **generic** name for the output soundfiles. The new soundfiles will be called *outname001 outname002* etc.
count – the number of inbetween outfiles to produce; amplitude ratios for the new sounds are generated automatically
ratios – a ratio or list of ratios in a textfile. These define the relative levels of the outfiles produced: the level of *file2* **relative** to *file1*, for each new outfile (Range: 0 to 1). (If a textfile is used, there will be as many output soundfiles as there are ratios in the textfile.)

Modes

1 Automatic relative levels for *count* inbetween files
2 User-defined relative levels, as a single ratio or as a series of ratios in a textfile.

There must be an even number of values, and they must be arranged in ascending order (i.e., getting larger: the ratio is a gain factor applied to *infile1*, copies of which are meant to become progressively softer).

N.B. Ensure that NONE of the files to be created already exists!

Understanding the SUBMIX INBETWEEN Process

This is an amazing function which enables us automatically to weight the relative amplitude of two soundfiles. The output consists of *count* actual soundfiles (or the numer of ratios provided) – not just a mixfile.

In Mode **1** the first of these output soundfiles, *infile1* is at its loudest point, and *infile2* at its softest, though the difference in level is not necessarily extreme. In the last of these output soundfiles, the situation is reversed: *infile1* is at its softest and *infile2* is at its loudest. The relative levels are calculated automatically, so will be affected by the number of soundfiles to process, as given by *count*.

In Mode **2** the ratio values represent fractional proportions of the two soundfiles. The value you enter for *ratios* is the amount of the 2nd file relative to the 1st. Thus 0.25 will be ¾ of *infile1* and ¼ of *infile2*, 0.75 will be ¼ of *infile1* and ¼ of *infile2*, and 0.9 will be 1/10th of *infile1* and 9/10th of *infile2*, etc. Any proportion can be used, but they must be placed in ascending order (i.e., getting progressively closer to 1). **NB** – don't forget that there must be an **even** number of ratios in the file.

Example file of ratios:

0.1
0.25
0.5
0.6
0.75
0.9

Musical Applications

It is legal to enter a *count* of 1, but redundant to do so. One output soundfile is produced, with an equal amplitude weighting. The result is therefore the same as **SUBMIX MERGE**.

The several output soundfiles produced can be used individually. The SUBMIX INBETWEEN process is used as a quick way to produce several soundfiles with different amplitude weightings of the two input soundfiles.

The several output soundfiles produced can then be used in a further mix. For example, their start times could be staggered, see **SUBMIX TIMEWARP**, to achieve an extended passage in which the sounds repeat and overlap, and gradually move in volume from the first to the second. This could be a rhythmic or a smooth process depending on the attack transients of the input soundfiles.

End of SUBMIX INBETWEEN

SUBMIX INBETWEEN2 – Generate a set of sounds in-between the 2 input sounds by interpolation pegged to zero-crossings

Usage

submix inbetween2 *infile1 infile2 outname count cutoff*

Parameters

infile1 – first input soundfile
infile2 – second input soundfile
outname – generic name for the various 'inbetween' output soundfiles. The new soundfiles will be called outname001, outname002 etc. **Ensure that NONE of the files to be created already exists.**
count – the number of inbetween outfiles to produce
cutoff – the frequency above which 'cycles' are ignored: usually noise, they are incorporated into other cycles.

Understanding the SUBMIX INBETWEEN2 Process

SUBMIX INBETWEEN2 applies the '*inbetweening*' process to two sounds while attempting to synchronise the zero-crossings in the two input soundfiles. It does this by stretching or contracting the waveform to fit into the appropriate timeslot.

If you put in two sinetones at e.g., 440Hz and 330Hz and ask for a single interpolation, you get a sinetone intermediate in frequency. When dealing with natural signals, you might want to avoid very short 'wavesets' which do not reflect the underlying pitch or harmonics of the sounds. To do this you set a *cutoff* (upper frequency limit, i.e., a minumum length) for the acceptance of 'true' wavesets to be used in the process. These wavesets that are declared to be 'too small' are incorporated into the adjacent 'true' waveset.

Musical Applications

This is a fantastic heuristic tool. Some extraordinarily unusual sounds can be created by this process. Don't try to hard to predict what they will be!

End of SUBMIX INBETWEEN2

SUBMIX INTERLEAVE – Interleave mono files to make a single multichannel outfile

Usage

submix interleave *sndfile1 sndfile2 [sndfile3 sndfile4] outfile*

Parameters

sndfile1 ... – mono soundfiles to interleave
outfile – resultant multichannel soundfile

Maximum number of channels in output is now 1000 (was originally 4)

First soundfile goes to left channel of stereo, or channel 1 of 4, etc.

Understanding the SUBMIX INTERLEAVE Process

INTERLEAVE produces multi-channel files by interleaving the specified mono files. It will not accept input files with more than one channel. Note that all the input files should have the same sample rate.

Musical Applications

The usual application is to create a stereo file from two mono files. See also **HOUSEKEEP CHANS** mode 4 (convert stereo to mono). **MODIFY SPACE** mode 1 (pan) will also create a stereo output from mono inputs.

End of SUBMIX INTERLEAVE

SUBMIX MERGE – Quick mix of 2 soundfiles (with the same number of channels)

Usage

submix merge *sndfile1 sndfile2 outfile* [**-s***stagger*] [**-j***skip*] [**-k***skew*] [**-b***start*] [**-e***end*]

Parameters

sndfile1 first soundfile to be mixed
sndfile2 second soundfile to be mixed
outfile resultant mix
-s*stagger sndfile2* enters *stagger* seconds after (Default: 0)
-j*skip* move *skip* seconds into *sndfile2* before starting to mix (Default: 0)
-k*skew sndfile1* has *skew* times more gain than *sndfile2* (Default 1: same level)
-b*start* – start the mix at the time specified
-e*end* – stop the mix at the time specified

Stagger and *skip* are approximated to within about one hundredth of a second.

Understanding the SUBMIX MERGE Process

Simple mixing of two soundfiles with facilities to have the second soundfile enter later (*stagger*), start some time after its beginning (*skip*), or have less gain than the first soundfile (*skew*).

Musical Applications

For very simple mixing jobs, where two input files need to be mixed, MERGE offers an easier alternative to MIX. The files may be of any length, but are expected to have similar sampling rates and numbers of channels.

If the skip argument is given, the stagger argument must also be given, although it can be zero.

The files are by default mixed at equal levels with each file scaled to half its original level. For this reason overflows will not take place when using MERGE with soundfiles of sample type SHORTS.

ALSO SEE: **SUBMIX BALANCE**.

End of SUBMIX MERGE

SUBMIX MERGEMANY – Quick mix of several soundfiles (with the same number of channels)

Usage

submix mergemany *insndfile1 insndfile2 [insndfile3 ...] outsndfile*

Parameters

insndfile1 – first input soundfile
insndfile2 – second input soundfile (number of channels must be consistent)
[insndfile3 ...] – third and subsequent input soundfiles
outsndfile – output soundfile produced by the program

Understanding the SUBMIX MERGEMANY Process

Makes a quick mix of several sounds (all having the same number of channels) with all sounds at a standard loudness level. **SUBMIX MERGE** already exists to mix two sounds in this way.

- SUBMIX MERGE avoids calculating the peak sample (in order to avoid clipping) by simply reducing each source sound to half-level before mixing.
- SUBMIX MERGEMANY, however, calculates the maximum output sample, in a first pass, and then adjusts the standard level of each input file to avoid any clipping of the output.

It is best, therefore, to use SUBMIX MERGEMANY for quick mixes of several sounds, if you want to *use* the output (rather than just hear it).

Musical Applications

This quick mix of several soundfiles with a normalising function can be used to test combinations of sounds, or to mix when all the sounds are at starttime 0. A *mixfile* is not produced, so no further adjustments are possible.

ALSO SEE: **SUBMIX MERGE**.

End of SUBMIX MERGEMANY

SUBMIX MIX – Mix sounds as instructed in a mixfile

Usage

submix mix *mixfile outsndfile* [**-s***start*] [**-e***end*] [**-g***attenuation*] [**-a**]

Parameters

mixfile – soundfiles to be mixed, with appropriate information

- Now that there may be various CDP extensions for files with the same root name, **it is necessary to include the soundfile extension in the *mixfile***.
- **NB** the names of the mixfile components (*infile* etc.) as shown in bold print in the example mixfile below are NOT included in the mixfile, just the data itself. There is no provision for comments in any CDP text or breakpoint file.

outsndfile – resultant mix of soundfiles (Mono or Stereo depending on Pan settings in the mixfile)
-s*start* – begin the mixing process at time *start* (to start mixing later than time zero)
-e*end* – stop the mixing process at time *end* (to stop the mix before its true end

Note that the *start* and *end* parameters are provided to make it possible to test what will happen with a certain portion of the mix. They are intended for use in this way, in a test context, and, if you save the output of such a test, its beginning and end may be abrupt. This can be corrected by running **HOUSEKEEP EXTRACT** Mode **3** or **ENVEL DOVETAIL** to 'top and tail' the output soundfile.

-g*attenuation* – reduce the level of the entire mix (Range: > 0 to 1)
-a – alternative mix algorithm, slighly slower, but it may avoid clipping in special circumstances

The format of a mixfile with example contents

infile	start_time	no_of_channels	ch1_level	ch1_pan	[ch2_level	ch2_pan]
soundfile1.wav	0.23	2	-6dB	L	-6dB	R
soundfile2.wav	0.9	1	-20dB	C		
soundfile3.wav	3.7	1	-4dB	-.5		

Also see **FILEFORMAT** which provides detailed information on the format requirements of the mix file. A concise but detailed summary is also given in **CDP Files & Codes**.

Understanding the SUBMIX MIX Process

MIX is used to assemble musical passages by mixing several soundfiles into one new soundfile, arranging the way they overlap by specifying the start time of each soundfile. The amplitude of overlapping soundfiles is summed, so the provision for altering the amplitude in dB enables the user to avoid exceeding the maximum (32767) which would cause distorton.

The *mixfile* contains one line of text for each file in the mix.

Note that the **level** value is used as a simple multiplier by the MIX program. This means that files may also be amplified.

It is important to be careful about dynamic levels when mixing soundfiles with high amplitudes. As a rule of thumb, always mix two files as close to 0dB (maximum level) as you can, but run **SUBMIX GETLEVEL** first to check for overload. If overload is reported, use **SUBMIX ATTENUATE** to reduce the overall level of the *mixfile*.

On the other hand, it is also possible to reduce the volume of soundfiles in an undesirable way, and experience indicates a fairly high tolerance: so do not reduce the volumes too much unless a previous run has produced distortion. **SNDINFO MAXSAMP** will display the maximum amplitude in any given soundfile, so it can be used to help determine precise values when this seems necessary. Also remember that every 6dB reduction halves the amplitude.

Pan parameters may take on values from -1.0 (left) to +1.0 (right) to place the sound in the stereo space between the loudspeakers. Alternatively, L C and R may be used for Left Centre and Right. Values greater than 1 (or less than -1) will force the sound to full Right (or Left), but unattenuated, to give an illusion of even greater rightward (or leftware) distance.

Note that by selecting values inbetween 0 and -1 or +1, soundfiles may be placed at specific locations along the horizontal plane. These Pan parameters do not, therefore, create moving sounds as does the **MODIFY SPACE** function, Mode **1** (Pan), but they do enable the composer to spread out the musical material in the aural field. This is common practice on mixing desks where the manual pan control is used to maintain the aural clarity of all the sounds being mixed. It is also possible, of course, to locate all the soundfiles of a mix in one area of the soundfield, in preparation for placing the results of a different mix in another area.

Entering Pan information as 'L' for all the sounds in the mixfile (or as all 'R') will force the creation of a mono soundfile as output.

Musical Applications

Because the results of previous mixes can be used (they are simply new soundfiles), very complex layering of musical material can be achieved. MIX can also be used to splice sounds together, as this is merely a special case of mixing (see **SFEDIT JOIN**).

The number of soundfiles which may be open at one time during a mix is limited to 1000 (!), which shows how useful this program can be. Not only can many soundfiles be mixed at one time, but the previous rigmarole of synchronising, starting and stopping several tape recorders becomes a thing of the past.

End of SUBMIX MIX

SUBMIX MODEL – Replace soundfiles in an existing mixfile

Usage

submix model *inmixfile sndfile1 [sndfile2 ...] outmixfile*

Parameters

inmixfile – input *mixfile* to alter
sndfile1 – first new soundfile to replace
sndfile2 – second new soundfile to replace
[*sndfile3 ...*] – additional soundfiles to replace
outmixfile – *mixfile* created by the program

Understanding the SUBMIX MODEL Function

This process allows you to use an existing *mixfile* as a model for a new mix using different sounds. You submit to the process the original *mixfile* together with the new soundfiles you now want to use – **you need to have the same number of soundfiles as in the original mix and in the same order)**. The original sounds are then replaced by the new sounds and a new *mixfile* generated. Note that the new sounds must have the same number of channels (and sample-rate) as those that they replace.

Musical Applications

As you continue to work, you may build up a repertoire of *mixfiles* with specific level and pan patterns. This program enables you to reuse them with different soundfiles. Alternatively, you may want to try the mix you're working on with some or all new sounds. (If some, you will need to re-enter the sounds you want to retain so that the same number of soundfiles are specified.)

End of SUBMIX MODEL

SUBMIX ONGRID –Convert listed soundfiles to a basic mixfile on timed grid (for editing)

Usage

submix ongrid *infile1 infile2 [infile3...] outmixfile gridfile*

Parameters

infile1 – first input soundfile
infile2 – second input soundfile
[infile3...] – third and subsequent input soundfiles
outmixfile – output mixfile generated by the program
gridfile – a list of *times* (one for each input sound) of the sounds in the mix,
OR
a list with some times preceded by 'x' (no space after 'x'), where 'x' marks a time to be used – other grid times to be ignored. (The number of input soundfiles must match the number of marked items.)

Numbers, or gridpoint names, may follow *times*, but they have to be on the same line.

Understanding the SUBMIX ONGRID Function

CREATE MIXFILE creates a basic mixfile in which the sounds all start at time zero, or where each sound starts as the previous sound has finished. SUBMIX ONGRID makes this more flexible by allowing you to specify the times at which your set of soundfiles are to start in the output *mixfile*.

NB: If no sound is used at time zero, the mix will skip to the first sound actually used. To avoid this, use a silent soundfile at time zero.

Musical Applications

SUBMIX ONGRID provides another quick way to create a *mixfile*. In this case, both the file names and the start times are set. That leaves only the level and pan data to be edited.

End of SUBMIX ONGRID

SUBMIX PAN – Pan a mixfile

Usage

submix pan *inmixfile outmixfile pan*

Parameters

inmixfile – input mixfile
outmixfile – output mixfile generated by the program
pan – locates the sounds in the mix at different positions between the speakers. (This is dependent on the time at which they begin. The sounds themselves are not panned, i.e., internally, during their duration.)

Pan may vary over time.

Understanding the SUBMIX PAN Function

This function allows the sounds in a mixfile to be repositioned to a given location in the stereo field, or to different locations at different times. Note that it is the start pan positions of the sounds which are (re)set by this process. Once a sound has begun to play, it remains at its start position.

The *panfile* contains pairs of numbers: a *time* and a *pan_position*:

```
[time  pan-pos]
0.0    -1.0
2.5     1.0
6.1    -0.5
8.2     0.5
11.6    0.0
```

In this case, we can assume that the original pan positions were different and that we have decided to have the successive entries of the sounds swing from left to right, gradually moving closer to the centre of the stereo field. In this first example, the **same times** as the original *mixfile* are used. The output rewrites the mixfile with the new pan positions.

We can also do it a little differently, taking advantage of the fact that the program does **interpolations** if the **times differ** from the original. What we have in mind is to have this motion converging on the centre to happen very evenly over the time of the mix. Therefore we rewrite the *panfile* with the times evenly spaced. The pan positions now placed in the output *mixfile* is *where they would be along the movement path from the previous position to the next at that particular time*. Here is the new *panfile* (in black) used as the input to **SUBMIX PAN** (the same *mixfile* with times as above is the other input). The corresponding original times are shown in red on the left and the new pan positions output by the program are shown in red on the right. (Note that the original times will still be shown in the output *mixfile*).

```
[orig-time new-time  pan-pos  output-pan-pos]
 0.0         0.0      -1.0     -1.0
 2.5         3.0       1.0      0.5
 6.1         6.0      -0.5     -0.4667
 8.2         9.0       0.5      0.2333
11.6        12.0       0.0      0.0667
```

SUBMIX PAN therefore provides very sophisticated control possibilities for your panning operations.

Remember that to get the sounds themselves to move about, they must (as usual) be panned prior to being placed in the mix, OR the output of the mix itself (provided it is mono) may itself be panned.

Also note that stereo sounds in the mix input will appear mono, but specifically positioned in space, in the output. With this process you could, for example force successive sounds to enter at positions which change gradually from stage left to stage right.

Musical Applications

- ease of pan redesign
- possibility of sophisticated positioning via interpolation
- the same *panfile* could be used with **block processing** in *Sound Loom* to alter several *mixfiles* at once, as long as they had the same number of sounds / start times.

ALSO SEE: **MODIFY SPACE** Mode 1 (PAN).

End of SUBMIX PAN

SUBMIX SHUFFLE – Shuffle the data in a mixfile

Usage

submix shuffle 1–6 *inmixfile outmixfile* [**-s** *startline*] [**-e***endline*]
OR
submix shuffle 7 *inmixfile outmixfile newname* [**-s** *startline*] [**-e***endline*] [**-x**]

Parameters

inmixfile – input mix function file
outmixfile – output (shuffled) mix function file
-s*startline* – line of *inmixfile* at which to start shuffling (default is 1st line)
-e*endline* – line of *inmixfile* at which to end shuffling (default is last line of file)

Modes

1 Duplicate each line
2 Reverse order of filenames
3 Scatter order of filenames
4 Replace sounds in selected lines with sound in startline
5 Omit lines (closing up timegaps appropriately)
6 Omit alternate lines (closing up timegaps appropriately)

In Modes 5 & 6, the mix must be in the correct time order. (Mixfiles can be time-ordered using **SUBMIX TIMEWARP** Mode 1)

7 Duplicate and rename: duplicate each line with a new sound, new name

SUBMIX SHUFFLE checks that 'newname' is a compatible soundfile for the mix, but note that the **-x** flag turns off 'newname' checking in Mode 7. WARNING: SUBMIX SHUFFLE cannot handle a mixture of mono and stereo files.

Understanding the SUBMIX SHUFFLE Function

There are 3 sets of mix-shuffle functions, all dealing with mixfile data. **SUBMIX SPACEWARP** handles the spatial location of the sounds, and **SUBMIX TIMEWARP** handles their onset times. **SUBMIX SHUFFLE** itself massages lines in the mixfile and filenames.

Musical Applications

The purpose of these functions is to enable the user to adjust a (usually complex) mixfile in a quick and easy manner. SUBMIX SHUFFLE can duplicate filenames in a mix, or replace all the names with that of the first sound. Omitting lines thins out the mix, reversing the order of the names reverses the sequence of a whole series of events, but each event still goes forward (i.e., not just a soundfile played backwards). Scattering the order of soundfiles loosens up the mix and can be quite handy when making sonic collages or landscapes.

Only one option can be entered at once, but this should not be a problem. The operation of the program is very fast, as it is only writing a short textfile. A variety of changes can be carried out in sequence by using two *outmixfile* names: write the first change to *outmix1*, then use *outmix1* as the input for the next run of SUBMIX SHUFFLE, naming the *outmixfile outmix2*. On the next run, *outmix2* is the *inmixfile* and *outmix1* is the *outmixfile*, the original version being overwritten by the program, etc.

End of SUBMIX SHUFFLE

SUBMIX SPACEWARP – Alter the spatial distribution of a mixfile

Usage

submix spacewarp 1–2 *inmixfile outmixfile Q* [**-s***startline*] [**-e***endline*]
OR: **submix spacewarp 3–6** *inmixfile outmixfile Q1 Q2* [**-s***startline*] [**-e***endline*]
OR: **submix spacewarp 7** *inmixfile outmixfile*
OR: **submix spacewarp 8** *inmixfile outmixfile Q*

Parameters

inmixfile – input mix function file
outmixfile – output (shuffled) mix function file
Q – position along horizontal spatial axis (Range: -1 to 1)
-s*startline* – line of *inmixfile* at which to start shuffling (default is 1st line)
-e*endline* – line of *inmixfile* at which to end shuffling (default is last line of file)

Modes

1 Sounds to same position – *Q* is position (stereo files become mono)
2 Narrow spatial spread – *Q* is a positive number < 1
3 Sequence positions leftwards – over range *Q1* to *Q2* (stereo files become mono)
4 Sequence positions rightwards – over range *Q1* to *Q2* (stereo files become mono)
5 Random-scatter positions – within range *Q1* to *Q2* (stereo files become mono)
6 Random-scatter positions, but alternate to the left and right of the centre of the spatial range specified – range *Q1* to *Q2* (stereo files become mono)
7 Invert stereo in alternate lines of mixfile – use to avoid clipping
8 Invert stereo in specified line of mixfile – *Q* is line number

Understanding the SUBMIX SPACEWARP Function

SUBMIX SPACEWARP massages the spatialisation information in existing mixfiles, presuming that there are several sounds involved. It provides a number of 'preset' operations to focus the sounds, move them gradually left or right, scatter them about the aural field, or switch the stereo channels (left to right, right to left).

Modes **7** and **8**, to 'invert stereo' can be used to avoid clipping which can occur because too many sounds are in one part of the aural field (the amplitudes can accumulate).

Musical Applications

The spatial placement of sounds in a mix is part of the art of 'orchestrating' in this type of medium. If all the sounds are in the same position, they will tend to fuse more than if they are spread out in the aural space. Creating different spatial locations for sounds is therefore a way to clarify the aural image. It can also be used to move sounds along the horizontal trajectory or to have them appear in varying (random) locations. The latter might be especially suitable for complex collages or for sound textures which emulate a 'natural' environment.

End of SUBMIX SPACEWARP

SUBMIX SYNC – Synchronise soundfiles in a mixfile, or generate such a mixfile from a list of soundfiles

Usage

submix sync mode *intextfile outmixfile*

Parameters

intextfile – either an existing mixfile or the name of textfile containing only the names of soundfiles. If CDP_SOUND_EXT is set, the extension does not have to be specified.
outmixfile – output mixfile generated by the program

Modes

1 Synchronise soundfile midtimes
2 synchronise soundfile endtimes

Understanding the SUBMIX SYNC Function

This function aligns soundfiles either at the mid-point in the duration of each sound, or at their endings (like a 'right justify' in word processing). This is a calculation based on duration only, and does not take any account of audio data. It just adjusts the start-times of all the files in order to achieve the alignment specified by the Mode and writes a mixfile with these timings.

It actually looks at the header of each soundfile when preparing the mixfile. Thus it can include the number of channels in the output mixfile when provided only with a list of soundfile names. But note that this mixfile may need some editing of its level and pan parameters, which are given the default settings of '1.0' and 'C' respectively for all soundfiles.

Musical Applications

SUBMIX SYNC helps block out a mixfile when the sounds are aligned at their mid-points or their endings. It is also a way to create a rough mixfile quickly, and can therefore be used just to create a template mixfile to edit.

ALSO SEE: **SUBMIX SYNCATTACK** and **SUBMIX DUMMY**

End of SUBMIX SYNC

SUBMIX SYNCATTACK – Synchronise the attacks of soundfiles in a mixfile, or generate such a mixfile from a list of soundfiles

Usage

submix syncattack *intextfile outmixfile* [**-w***focus*] [**-p**]

Parameters

intextfile – an existing mixfile or a list of soundfiles. If CDP_SOUND_EXT is set, the soundfile extension does **not** have t o be specified.

> When using a list of soundfiles (only), each soundfile name may be followed by 2 times (in seconds); these times limit the search area for the sound's attack. (See below!)

outmixfile – output mixfile generated
-w*focus* – a fixed integer factor which can be used to shorten the window which scans for the attack. Range:**1** (max window size, min *focus*) – **5** (min window size, max *focus*). **-p** – find the peak power segment before locating its maximum sample. Default: only look for the maximum sample.

> SUBMIX SYNCATTACK estimates the output levels required to prevent clipping. However, this estimate may be over-cautiously low, so you may need to adjust the levels. **SUBMIX ATTENUATE** can be used to adjust the overall level of a mixfile.

Understanding the SUBMIX SYNCATTACK Function

SYNCATTACK means 'synchronised attacks'. This is synchronisation based on audio content, 'connecting' soundfiles at their respective points of high amplitude. SUBMIX SYNCATTACK by default looks for the highest amplitude in each file, and lines up the sounds accordingly. This point could be near the beginning, later on in the file, or within the times specified.

Very often, for example, sounds played on wind instruments 'swell': the amplitude increases as the tone continues. This means that the peak point may occur considerably later on as the sound continues. If this is the case, you will find that SYNCATTACK will bring in another sound when this peak occurs. To get the sounds to synchronise at the beginning, specify times in *intextfile* which will locate the search range for the peak at/near the beginning of the sound(s) which get louder later: e.g., times 0 and 0.1.

The 'peak power' option looks for the largest region of high amplitude rather than just the highest amplitude (which could have a very short duration).

This process is well-suited for use with vocal sounds, the default maximum amplitude sync point works best for consonants (short durations), while the peak power option (**-p**) works best for vowels (longer durations).

It automatically creates a mixfile, which is used to mix the sounds into a new, synchronised, soundfile.

Focus enables the user to control the size of the window in which the attack is to be found, i.e., to tighten the focus on the precise attack moment.

The datafile lists the soundfiles to be synchronised, with optional start and end times for each file search.

The mixfile is a text file in mixfile format generated by the program. Run this with **SUBMIX MIX** to create the final, synchronised soundfile. It is usually a good idea to look at the *mixfile* before using it in order to check that the levels and pan settings are as you would like them to be for those particular soundfiles. If you hear a click at the entry of one (or more) of the sounds, you will have to reduce the level(s) in the mixfile produced by **SUBMIX SYNCATTACK** before running **SUBMIX MIX** to carry out the mix. **SUBMIX GETLEVEL** will report any possible overloads in the mix, while **SUBMIX ATTENUATE** can be used to reduce the overall level of the mixfile.

Musical Applications

The basic default operation of this function is a way to assemble sounds in a very smooth and natural way by aligning them at their respective loudest points. This means that they all peak at the same time – and more often than not, their respective beginnings will occur before this point.

SUBMIX SYNCATTACK can also be used to create an aural 'mask': with the loud portion of one file 'covering' the entrance of the next sound. If this doesn't happen through the default operation of SUBMIX SYNCATTACK, it can be achieved by using the times mechanism to locate the search range at the beginning of the sound.

For example, two sounds may swell a bit, but you want one of them to come in, to begin, at the high point of the other. You therefore enter times such as 0 0.1 in *intextfile* after the name of the file you want to sync at its beginning. The result is this: you hear the first sound begin and swell in volume. At its peak, the second sound enters, probably with a masking effect, because its beginning is a bit quieter relative to what occurs later. Thus its entry will be 'covered' by the peak of the first sound.

As mentioned above, synchronisation can be forced to the beginning of the sounds by specifying times in *intextfile*.

ALSO SEE: **SUBMIX SYNC**

End of SUBMIX SYNCATTACK

SUBMIX TEST – Test the syntax of a mixfile

Usage

submix test *mixfile*

Parameters

mixfile – full name of mixfile to be tested

Understanding the SUBMIX TEST Function

SUBMIX TEST checks both that all the soundfiles listed in it are valid soundfiles and that the syntax of the *mixfile* is correct.

In checking the validity of the soundfiles, it actually looks for, needs to find, and reads the header of the soundfiles: so they must be present in the same directory as the *mixfile*. This is more than a check for mis-spelling the name of the file. There are many different types of file which might be present, possibly with a .wav extension, which will not be soundfiles: such as analysis files, binary envelope files, binary formant files, and binary pitch data files – not to mention breakpoint text files, which may or may not have an extension. You might forget which is which, so SUBMIX TEST actually looks into the header of the file to double-check its type.

NB – To minimise this kind of ambiguity about files and to make it easier to name them, Release 4 supports the use of different file extensions according to the following conventions:

Filetype	Extension
sound	**.wav/.aif**
analysis	**.ana/any**
binary formant	**.for**
binary envelope	**.evl**
binary pitch	**.frq**
binary transposition	**.trn**
text breakpoint	**any/none** (usually **.txt** and **.brk** respectively)
text MIDI tuning	**.tun**
shortcut	**.lnk**

We have also added a filetype field to the **DIRSF** report. It is labelled 'fmt', which stands for 'format'. Under this heading, DIRSF displays 'W' for .wav or 'A' for .aif, because any of the above binary files could be .wav or .aif, which is not indicated by the extension.

The syntax checks will pick up errors such as an invalid number of channels, a channel count which doesn't match the soundfile, left and right level or pan data for sounds which are only mono, or only one set of data for a stereo sound, or a missing start time (reports an invalid channel number because it's reading the level data in the channel position). It doesn't check the validity of amplitude levels.

Musical Applications

This is a useful tool when mixes are large and complex.

End of SUBMIX TEST

SUBMIX TIMEWARP – Timewarp the data in a mixfile

Usage

submix timewarp 1 *inmixfile outmixfile Q*
OR: **submix timewarp 2–5** *inmixfile outmixfile Q1* [**-s***startline*] [**-e***endline*]
OR: **submix timewarp 6–16** *inmixfile outmixfile Q2* [**-s***startline*] [**-e***endline*]

Parameters

inmixfile – input mix function file
outmixfile – output (shuffled) mix function file
Q – position along horizontal spatial axis (Range: -1 to 1)
-s*startline* – line of *inmixfile* at which to start shuffling (default is 1st line)
-e*endline* – line of *inmixfile* at which to end shuffling (default is last line of file)

Modes

1 Sort into time order
2 Reverse timing pattern – e.g., a ritardando of sound entries becomes an accelerando
3 Reverse timing pattern & order of filenames
4 Freeze timegaps – between sounds at **first** timegap values
5 Freeze timegaps & names – same as Mode **4** and all files take the name of the first soundfile
6 Scatter entry times – around the original time values. *Q* is scattering (Range: 0 to 1)
7 Shuffle up entry times – shuffle times in mixfile forward by time *Q* seconds
8 Add to timegaps – add fixed duration *Q* seconds to all timegaps between sounds
9 Create fixed timegaps 1 – same gap between all sounds, timegap = *Q* seconds
10 Create fixed timegaps 2 – expanding: starttime + *Q*, original time + 2*Q*, etc.
11 Create fixed timegaps 3 – expanding: starttime * (1+*Q*), original time * (1+2*Q*), etc.
12 Create fixed timegaps 4 – expanding: starttime * *Q*, original time **Q***Q*, etc.
13 Enlarge timegaps 1 – multiply original times by *Q*
14 Enlarge timegaps 2 – by adding *Q*, 2*Q*, etc.to each successive time
15 Enlarge timegaps 3 – multiply original times by (1=*Q*), (1+2*Q*), (1+3*Q*),etc.
16 Enlarge timegaps 4 – multiply original times by *Q*, *Q***Q*, *Q***Q***Q*, etc. (Care!)

Understanding the SUBMIX TIMEWARP Function

The focus here is on the start times for each of the sounds in the mix. With SUBMIX TIMEWARP the several times in the mixfile can be adjusted in a single operation, according to the 'presets' available. The times can be sorted, all made the same, or all enlarged by the same amount, scattered in time (not far, just some 'jitter'), progressively expanded with reference to the start time, or each time in the mixfile can be progressively expanded.

Non-mathematicians: take special note of the results of operations carried out on values less than one. For example, repeated (recursive) multiplications of 2 numbers, both of which are less than one soon produces very tiny values: e.g., .2*.2 = .04, .04*.2 = .008 etc.

Musical Applications

The purpose of these functions is to enable the user to adjust a (usually complex) mixfile in a quick and easy manner. For example, you may have found the mix too aurally dense and want to spread it out a little. In this case you could for example enlarge the timegaps a little and perform the mix again.

Mode **13** can be seen as a simple way to change 'tempo'. We can illustrate this with the following 3 steps:

1. Suppose we create a mixfile with 3 soundfiles by using **SUBMIX DUMMY** Mode 1, in which all soundfiles are set to start at time 0. (Remember to include the soundfile extensions when you enter the name of each soundfile – they have to be explicitly included).
2. We want to stagger their entries by a regular timegap. Let's make this 1 second, in the sense that 1 second will equal a tempo of crotchet (quarter note) = 60. We can do this with SUBMIX TIMEWARP Mode **8** which adds a constant value to each start time. So we would enter by GUI or command line: `submix timewarp 8 mix3dum.mix mix3dum2.mix 1`. Now the first soundfile in the new mixfile starts at time 0.0, the second at time 1.0 and the third at time 2.0 – a regular tempo at crotchet = 60.
3. We want to increase this to crotchet = 76. The formula for the change is `original_tempo / new_tempo`, in this case `60 / 76 = 0.83`.
4. Now we go to SUBMIX TIMEWARP Mode **13** (multiply by a constant value) and use 0.83 as the multiplier: `submix timewarp 13 mix3dum2.mix mix3dum3.mix 0.83`. Now the first soundfile will begin at time 0.0, the second at time 0.83 and the third at time 1.66, so they enter at a tempo of crotchet = 76.

   This illustration, hopefully, gives some clues as to how to make use of the other modes of SUBMIX TIMEWARP.

   SUBMIX TIMEWARP is, in effect, applying a bit of algorithmic processing to the times in a mixfile. Although limited to what are in effect a set of 'presets', its functions nevertheless comprise a useful and time-saving set of routines to focus, regularise, or varyingly expand the interval between the start of each sound in the mix. It points towards further possibilities of score processing/generation by user-defined algorithmic means. The interesting questions are what processes are important musically, and how does one make the output musically supple.

End of SUBMIX TIMEWARP

Last Updated 28 Mar. 2016
Documentation: Archer Endrich, revised R. Fraser