CDP TEXTURE Functions

(with Command Line Usage)

Functions to make multi-event textures with soundfiles

~ The Texture Pack contains batch and data files for all the examples ~
NB: – See **Special Background Information**

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

SIMPLE
A texture of events shaped by random selections from parameter ranges, with one or more input sounds; a Harmonic Field/Set may be used

GROUPED
A texture of separate event groups shaped by random selections from parameter ranges, with one or more input sounds; a Harmonic Field/Set may be used

DECORATED/PREDECOR/POSTDECOR
A texture of 'decoration' events shaped by internal grouping parameters, selected at random from a pitch range or from a Harmonic Field/Set and attached to an underlying 'line'; one or more input sounds. [A Line with random-range-shaped Decorations, Harmonic Field/Set optional.]

MOTIFS/MOTIFSIN
A texture of fully defined motifs attached to pitches selected at random from a pitch range or from a Harmonic Field/Set (MOTIFS: only first note of motif constrained, MOTIFSIN: all notes of motif constrained); one or more input sounds. [A Motif, Harmonic Field-Set optional – no Line.]

ORNATE/PREORNATE/POSTORNATE
A texture of events with fully user-specified ornaments placed on a 'line', optionally with pitches restricted to a Harmonic Field/Set; one or more input sounds. [A Line with fully defined Ornament, Harmonic Field-Set optional.]

TIMED
A texture with events constrained to a rhythmic template and pitches selected at random from a pitch range or a Harmonic Field/Set; one or more input sounds

TGROUPED
A texture with the onsets of separate internally shaped event groups constrained to a rhythmic template, with pitches drawn from a pitch range or a Harmonic Field/Set; one or more input sounds

TMOTIFS/TMOTIFSIN
A texture with the onsets of fully user-specified motifs constrained to a rhythmic grid and attached to pitches drawn from a pitch range or from a Harmonic Field/Set; one or more input sounds

Background Info:
Special background information for Texture

Also see (multi-channel):

[TEXMCHAN]
Create textures over a multi-channel frame

[NEWTEX]
Generate a texture of grains made from a source sound or sounds

[WRAPPAGE]
Granular reconstitution of one or more soundfiles over multi-channel space

Author's Note

This is an extraordinary program set devised and written by Trevor Wishart. It enables you to build and shape multi-event textures of varying complexity by semi-algorithmic mixing processes. My work on the CDP HTML Reference Documentation took several months, spread out over 2 years, and was followed by writing the supplementary Texture Workshop to provide a more step-by-step introduction to the musical potential of the Texture programs. There is also a section in the *12-Step Tutorial* that summarises the key musical possibilities of the TEXTURE programs. I hope you enjoy using them as much as I do.

– Archer Endrich, May 2005.

Special Background Information for Texture

Given the complexity of these programs, you are recommended to take advantage of the Presets in *Soundshaper*, or the Patches function in *Sound Loom*, or the batch file mechanism when using command lines. All the relevant files are in the **\txpack** folder. Once set up, they can be varied without too much difficulty.

The Texture Pack contains *Sound Loom* Patches, or batch and text file versions of all the parameter settings used in the examples, along with the supporting node data and breakpoint files. The batch files can be used with the CDP executables, as well as with the *Sound Loom* batchfile mechanism – but it is easier to use the **Presets / Patches**. *Soundshaper* has presets for all the Texture Pack examples. *Readtxtu.txt* summarises all the files in the Texture Pack so you know what is what.

The Texture Pack focuses on pitch-oriented applications of the TEXTURE programs, as these nicely demonstrate the many options available. Bear in mind, however:

- There are many spatialisation possibilities. All the examples here scatter the output across the stereo field. But it is possible to generate sound output at a specific location, as a moving stream, as a gradually spreading event, and so on, using the spatial position and spread parameters.
- Motifs, and harmonic fields can be tuned to any scale or system you want. If you need 500 irregularly spaced notes per octave, that's fine. Fractional MIDI values can be used to specify any tuning.
- Textures can be as dense as you want, and the variation of density can be used to obscure or unveil recognition of the source, the source pitch, and so on.

In the examples below, the pitches of motifs, ornaments and harmonic fields, specified in the *note data* file, are taken from the standard 12-note tempered scale (by using *whole number* MIDI values), but this is not essential. **You may choose any tuning system you wish**, simply by using *fractional* MIDI values for the note specifications. Ornaments and the line they ornament could be in different tuning systems. And you may define any number of notes (e.g. 500) within an octave.

In *neutral* modes, no tuning system at all is imposed on the notes, which are taken from the whole pitch continuum.

TOPICS:

Pitch Grouping Types

- **'Ornaments'** and **'Motifs'** have user-specified pitch shapes (in the *notedata* file)
- **'Decorations'** and **'Groups'** are random assemblies of pitches
- **'Harmonic sets'** use only the pitches specified
- **'Harmonic fields'** duplicate the specified pitches in all octaves

The Special Parameters *gspace*, *contour* and *centring*

- ***gspace*** – 5 options for the spatialisation of event-groups
  0. – still
  1. – scattered (default)
  2. – towards texture centre
  3. – away from centre
  4. – follow texture motion (only if spatial position varies in time)
  5. – contrary to texture motion (only if spatial position varies in time)
- ***contour*** – 9 options for the amplitude contour of groups
  - **0 –** mixed (default)
  - **1 –** crescendo
  - **2 –** flat
  - **3 –** decrescendo
  - **4 –** crescendo or flat
  - **5 –** crescendo or decrescendo
  - **6 –** decrescendo or flat
  - **7 –** directed-to-event (in some cases only)
  - **8 –** directed-to-event or flat (in some cases only)
- ***centring*** – 7 options for how the decoration pitches centre on decorated line pitches
  - **0 –** centred (default)
  - **1 –** above
  - **2 –** below
  - **3 –** centred and above
  - **4 –** centered and below
  - **5 –** above and below
  - **6 –** centred and above and below

  In all cases except 0, the pitch range of decorations are shifted to tally with line pitch.

About the Note data file

Notedata is a textfile containing:

1. MIDI 'pitch', which may be the real or arbitrary pitch of EACH input sound, specified on the first line. 60=original pitch.
2. NOTELIST(S), where necessary (as below)

Each **Notelist** is specified as follows:

```
#N
time instr_no pitch amplitude duration
time instr_no pitch amplitude duration
time instr_no pitch amplitude duration
time instr_no pitch amplitude duration
etc.
```

N = the number of notes in this notelist, and is followed by *N* lines as above
time – the start time of that note in seconds
instr_no – CURRENTLY INACTIVE IN ALL CASES; to be either implemented or removed. It is intended to provide a means of associating a specific note-event with a specified infile, but may be overridden by other parameters. Always give '1' as the

instrument number until a future update changes this situation.
pitch – within the MIDI range of 1 to 127
amplitude – within the MIDI range of 1 to 127; this field is inactive except for motifs and ornaments
duration – duration (sustain-time) of that note in seconds; this field is inactive except for motifs and ornaments

In the following example, the **1st** line provides the 'assumed' (i.e., possibly arbitrary) pitch level of the input soundfile. If there is more than one input soundfile, a pitch level must be provided for each one, separated by spaces. The **2nd line** indicates how many notes are in the notelist, and then the following lines define **each note** in turn. In the example file below, all the times in the note data file are given as zero. The timing between events (and therefore the number of events) is in this case controlled by the *packing* parameter. All durations (sustain-times) in the note data file are zero in this case, and the duration of any given note event is selected at random from within the duration range in the command parameters. In this example events of various durations are placed randomly on the lattice of a C-minor 7th chord. **Durations must, however, be specified in the note data file for motifs and ornaments.** Durations are also referred to as 'sustain-times' because they refer to the amount of time that the input soundfile is 'sustained': i.e., how much of it is used. Each note event in the texture begins at the beginning of the input soundfile and continues – is sustained – for the amount of time specified. This makes the nature of the attack portion of the input soundfile particularly important when creating textures.

The note data file is a text file which may be written with a text editor, with an editing facility within a graphic interface. There are newlines at the end of each line and spaces (or tabs) between the data items for each note. The following illustrates the format of a typical note data file.

```
60
#4
0 1 60 0 0
0 1 63 0 0
0 1 67 0 0
0 1 70 0 0
```

Notelists represent **in this order**:

a. Notelist of notes in any timed, ornamented or decorated **line** in the texture (basic substructure notelist)
b. Notelist of notes in any **harmonic field(s)** or **harmonic set(s)** specified (decoration field)
c. Notelist(s) of notes in any **ornament(s)** or **motif(s)** specified

Not all of these lists are used for the various TEXTURE functions, and the order may vary. The note data files shown in each function are more specific regarding their contents. Also see the **Note Data File Chart** and **2 additional reference charts** for overviews of key information. You can print these out (and laminate them) for easy reference. Also useful is the Chart of Equivalent Pitch Notatations – a special version named *notechpr.htm* has been created for printing neatly on two sheets, single or double-sided.

Also note:

a. Timed textures require a sequence of timed notes ...
   - Their pitches and timings will be used to define the 'line' (= nodal substructure) in **ornamented** and **decorated** textures
   - Their timings ONLY will be used in **timed**, **tmotifs** and **tmotifsin** textures (Specify arbitrary pitches).
   - Their durations may be used in certain options to specify durations of **motif** or **ornament** note-event shapes.

b. For more than 1 **harmonic field** or **harmonic set** ...
   - Specify the 1st field with all notes at time 1
   - Specify the 2nd field with all notes at time 2
   - etc. ...

c. Times *within motifs* must increase (or remain the same during chords)

TEXTURE SIMPLE – A texture of events shaped by random selections from parameter ranges, with one or more input sounds; a Harmonic Field/Set may be used

Usage

texture simple mode *infile [infile2] outfile notedata outdur packing scatter tgrid sndfirst sndlast mingain maxgain mindur maxdur minpich maxpich*
[**-a***atten*] [**-p***pposition*] [**-s***spread*] [**-r***seed*] [**-w**]

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

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line. In Mode **5** this is all that is needed. 60=original pitch.
2. In Modes **1 - 4** there is also a NOTELIST to define a Harmonic Field/Set, specified thus:
   - **#***N* (where *N* is the number of pitches in the notelist).
   - This is followed by *N* lines in the format:
     time (secs) *infile_no pitch* (MIDI) *amp* (MIDI) *dur* (secs)
   - *Amp* and *dur* are inactive.
   - Any times within the field must increase (or remain the same, during chords)

Form: **MPV/–/–/–**		Form: **MPV/HF-S/–/–**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
		#*N*	No. of lines to follow
		time instr pitch amp dur	List of Harmonic Field/Set

outdur – minimum duration of the *outfile*
packing – (average) time in seconds between event onsets
scatter – randomisation in seconds of event onsets (Range: 0 to 10)
tgrid – minimum step in milliseconds or quantised timegrid (for group start times) (Range: 0 to 10000; Default: 0)
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds). Note that this is a time-varying parameter:

- Enter 1 for both *sndfirst* and *sndlast* to use just one input soundfile.
- When you use more than one sound, the names of these input soundfiles are given to the program. The note data file must provide a reference pitch level for each of these sounds. If you just enter, e.g., 1 for *sndfirst* and 3 for *sndlast*, the program will select one of the three input soundfiles at random for each event in the texture.
- To specify when each soundfile is to be used in creating the output texture, a text file is needed to provide a *time sound_number* pair of values, the numbers corresponding to the order in which the soundfiles have been submitted to the program.
- For one file at a time changing at different times in the output (the times given are as in the output soundfile), use the same file for both *sndfirst* and *sndlast*, e.g.,

```
sndfirst  sndlast
 0.0  1    0.0  1
10.0  2   10.0  2
18.8  3   18.8  3
```

- For a changing number of files, such as expanding the number of files being used at any one time (illustrated below), use a lower number at a given time for *sndfirst*, and a higher number at the same time (or an overlapping time) for *sndlast*.

```
sndfirst  sndlast
 0.0  1    0.0  1
10.0  1   10.0  2
18.8  1   18.8  3
```

  Note that the rate at which each soundfile is employed will slow down as more are invoked over the same time period.

mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
mindur, maxdur – minimum and maximum duration of events in texture
minpich, maxpich – minimum and maximum pitch (MIDI note value)
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values

Understanding the TEXTURE SIMPLE Process

Getting started
Please make sure you have gone through the background information before continuing.

This Reference Document for the TEXTURE suite contains numerous worked examples which you can hear directly from this manual. They are stored in **\txsnds**. You can also go to the **\txpack** folder, where you will find all the files you need run to create the sound examples yourselves. There are two versions: one for the Command Line and *Soundshaper* environments and one for the *Sound Loom* environment.

What you need from the **txpack** are the note data and breakpoint files, as well as the two source sounds, *horn.wav* and *marimba.wav*, **or source sounds of your choice**. The parameter information is stored in **Presets** supplied with *Soundshaper* and **Patches** supplied with *Sound Loom*. In a Command Line environment, you can run the batch (.bat) files supplied in **txpack**.

Location of the source sounds and other files:

- *Soundshaper* – In OPTIONS > SETTINGS go to **File** in the top left corner and open **txpack.cfg**. This Will take you to **\txpack**, which contains the source sounds, note data and other files needed to run the examples. Also load the **Presets** file **Presets.dat**, which is the default file supplied with the Program).
- *Sound Loom* – select **txpack** as your working directory, **Grab** everything and place the files on the **Workspace**. Now use **Choose Files** to select the appropriate sooundfile(s) for processing. The Patches go in the **_cdpatch** folder, and should be there already in your distribution.
- Command line environment – 'change directory' to **\txpack** and run the various **batch** files, e.g., **simplexs.bat** for the TEXTURE SIMPLE examples.

The complete list of files is given in *Readtxtu.txt*. The list at the date of this file is as follows:

Note data files:

```
ndfsim1.txt     ndfsim2.txt
ndfsim3a.txt    ndfsim3b.txt
ndfsim4a.txt    ndfsim4b.txt
ndfsim5.txt     ndfsim6.txt
ndfdec1.txt     ndfdec2a.txt    ndfdec2b.txt    ndfdec3.txt
ndfmot1.txt     ndfmot2.txt     ndfmot3.txt     ndfmot4.txt
ndforn1.txt     ndforn2.txt     ndforn3.txt     ndforn4.txt     ndforn5.txt
ndftim1.txt     ndftim12.txt    ndftim3.txt
ndftgr2.txt
ndftmo1.txt     ndftmo2.txt
```

Breakpoint files:

```
packchng.brk    simplpak.brk    grppack.brk
gprlo.brk       gprhi.brk
dm5gprhi.brk    dx3gprhi.brk
Names for extra breakpoint files for ORNATE examples,
to make yourself.  These are described in ORNATE section:
oex3mulo.brk    oex3muhi.brk
oex3mulb.brk    oex3muhb.brk
oex3skip.brk    oex3skpa.brk    oex3skpb.brk
```

Please check your *Soundshaper* and/or *Sound Loom* reference manuals on handling Presets and Patches. If you are a *Sound Loom* user, for your convenience, we also supply *ctextusl.htm*, a version of this document which is *Sound Loom* specific, with all parameters as named in *Sound Loom*. Also note that *Sound Loom* Version 5.6 and above can run batch files.

First example
Mode **5** is a good place to begin. In this Mode, the note data file contains only one value: the (real or arbitrary) MIDI pitch of the input soundfile. *There is no further data because there is no list of notes.* All the note events are generated from the information of the command parameters.

To get a feel for the 'lie of the land', we can run TEXTURE SIMPLE in Mode **5** with the following parameters (*ndfsim1.txt* has only the number 60 in it). The *infile* is the horn.wav sound used in the *GrainMill* Tutorial (2.9 sec). The 'b' example has time varying packing, as defined in *packchng.brk*:
```
0 0.025
3 0.1
6 0.05
9 0.25
12 0.025
```

SIMPLE Examples 1a/b (Presets/Patches **simplex1a** and **simplex1b**):
outdur (12) packing (0.25) scatter (0) tgrid (0) sndfirst (1) sndlast (1) mingain (36) maxgain (84) mindur (0.2) maxdur (1.5) minpich (60) maxpich (60) – in command line format:
```
texture simple 5 horn simplex1a ndfsim1.txt 12 0.25 0 0 1 1 36 84 0.2 1.5 60 60
texture simple 5 horn simplex1b ndfsim1.txt 12 packchng.txt 0 0 1 1 36 84 0.2
1.5 60 60
```

You will hear a regularly occuring randomised selection of mostly overlapping note events, some longer and some shorter, repeating the pitch Middle C (randomly) in both channels – note the Left-Right movement between speakers. Try making *maxpich* 61, and then 64. With *maxpich* still at 64, try reducing the *packing* to 0.025.

All the examples for TEXTURE SIMPLE can be made at once by running *simplexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *simpldel.bat*.

Now let's start to look more closely at the salient parameters:

- *packing* determines the density of the texture by setting the time onsets of the note events. In the above example, 0.25 sec sets 4 events per second, i.e., semiquavers at crotchet (quarter note) = 60). These events will occur evenly, because both *scatter* and *tgrid* are set to zero.

- *mindur* & *maxdur* set the range within which the duration of each note event will be randomly chosen. If only one duration is wanted, set both of these to the same value.

- *minpich* & *maxpich* set the range within which the pitch of each note event (in MIDI note values) will be randomly chosen. If only one pitch is wanted, set both of these to the same value. Given a pitch range greater than just one pitch, the program selects pitches according to a random function. **NB:** The specification of MIDI pitch values is not limited to integers. Thus a pitch range of 60 60.75 will result in microtonal detuning.

OK, now let's note and experiment carefully with the two additional timing parameters: *scatter* and *tgrid*.

- The onset timing can be somewhat randomised by utilising the *scatter* parameter. For example, if in the above example *scatter* is set to 0.07 and *tgrid* is still 0, you will hear how the previous evenness is somewhat displaced.
- If you then set *tgrid* to 330ms (*scatter* still at 0.07), a quantisation grid of about a 3rd of a second is set. In a 'snap to grid' fashion, this will draw the still somewhat randomised note events towards these 3rd of a second 'grid lines'. Because quantisation sets a **time grid**, when there are more events, they will tend to bunch at the grid points, some virtually simultaneous such that they sound like one event.
- If *scatter* is then set to 0 (with *packing* still at 0.25 and *tgrid* at 330, i.e., if the onset randomisation factor is eliminated, the note events will much more regularly occur on/near the 3rd of a second time grid points.

The following table summarises some basic combinations:

TEXTURE SIMPLE Mode 5 Examples

Running some changes on the initial example				
outdur	***packing***	***scatter***	***tgrid***	**Result**
12	0.25	0	0	48 events will occur regularly on the ¼ second
12	0.25	0.07	0	48 event onsets will be displaced, becoming a little uneven rhythmically
12	0.25	0.07	330	48 events somewhat irregularly placed will gather at the 3rd of a second quantisation time grid points – but some will be doubled.
12	0.25	0	330	48 events will gather at the the 3rd of a second quantisation time grid points
Some other generic illustrations				
outdur	***packing***	***scatter***	***tgrid***	**Result**
12	1	0	500	12 events will occur on the second: quantisation is 'pulled' to the 1 second 'boundary'
12	1	1	500	12 events on the second or half-second: e.g., 1, 2, 2.5, 4 etc.
12	1	0	2000	6 double events every 2 seconds: *packing* demands 12 events: *tgrid* forces them to a 2 sec grid
12	0.5	0	330	24 events will occur. They will be shifted to the 3rd of a second grid.
12	0.1	0	330	The many events will group at the 3rd of a second grid.
12	0.1	0	150	The slighter quantisation here will reduce the bunching of the previous example

You are recommended to experiment with TEXTURE SIMPLE Mode **5** for a while to see how easy it is to create random textures of varying density, and then to get a firm grasp of how *packing*, *scatter* and *tgrid* affect each other. This will provide you with a firm foundation with which to make full use of the more complex facilities which the TEXTURE Group provides.

Harmonic fields or sets – handling the pitches
TEXTURE SIMPLE can also deal with harmonic fields or sets. **Fields** transpose the specified pitch set into all (MIDI) octaves. **Sets** use only the specified pitches. These form pitch lattices onto which the various note events are placed. The first thing to understand here is how the pitches specified in the command parameters relate to those specified in the note data file.

But first, try this example of a fixed pitch set (Mode **3**). The note data file *ndfsim2.txt* has two pitches, both starting at time 0: 60 and 67.

```
60
#2
0 1 60 0 0
0 1 67 0 0
```

You will hear, due to the rapid repetitions, a textured interval of a perfect 5th.

SIMPLE Example 2 (Preset/Patch **simplex2**):
outdur (12) packing (0.025) scatter (0) tgrid (0) sndfirst (1) sndlast (1) mingain (36) maxgain (84) mindur (0.2) maxdur (1.5) minpich (60) maxpich (67) – in command line format:

```
texture simple 3 horn simplex2 ndfsim2.txt 12 0.025 0 0 1 1 36 84 0.2
1.5 60 67
```

- **The rule of thumb is that the program first generates the MIDI note values within the range specified by the parameters, and *then* looks at the pitches listed in your note data file and adjusts the former to fit onto the pitch grid specified by the latter.** (Reminder: The 'pitch parameters' referred to are those entered on the command line or with a graphic interface. The note data file is a text file prepared beforehand.)

- Thus the range of pitch values specified with the *minpich* and *maxpich* parameters is very important:
  - If the parameter pitch range is the same as that in the note data file, all notes will lie within the pitch range.
  - If the parameter pitch range is only one pitch, only one pitch will be produced, regardless of whether or not there are different pitches in the note data file.
  - If the parameter pitch range is greater than that of the range of the list of notes in the note data file, in Mode **1** pitches both inside *and outside* the note data file range will be produced, but keeping to the pitch grid, albeit in another octave. In Mode **3**, only the pitches in the note data file will sound, as any other pitches generated by the pitch parameters will be transposed to lie within the pitch set of the note data file.
  - If the parameter pitch range is within one octave and all the pitches in the note data file are the same, only one pitch will be produced.
  - If the parameter pitch range spans more than one octave and all the pitches in the note data file are the same, the one pitch will sound in different octave transpositions in Mode **1** (harmonic fields – all octaves), and only one pitch will sound in Mode **3** (harmonic sets – uses only the specified pitches).

- The above list of possibilities brings out the difference between Mode **1** and Mode **3**, namely that Mode **1** (Harmonic Fields) allows the pitch events listed in the note data file to sound in all octaves (within the parameter pitch range), whereas Mode **3** (Pitch Sets) forces all pitches to lie on the precise pitch grid specified in the note data file. However, **note that the parameter pitch range needs to be wide enough to accommodate the range in the note data file.** If, for example, the paramter pitch range is a perfect 5th and the note data range is a 12th, the note data pitches which lie outside the perfect 5th will have nowhere to go, and will therefore sound inside the perfect 5th.

- **Note that in TEXTURE SIMPLE, the velocity and the duration fields in the note data file are ignored – this information is taken from the command parameters. Zeros are placed in these fields in the examples in order to emphasise this point.**

  Harmonic fields or sets – handling the durations
  We have already seen how the timing of the events is random within constraints, with the density (and number of events) controlled by the *packing* parameter, while *scatter* can somewhat randomise the time onsets of these events.

  In TEXTURE SIMPLE, the duration parameter in the note data file is inactive and may be zero (or any other value – it is ignored.) When this parameter is active, the interaction between the command and note event durations listed in the note data file works in a similar way as that for pitch.

  - Thus, the parameter duration range (*mindur* and *maxdur*) needs to be wide enough to accommodate the shortest and longest duration listed in the note data file.

  - If the parameter duration range is smaller than the range of durations in the note data file, the durations of the note events will be constrained within this smaller range.

  - However, if the parameter duration range is wider than the duration range of the note events in the note data file, you will find that the note events will use this wider range.

  - So with durations, it may usually be best to match the duration ranges of the parameter data and the note data file data.

Examples for harmonic sets and fields

The following example illustrates Mode **3**. The notelist uses the pitches 60-67-72-76 (i.e., C-G-C-E': root-fifth-octave-tenth). The parameter pitch range is set as 60 to 76 to encompass the range in the note data file. Because Mode **3** is for pitch sets, we will hear only the 4 pitches listed in the note data file, spread out in a randomised order over the specified output duration. Here is the note data file for this (*ndfsim3a.txt*):

```
60
#4
0 1 60 0 0
0 1 67 0 0
0 1 72 0 0
0 1 76 0 0
```

Ndfsim3b.txt is the same, except the times (first column) are 0 4 7 and 11. You can run these notedata files with the following command lines:

> **SIMPLE Examples 3a/b/c**(Presets/Patches **simplex3a,b &c**):
> *outdur (12) packing (0.25) scatter (0) tgrid (0) sndfirst (1) sndlast (1) mingain (36) maxgain (84) mindur (0.2) maxdur (1.5) minpich (60) maxpich (76)* – in command line format:

```
texture simple 3 horn simplex3a ndfsim3a.txt 12 0.25 0 0 1 1 36 84
0.2 1.5 60 76
texture simple 2 horn simplex3b ndfsim3b.txt 12 0.25 0 0 1 1 36 84
0.2 1.5 60 76
texture simple 2 horn simplex3c ndfsim3b.txt 12 0.25 0 0 1 1 36 84
0.2 1.5 36 96
```

Let's look at what the various parameter options chosen mean – reminder: the parameters after the notedata file are (EX 3a):
outdur (12) packing (0.25) scatter (0) tgrid (0) sndfirst (1) sndlast (1) mingain (64) maxgain (64) mindur (0.2) maxdur (1.5) minpich (60) maxpich (76)

- The duration of the output soundfile is 12 seconds
- The *packing* (i.e., density of – time between) – the note events is ¼ sec. If you give larger and smaller values for this parameter, you will see that the program will generate fewer or more note events. (Try a *packing* of 0.025.) No timings are given in the notelist: all are set at 0.
- *Scatter* is set at 0. This means that the note events will form regular divisions of the *outdur*. The start time of each event will be increasingly randomised as the value for *scatter* is increased – note the large range of 0 to 10 available for *scatter*. Try adding some scatter.
- *Tgrid* is also set to 0.
- The first and last sounds are both given as '1'. This means that there is only one input sound. If there are two to be used, you would put 1 2. The program reads your list of input soundfiles and assigns numbers in the order in which they occur in command parameter for input soundfiles. If you have more than one *infile*, you need additional entries for the real or arbitrary pitch of each in the note data file (on the first line, separated by spaces).
- The gain parameters set a MIDI 'velocity' range, in this case all notes will have an equal mid-range amplitude.
- The *mindur maxdur* range is set as '0.2 1.5', to match the range in the note data file.
  - When harmonic fields or sets are used, the duration (sustain) parameter in the note data file has no effect and can be set to zero. **This means that in TEXTURE SIMPLE and TEXTURE GROUPED, the duration parameter in the note data file is not used in any of the 5 modes.** The note durations are taken from the parameter duration range, unless you flag it to be ignored (**-w** flag), whereupon it always uses the full duration of the input soundfile for each note events.
  - When decorations and ornaments are used (see the relevant programs), the durations in the note data file must be set and have precedence over the parameter duration range.
  - Also note that notelength is constrained by the duration of your input sound. TEXTURE **cannot** lengthen the input sound – it can only shorten it.

- Also, as you transpose your sound up, it will get shorter (time-domain transposition), which also constrains how long it can be.
- When the duration parameter range takes precedence over the durations specified in the notelist, be sure that the shortest is given as *mindur* and the longest as *maxdur* to ensure that full desired range is employed.

  In the above example, also note that the 1.5 second durations are longer than the ¼ second *packing* parameter, so we hear the notes overlap in the output texture – a rather nice effect.
- The last parameter set used here specifies the pitch range. As noted above, this also affects what is specified in the notelist, so if you want the notelist pitches to be used, be sure that the lowest is given as *minpich* and the highest as *maxpich*.

- New possibilities emerge when the *time* for each note event is specified in the note data file. Try editing *ndatseta.txt* so that the four note events begin at 0.0, 4, 7 and 11 sec. respectively (save as ndatsetb.txt).

- Now run the same again in Mode **2**. You will hear pitches on the harmonic lattice being drawn in from different octaves.

  You should now be able to experiment successfully with TEXTURE SIMPLE.

Two Fields/Sets

The next step is to use **two harmonic fields or sets**. *Ndfsim4a.txt* presents a revised *ndfsim3b.txt* for use with Modes **2** or **4** of the program: 'changing harmonic fields/sets'. There are 8 note events in two 'sets'. The first set starts at time 0, and in the second set, which starts at 6 sec., we change to the minor tenth (75). In the output file, we hear the harmony change from major to minor the second set at the 6 second mark. The 'b' example illustrates how a much tighter packing results in a textured chord.

[ndfsim4a.txt	*ndfsim4b.txt]*
60	60
#8	#8
0 1 60 0 0	0 1 60 0 0
0 1 67 0 0	0 1 67 0 0
0 1 72 0 0	0 1 72 0 0
0 1 76 0 0	0 1 76 0 0
6 1 60 0 0	6 1 60 0 0
6 1 67 0 0	8 1 67 0 0
6 1 72 0 0	10 1 72 0 0
6 1 75 0 0	10 1 75 0 0

SIMPLE Examples 4a/b/c (Presets/Patches **simplex4a,b, &c**):
outdur (12) packing (0.25) scatter (0) tgrid (0) sndfirst (1) sndlast (1) mingain (36) maxgain (84) mindur (0.2) maxdur (1.5) minpich (60) maxpich (76) – in command line format:

```
texture simple 4 horn simplex4a ndfsim4a.txt 12 0.25 0 0 1 1 36 84
0.2 1.5 60 76
texture simple 4 horn simplex4b ndfsim4a.txt 12 0.025 0 0 1 1 36 84
0.2 1.5 60 76
texture simple 4 horn simplex4c ndfsim4b.txt 12 0.025 0 0 1 1 36 84
0.2 1.5 60 76
```

The 'c' example tells us more about how the timing works by staggering the times of the 2nd four note events in *ndfsim4a.txt*: i.e., starting at times 6, 8, 10 and 10 seconds respectively, to create *ndfsim4b.txt*. To do this, we again need to invoke Mode **4** (changing harmonic sets). Now we hear the first four notes, the notes of the 1st set, playing in random order for the first 6 seconds. Then, at time 6 seconds, we hear mainly the 5th note event (with some overlaps from the previous set), at time 8 seconds, the 6th note event, and at time 10 seconds, we hear the 7th and 8th notes in rapid succession, sounding a minor third – except that we will hear multiple iterations of those notes according to the setting of the *packing* parameter. Again the close *packing* at 0.025 sec creates a highly textured surface.

Were we to use Mode **3** with *ndfsim4b.txt*, the different start times would be disregarded and all the note events would play in random order throughout.

The above information tells us that we can create a single melodic line by using Mode 4 and setting a different start time for a series of note events. Some overlapping will occur if the notes are of different durations. *Packing* of onsets closer than the duration of any of the note events will texture the lines, while *packing* of onsets spaced wider than the duration of any of the note events will produce gaps.

So try this 'melodic' example for TEXTURE SIMPLE. The note data file *ndfsim5.txt* contains:

```
60
#10
0.0  1 60 0 0
1.0  1 67 0 0
3.0  1 66 0 0
3.5  1 62 0 0
4.5  1 64 0 0
6.0  1 69 0 0
7.5  1 66 0 0
8.5  1 60 0 0
9.5  1 62 0 0
10.0 1 67 0 0
```

SIMPLE Example 5 (Preset/Patch **simplex5**):
outdur (12) packing (0.4) scatter (0.3) tgrid (0) sndfirst (1) sndlast (1) mingain (36) maxgain (84) mindur (0.2) maxdur (1.5) minpich (60) maxpich (69) – in command line format:

```
texture simple 4 horn simplex5 ndfsim5.txt 12 0.4 0.3 0 1 1 36 84 0.2
1.5 60 69 -w
```

Note that the *packing* is set to be just under the shortest time between two note events, so the melodic notes come through without too much texturing, following the note sequence in *ndfsim5.txt*: C-G-F#-D-E-A-F#-C-D-G. The **-w** flag makes the note duration range redundant, because when this is set the whole duration of the *infile* is used – i.e., the whole length of the horn sound for each note of the *outfile*. In this case, this produces a great deal of overlapping notes – hence the 'resonance'.

Our final topic is **time-varying texture shapes** – i.e., envelope shapes with sloping edges. **Example 6** creates two of these with the files *gprlo.brk* and *gprhi.brk*. The first begins at C-60 and expands up to 70 and down to 49 at 5 seconds, returning to C-60 at 10 sec. Then the second shape starts at 10.01 seconds at 78 high and 58 low and compressing towards the centre G-67 by 18.0 sec. The last portion repeats the G. I've added time-varying packing to create more density where the texture shapes are widest. These are the files:

```
[ndfsim6.txt     simplpak.brk     gprlo.brk        gprhi.brk]
60               0      0.25      0      60        0      60
#22              5      0.1       5      49        5      70
0.0  1 49 0 0    10     0.3       10     60        10     60
0.0  1 52 0 0    10.01  0.05      10.01  58        10.01  78
0.0  1 54 0 0    18     0.15      18     67        18     67
0.0  1 55 0 0    20     0.25      20     67        20     67
0.0  1 58 0 0
0.0  1 60 0 0
0.0  1 61 0 0
0.0  1 64 0 0
0.0  1 66 0 0
0.0  1 67 0 0
0.0  1 70 0 0
10.0 1 58 0 0
10.0 1 60 0 0
10.0 1 61 0 0
10.0 1 64 0 0
10.0 1 66 0 0
10.0 1 67 0 0
10.0 1 70 0 0
10.0 1 72 0 0
10.0 1 73 0 0
10.0 1 76 0 0
10.0 1 78 0 0
```

And the parameters are set as follows:

SIMPLE Example 6 (Preset/Patch **simplex6**):
outdur (12) packing (simplpak.txt) scatter (0) tgrid (0) sndfirst (1) sndlast (1) mingain (36) maxgain (84) mindur (0.2) maxdur (1.5) minpich (gprlo.brk) maxpich (gprhi.brk) – in command line format:

```
texture simple 3 horn simplex6 ndfsim6.txt 21 simplpak.brk 0 0 1 1 36
84 0.2 1.5 gprlo.brk gprhi.brk
```

Musical Applications

We can already see that this is an extremely powerful set of programs, which can undoubtedly be used in endlessly imaginative ways. For the moment, let us simply outline what we have learned so far. With the TEXTURE program group, we can create:

- a melodic line, with gaps and/or overlaps.

  METHOD: use Mode **4**, specify specific start times for each note event in the *ndata* file, and set the *packing* to be about the same as the shortest time between notes. Note durations in the note data file can be zero (they are disregarded with harmonic fields or sets); note overlaps result from making the successive start times shorter than at least some of the note durations set with the parameter duration range. With the **-w** flag set, there may be even more overlapping if the sound's duration is longer than the *packing* setting, with possibilities for resonances and harmonies emerging from the melody.

- a melodic line, with a grainy, textured surface.

  METHOD: use Mode **4** and specify specific start times in the *ndata* file and set *packing* to be very short, certainly much shorter than the time between note events. A very close *packing* (e.g., 0.01) may cause amplitude overload. You can apply the gain reduction recommended by the program, or perhaps reduce the *maxdur* value to lessen the amount of note overlap.

- a texture of randomly selected notes drawn from a defined list of notes – i.e., these can, for example, form a sonorous chord when the *packing* is tight.

  METHOD: use Mode **3**, set all start times in the *ndata* file to 0 and specify pitches as desired. A *mindur* – *maxdur* range will produce note overlaps (unless *maxdur* is less than the time gap between events).

- a texture which moves between two harmonic fields (Mode **2** all octaves) or sets (Mode **4** only the pitches specified).

  METHOD: in the *ndata* file, start all note events of the first set at one time, and all note events of the second set at another, later, time.

- a texture with envelope shapes, with pitches drawn from a defined harmonic set; time-varying packing can help make the shapes audible by changing the note density.

  METHOD: define harmonic sets, which may differ at different times. Then create time-varying group pitch range files to define the envelope shapes.

End of TEXTURE SIMPLE

TEXTURE GROUPED – A texture of separate event groups is shaped by random selections from parameter ranges, with one or more input sounds; a Harmonic Field/Set may be used

Usage

texture grouped mode *infile [infile2...] outfile notedata outdur packing scatter tgrid*
sndfirst sndlast mingain maxgain mindur maxdur minpich maxpich phgrid
grpspace gpsprange amprise contour
grpsizelo grpsizehi gppaklo gppakhi gpranglo gpranghi
[**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*] [**-w**] [**-d**] [**-i**]

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

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line. In Mode **5** this is all that is needed. 60=original pitch.
2. In Modes **1 - 4** there is also a NOTELIST to define a Harmonic Field/Set, specified thus:
   - **#***N* (where *N* is the number of pitches in the notelist).
   - This is followed by *N* lines in the format:
     time (secs) *infile_no pitch* (MIDI) *amp* (MIDI) *dur* (secs)
   - *Amp* and *dur* are inactive.
   - Any times within the field must increase (or remain the same, during chords)

Form: **MPV/–/–/–**		Form: **MPV/HF-S/–/–**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
		#*N*	No. of lines to follow
		time instr pitch amp dur	List of Harmonic Field/Set

outdur – minimum duration of the *outfile*
packing – (average) time in seconds between group onsets
scatter – randomisation of group onsets (Range: 0 to 10)
tgrid – minimum step *in milliseconds* for quantised timegrid (for group start times) (Default: 0)
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
mindur, maxdur – minimum and maximum duration of events in texture
minpich, maxpich – minimum and maximum pitch (MIDI note value)
phgrid – a timegrid *in milliseconds* applying WITHIN the groups (Range: 0.0 to 1000.0)
gpspace – spatialisation of event-groups (Range: 0 to 5; Default: 1)

0 no change
1 scattered (Default)
2 decorations will move **towards** where the event is
3 decorations move **away from** where the event is
4 decorations follow the texture motion from Left to Right
5 decorations follow the texture motion from Right to Left

gpsprange – spatial range of event-groups (Range: 0 to 1; Default: 1)
amprise – amplitude change within groups (Range: 0 to 127; Default: 0)
contour – amplitude contour of groups (Range: 0 to 6; Default: 0)

0 mixture of the other types (Default)
1 crescendo
2 flat (no change)
3 decrescendo
4 crescendo or flat
5 crescendo or decrescendo
6 decrescendo or flat

gpsizelo, gpsizehi – smallest & largest number of events in any given group
gppaklo, gppakhi – shortest & longest time between event-onsets within a group (Range *in milliseconds*: 0.023 to 60000.0 – i.e., 1 minute)
gpranglo, gpranghi – minimum & maximum pitch range for the note events in any given group OR, for harmonic fields, the number of notes in the range of the harmonic field
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values
-d – fixed timestep between groupnotes
-i – each group not confined to a fixed instrument (Default: fixed)

Understanding the TEXTURE GROUPED Process

Please refer to TEXTURE SIMPLE if you are not already familiar with the Texture functions, and also make sure you have gone through the **background information** before continuing. The following text will focus on what is specific to TEXTURE GROUPED.

The idea behind TEXTURE GROUPED is easily grasped by imagining several quick bursts of note events separated by silence. A 'group' is a 'burst of notes', and the program handles creating silences inbetween bursts. The program is, however, more flexible than that because:

- the note groups can in fact overlap
- the timing of the note onsets within a group doesn't have to be rapid
- the pitch location where each group begins can vary

You can easily understand, therefore, how the program makes it possible to create textures which are more varied than those of TEXTURE SIMPLE, and, if desired, punctuated by note groups which form **gestural shapes**.

When using a command line rather than a graphic interface, to help with remembering all the parameters, I find it helpful to use a batch file which starts with 'rem' lines which list the parameters in groups:

```
rem minoutdur pack scat tgrid
rem sndf sndl ming maxg mind maxd minp maxp
rem phgrid gpspace gpsprange amprise contour
rem gpsizelo gpsizehi gppaklo gppakhi gpranglo gpranghi
rem -aatten -ppos -ssprd -rseed -w -d -i
```

Unfortunately, Carriage Returns are not acceptable in the command line itself, so this memory aid is still not as helpful as one might wish. Also, when exploring a program of this complexity, it can be helpful to make use of the CDP environment variable which enables you to overwrite a soundfile of a specific name: **set CDP_OVERWRITE_FILE= aname**. Then you can do several of versions without having to rename them, and when you find one that you want to keep, just save it to a new name.

As before, let's start with an example we can hear. Example 1 produces separated groups of rapid-fire note events; the groups start two seconds apart. We are using Mode **3** ('on a given harmonic set'), with the note data file *ndfgrp1.txt*, which is:

```
60
#4
0 1 62 0 0
0 1 64 0 0
0 1 66 0 0
0 1 67 0 0
```

GROUPED Example 1 (Preset/Patch **grouped1/groupex1**):
... minoutdur (10) packing (2) scatter (0) tgrid (0)
sndf (1) sndl (1) ming (30) maxg (64) mind (0.25) maxd (1.25) minp (36) maxp (84)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (10) gpsizehi (20) gppaklo (25) gppakhi (50) gpranglo (4) gpranghi (4)

```
texture grouped 3 horn groupex1 ndfgrp1.txt 10 2 0 0 1 1 30 64 0.25 1.2 36 84 0
1 1 0 0 10 20 25 50 4 4
```

All the examples for TEXTURE GROUPED can be made by running *groupexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *groupdel.bat*.

Timing from group to group
TEXTURE GROUPED builds on TEXTURE SIMPLE by dealing with **groups** of notes where formerly there would have been only one. *Packing* determines the amount of time **from the onset of one group to the onset of the next group**, *scatter* randomises this timing, and *tgrid* quantises it, i.e, introduces regularity. In the above example, there are 2 seconds between the onsets of each group. Note that in the note data file, the start times, velocities and durations are set to zero. The start times become relevant when using **changing** harmonic fields or sets.

Shaping each group – number of events & pitch range
The number of events allowed in any given group is set with the *gpsizelo* & *gpsizehi* parameters, 10 – 20 in the above example.

Note that these can both be set to 1 so that there will only be one event in each group. This can be a useful way to observe the time gaps between groups, and the effect that *scatter* and *tgrid* have on this timing.

Similarly, the size of the pitch range allowed in any given group is set with *gpranglo* & *gpranghi*. In the above example, each group will remain within a range of only 4 notes (matching the note data file).

The pitch start point of a group is selected from within *minpich* & *maxpich*. Thus, the overall pitch range, set by *minpich* & *maxpich* may, for example, be an 8^{ve}; but the pitch range in any given group, set by *gpranglo* & *gpranghi*, could be a minor 3^{rd}.

Shaping each group – durations
Since we are dealing with groups, however, the next salient question is how do we control **the timing of events within the groups**? Let's look at the following parameters:

- *mindur* & *maxdur* set a note event duration range; each note event in the group (in the texture) will be a random length chosen at random within this range. What happens here is indicated by a range for example of 0.25 to 1.2: we hear overlapping among the note events simply because some are considerably longer than others. As with TEXTURE SIMPLE, none of the Modes actually use the duration parameter in the note data file. Therefore, these can all be set to zero. The note data durations are used for ornaments and decorations.

- *phgrid* handles a timegrid in milliseconds applied *within each group* – note that it can be 0. You will probably need to think carefully about the relationship of the groups onset times (*packing*), the duration of the note events (*mindur* & *maxdur*), and the number of events in the group (*grpsizelo* & *grpsizehi*). The *packing* may need to be longer if the note durations are relatively long and you do want to retain a gestural separation between the groups. Note that *phgrid* (milliseconds) is a quantisation factor. The packing *within the group* (*gppaklo* & *gppakhi* is what mainly controls the *note event* onset timing – milliseconds! **NB:** The *gppakhi* parameter must be at least equal to *phgrid* for the latter to have the desired effect; i.e., the *phgrid* parameter must lie within the packing range within the group.

- *gppaklo* & *gppakhi* handles the onsets of events within groups. Note that very small values for the packing can be used, and consider what might be appropriate in each instance.

Musical Applications

Here are some suggestions for different types of texture that can be created with TEXTURE GROUPED.

- **GROUPED Example 1** (Preset/Patch **grouped1/groupex1**): – We can start with the example used at the beginning of the TEXTURE GROUPED documentation: rapid-fire note events within a group separated by silence. Thus each group becomes a musical gesture. Reminder: the initial command set was:

  rem minoutdur (10) packing (2) scatter (0) tgrid (0)
  rem sndf (1) sndl (1) ming (30) maxg (64) mind (0.25) maxd (1.25) minp (36) maxp (84)
  rem phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
  rem gpsizelo (10) gpsizehi (20) gppaklo (25) gppakhi (50) gpranglo (4) gpranghi (4)
  `texture grouped 3 horn groupex1 ndfgrp1.txt 10 2 0 0 1 1 30 64 0.25 1.2 36 84 0 1 1 0 0 10 20 25 50 4 4`

- **GROUPED Example 2** (Preset/Patch **grouped2/groupex2**): – The time between group onsets gradually decreases from 2 to ½ second, as set by *grppack.brk*: 0 2, 10 0.5.
  `texture grouped 3 horn groupex2 ndfgrp1.txt 10 grppack.brk 0 0 1 1 30 64 0.25 1.2 36 84 0 1 1 0 0 10 20 25 50 4 4`
- **GROUPED Example 3** (Preset/Patch **grouped3/groupex3**): – A regularly timed sequence of random notes taken from within the group range can be achieved by adapting the initial example by setting, for example, *phgrid* to 200, and raising *gppakhi* also to 200.
  `texture grouped 3 horn groupex3 ndfgrp1.txt 10 2 0 0 1 1 30 64 0.25 1.2 36 84 200 1 1 0 0 10 20 25 200 4 4`
- **GROUPED Example 4** (Preset/Patch **grouped4/groupex4**): – Experiment with more overlap by changing *mindur* to 0.75 (all other parameters kept the same).
  `texture grouped 3 horn groupsex4 ndfgrp1.txt 10 2 0 0 1 1 30 64 0.75 1.2 36 84 200 1 1 0 0 10 20 25 200 4 4`
- **GROUPED Example 5** (Preset/Patch **grouped5/groupex5**): – Alternatively, we could go for a staccato effect by using a very small note duration range: *mindur* 0.1 & *maxdur* 0.2 and making the inter-group packing range wider than the durations: *gppaklo* 25 & *gppakhi* 300.
  `texture grouped 3 horn groupsex5 ndfgrp1.txt 10 2 0 0 1 1 30 64 0.1 0.2 36 84 200 1 1 0 0 10 20 25 300 4 4`
- **GROUPED Example 6** (Preset/Patch **grouped6/groupex6**): – Now overlap the groups themselves by changing *packing* to 0.5. Reset *phgrid* to 0 and reduce *gppakhi* to 50 to restore the initial rapid-fire group note events.
  `texture grouped 3 horn groupex6 ndfgrp1.txt 10 0.5 0 0 1 1 30 64 0.2 0.3 36 84 0 1 1 0 0 10 20 25 50 4 4`

End of TEXTURE GROUPED

TEXTURE DECORATED|PREDECOR|POSTDECOR – A texture of 'decoration' events shaped by internal grouping parameters, selected at random from a pitch range or from a Harmonic Field/Set and attached to an underlying 'line'; one or more input sounds

Usage

texture decorated|predecor|postdecor mode *infile [infile2 ..] outfile notedata outdur skiptime*
sndfirst sndlast mingain maxgain mindur maxdur
phgrid grpspace gpsprange amprise contour
grpsizelo grpsizehi gppaklo gppakhi gpranglo gpranghi centring
[**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*]
[**-w**] [**-d**] [**-i**] [**-h**] [**-e**] [**-k**]

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

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line (60=original pitch)
2. followed by a 'line' substructure NOTELIST, specified thus:
   - **#***N* (where *N* is the number of pitches in the notelist).
   - This is followed by *N* lines in the format:
     time (secs) *infile_no pitch* (MIDI) *amp* (MIDI) *dur* (secs); *amp* and *dur* are inactive fields in TEXTURE DECORATED.
   - different times must be given, and they must must increase; all 0's for chords are not allowed.
   - **this is all that is needed for Mode 5**.
3. in Modes **1** to **4**, you need an additional set of lines for the decoration definition; these are also introduced by the number of lines (**#***N*, which acts as a separator). The decoration is 'tied into' the above 'line' substructure set by providing the same start time (may be several decoration notes at the same time) and by duplicating the MIDI note value for one of the decoration notes. (See illustrations below.)

Form: **MPV/NS/–/–**		Form: **MPV/NS/HF-S/–**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
#N	No. of lines to follow	***#N***	No. of lines to follow
time instr pitch amp dur	List of 'line' (Nodal Substructure)	*time instr pitch amp dur*	List of 'line' (Nodal Substructure)
		#N	No. of lines to follow
		time instr pitch amp dur	List of Harmonic Field/Set

outdur – minimum duration of the *outfile*
skiptime – time between repeats of motif-to-decorate in *notedata*, i.e., between runs of the 'line' substructure notelist
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
mindur, maxdur – minimum and maximum duration of events in texture
phgrid – a timegrid in milliseconds applying WITHIN the decorations (Range: 0.0 to 1000.0)
gpspace – spatialisation of decoration-groups (Range: 0 to 5; Default: 1)

0 no change
1 scattered (Default)
2 decorations will move **towards** where the event is
3 decorations move **away from** where the event is
4 decorations follow the texture motion from Left to Right
5 decorations follow the texture motion from Right to Left

gpsprange – spatial range of decoration-groups (Range: 0 to 1; Default: 1)
amprise – amplitude change within decorations (Range: 0 to 127; Default: 0)
contour – amplitude contour of decorations (Range: 0 to 6; Default: 0)

0 mixture of the other types (Default)
1 crescendo
2 flat (no change)
3 decrescendo
4 crescendo or flat
5 crescendo or decrescendo
6 decrescendo or flat

gpsizelo, gpsizehi – smallest & largest number of events in the decorations
gppaklo, gppakhi – shortest & longest time between event-onsets in the decorations (Range in milliseconds: 0.023 to 60000.0 – i.e., 1 minute)
gpranglo, gpranghi – minimum & maximum pitch range of the decorations, given not in MIDI note values but as the number of *adjacent tones* to be used: from the full chromatic field in Mode **5**, OR from the members of the harmonic field/set defined in the other Modes)
centring – how the decoration pitches centre on line pitches (Range: 0 to 7; Default: 0)

0 centred (default)
1 above
2 below
3 centred and above

4 centered and below
5 above and below
6 centred and above and below

-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values
-d – fixed timestep between groupnotes
-i – each group not confined to a fixed instrument (Default: fixed)
-h – decorate **top**note of chord (Default: decorate first note listed)
-e – decorate all the notes of chords
-k – discard original line, after decoration

Understanding the TEXTURE DECORATED Process

The key to TEXTURE DECORATED is that the (ascending) start times defining the the 'line' of nodes to be decorated are important. The decorations are connected to the line centred (DECORATED), before and ending on (PREDECOR), and starting precisely on (POSTDECOR), these time points. Breakpoint times also have to tie in with these times or nothing will happen. Musically, this means that the decorations on a series of nodes can be varyingly constructed. Remember that a 'decoration' consists of notes chosen randomly from the decoration field, which can be more or less harmonic. This randomness is what differentiates DECOR and ORNATE. In ORNATE, there can be figures with internal rhythm and velocity fully defined and also attached to nodes. In MOTIFS, fully defined figures are used, but not a defined nodal substructure.

The TEXTURE DECORATED node data file, then, uses two groups of data:

1. a 'line' of note events with start times to define a substructure on which to build the decorations, and
2. the decorations themselves.
    - In Mode **5** these are defined only by the parameters: group size, group packing and group range: i.e., not in the note data file. Thus, in Mode **5** there is no second (or third) section in the note data file.
    - In Modes **1–4** the decorations are defined as a second set of data in the note data file, beginning with the number of lines of data to come, e.g, #5. We will see examples of this below.

In all cases, the decorations are randomly selected from the pitch range or note lists provided – a 'decoration' is here defined as a random assembly of pitches. When harmonic sets are used, only the notes listed are used (in random order). When harmonic fields are used, all octaves of the notes listed are used (in random order). Thus the decorations can be given a harmonic quality. Longer timings *between the notes of the decorations* may cause new decorations to come in (on the next specified substructure node) before the previous one has finished. Such overlapping can be musically useful.

The decorations are by default located around (centred on) the 'line' substructure nodes. The *centring* parameter setting enables the decorations to be located above, below etc. the 'line' substructure nodes.

Similarly, with TEXTURE DECORATED the decorations are *centered* on the start times listed for the main substructure nodes, i.e., they will tend to come in a little before the node start time and finish a little after this time. When TEXTURE PREDECOR is invoked, the decorations happen *before* the main substructure nodes, ending at the node start time. The decorations start precisely on the node start time and therefore come *after* the node with TEXTURE POSTDECOR. Larger group packing times will spread out the decoration, however, and may cause it to overlap (still be unfolding) when the next node begins. By having a lower group size or shorter durations for the group packing, it is possible to have the decoration end before the next node begins, in which case there will be a silence until the time for that node is reached.

In the introductory notes about the note data file, the *order* of placement of the data was mentioned. First is the **line**, then the data for **harmonic fields/sets** – which is also the data used for **decorations**. Thus, this second set of data can be any group of notes: it doesn't have to be 'harmonic'. Thirdly, there is the data for **ornaments/motifs**, which we shall cover in TEXTURE ORNATE etc.

Basically, what happens in the note data file for TEXTURE DECORATED is this:

1. First, the notes in any timed, ornamented or decorated **line** in the texture are defined. 'Line' here is taken to mean just that: the basic linear substructure for the textural additions. This list of notes begins by defining the number of notes in this 'line', e.g., #5.

   This list is required in ALL modes of TEXTURE DECORATED, *with start times which must increase from note to note*. In Mode **5** it is the *only* field that is needed.

2. Secondly, the notes in any harmonic fields or sets are defined. This is the list which forms the basis of the decoration, which is in fact a 'random assembly' of these notes. This list of notes begins by defining the number of notes in this field or set.e.g., #5 – **thus the number-of-notes definitions act as separators in the note data file**. The actual note event data, i.e., the 'field' of notes for the decorations, follows. **All start times in this 'decoration field' are 0.**

   What happens is this: *gpranglo/hi* define **how many notes of this field** to use for any given decoration. It then draws that number of notes from this field (and no other notes), with each node note of the substructure list as the time and pitch point of reference, and plays them in random order ('random assembly'). Note that the overall pitch range as well as the top of the field itself must be able to must accommodate *gpranglo/hi* **number of field notes** around, above, or below the lowest/ highest node in the substructure, according to the *centring* parameter.

3. Thirdly, the notes or any ornaments or motifs are specified. Again, this list of notes begins by defining the number of notes in this ornament or motif, e.g., #3.

   This third list is not required by / used in TEXTURE DECORATED.

While similar to TEXTURE GROUPED, we notice that TEXTURE DECORATED lacks the trio of parameters which defined the timing between groups (*packing scatter tgrid*). Rather, we find the new parameters *skiptime* and *centring*. This is because it is building its texture on a linear substructure, which is what is first defined in the note data file.

So let us focus our attention on these parameters.

- *skiptime* – The first list of 'line' substructure notes in the note data file form the core outline of the decoration. Remember that this list includes start times for each note event, such that the whole list forms a single musical unit. If the output duration is longer than this unit, the whole sequence will repeat. *Skiptime* is the time between these repeats.

- *centring* – A decoration will usually have several notes, though it may also consist of repeated notes only. *Centring* deals with how these several notes lie against the substructure node of the 'line': centred on it, lying below or above it, etc.

We are now ready to look at some examples and will do so directly in the 'Musical Applications' section below.

Musical Applications

All the examples for TEXTURE DECORATED can be made by running *decorexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *decordel.bat*.

See *decorexs.bat* for a listing of the examples in the TEXTURE DECORATED section. You can run this batch file to create the examples. You will find it and the various note data and breakpoint files in your Texture Pack.

Example 1a
Our first example for TEXTURE DECORATED uses Mode **5**, where only the 'line' substructure note list is needed. We shall create 4 nodes and place a decoration consisting of 5 repeating notes on each of these nodes. In Mode **5** we don't have to specify the decoration as the information required is supplied by the parameters. Here is a note data file which specifies these four 'line' substructure nodes, *ndfdec1.txt*:

```
60
#4
0.0 1 60 0 0
2.0 1 67 0 0
4.0 1 63 0 0
6.0 1 62 0 0
```

This will create a line of substructure nodes C-G-Eb-D, where C is Middle C, at 0, 2, 4 and 6 seconds respectively. With an *outdur* of 20 seconds, this 4-node musical unit will repeat after 2 *skiptime* seconds. *Skiptime* begins at the start time of the last substructure node. As the time between nodes is a regular two seconds, the node sequence will repeat without breaking that regularity.

The decoration of 5 micro-trills (which may sound like just notes which repeat) is set by making both *gpsizelo* and *gpsizehi* equal to 5, and we'll have a reasonably quick, but not too fast, and regular, note speed *within* the decoration by making both *gppaklo* and *gppakhi* equal to 60 ms. The trills are very small (equal to or less than a semitone) because *gpranglo/hi* are both set to 1.

Here is the command line which summarises this data:

DECORATED Example 1a (Preset/Patch **decor1a/decorex1a**):
... minoutdur (20) skiptime (2.0)
sndf (1) sndl (1) ming (36) maxg (64) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (5) gpsizehi (5) gppaklo (60) gppakhi (60) gpranglo (1) gpranghi (1) centring (0)

```
texture decorated 5 horn decorex1a ndfdec1.txt 20 2.0 1 1 36 64 1.0
1.0 0 1 1 0 0 5 5 60 60 1 1 0
```

The role of the group pitch range *gpranglo/hi* is instructive:

- In the above example, we hear the substructure pitch shape very clearly because these are the only pitches used (*gpranglo/hi* are 1); the nodes are textured by rapid micro-trils due to the very short group packing. The whole shape repeats after 4 (*skiptime*) seconds.

- If we make *gpranglo/hi* both 2, we hear semitone trills starting on the note below and ending on and sustaining the final (substructure) node after the 5 notes of the decoration until the full 1 second has finished (*centring* is 0). If *centring* is then changed to 1 (i.e., 'above'), the trill begins on the node, trills with and ends on the note above (which in this case is harmonically odd). If we make *centring* 2 (i.e., 'below'), in this case the trilling effect starts below the node.

- If we make *gpranglo/hi* both 3, we get whole tone trills (*centring* back to 0). This tells us that the interval count seems to start at 1 for the same note (which will be repeated at group packing rates), 2 for a semitone, 3 for a whole tone, 4 for a minor third, etc., i.e., one more than the usual way we tend to count intervals. Actually, the numbers do refer to semitones in the normal way. It's just that, if you specify a maximum range of a semitone, for example, the notes generated (in the Mode **5** Neutral case) will all fall *within* a semitone range. The interval of a semitone (1) that you specify is the **maximum distance a note might land on**. In general most notes (in this case) will be less than a semitone away from the base pitch (0). What you hear, then, is not repeated notes, but 'micro-trills' on intervals within a semitone – and they are not exactly trills either, because no note is repeated! When larger intervals are specified, many of the notes will fall within this range, often far short of the full interval specified.

- If we make *gpranglo/hi* both 4, we get a wash of all the notes within a minor third, centred on the nodes. This wash illustrates the 'random assembly' aspect of TEXTURE DECORATED.

- If we make *gpranglo* = 1 and *gpranghi* = 4, we get a mixture of the above possibilities on the various nodes: i.e., the program will make a random selection of anywhere from 1 to 4 notes for the decoration range.

- if we make *gpranglo* = 1 and *gpranghi* = the breakpoint file *dm5gprhi.brk*, we hear the number of notes in the decorations alternating between 1 and several on each of the nodes. We present this as Example 2. Note that the breakpoint times in *dm5gprhi.brk* match those in the note data file (every 2 seconds), and *skiptime* is still 0.01 so that there is no appreciable delay between repeats of the substructure musical unit. (The timings in the breakpoint files must increase, but do not have to match the node times. The parameter values used at nodes will be those in force at the time of the nodes. If nodes lie between times specified in the breakpoint file, you will pick up the intermediate value at that time.)

- The first substructure note does not receive a decoration with PREDECOR (this is because the first event in the line is assumed to be at time zero – so no decoration can happen before this). In mode DECORATED, a random mixture of pre- and post-decorations is produced, so the first note is sometimes decorated and sometimes not.

Example 1b
Still in Mode **5** and using the same note data file as in Example 1a (*ndfdec1.txt*), Example 1b also employs the breakpoint file *dm5gprhi.brk* for the parameter *gpranghi*:

```
0       1
2       8
4       1
6       6
8       1
10      10
12      1
14      7
16      1
18      9
20      1
```

POST-DECORATED Example 1b (Preset/Patch **decor1b/ decorex1b**):
... minoutdur (20) skiptime (2.0)
sndf (1) sndl (1) ming (36) maxg (64) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (5) gpsizehi (5) gppaklo (60) gppakhi (60) gpranglo (1) gpranghi (dm5gprhi.brk) centring (0)

```
texture postdecor 5 horn decorex1b ndfdec1.txt 20 2.0 1 1 36 64 1.0
1.0 0 1 1 0 0 5 5 60 60 1 dm5gprhi.brk 0
```

POSTDECOR is used so that the decorations will begin precisely *on* the node start time: because in POSTDECOR the decorations come *after* the node. We hear little burbles on every other node because in *dm5gprhi.brk* every other node has a pitch range greater than 1.

Example 2a
If we now move on to the **harmonic set/field modes**, we will see how the 'random assembly' feature is still present, but tied into specifically named pitches. We start with Mode **3** and here we need to define the pitches from which the decoration will be formed, so a second note list is needed in the note data file. In the following example, the substructure nodes form a rising G-minor shape: G4-Bb4-D5-G5, where C5 is Middle C. Each note of this shape is then decorated with a whole tone trill. *Gpranglo/hi* is set to 2, **meaning in this case how many notes from the decoration field to use at any one time** – and only these notes are used, not any of the inbetween ones as in Mode **5**. **Note that all the times of the decoration field are set to 0.** Here is the note data file for this example, *ndfdec2a.txt*:

```
60
#4
0.0 1 60 0 0
2.0 1 67 0 0
4.0 1 63 0 0
6.0 1 60 0 0
#3
0.0 1 60 0 0
0.0 1 64 0 0
0.0 1 67 0 0
```

POST-DECORATED Example 2a (Preset/Patch **decor2a/ decorex2a**):
... minoutdur (12) skiptime (2.0)
sndf (1) sndl (1) ming (36) maxg (64) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (5) gpsizehi (9) gppaklo (60) gppakhi (60) gpranglo (3) gpranghi (3)
centring (1)

```
texture postdecor 3 horn decorex2a ndfdec2a.txt 20 2.0 1 1 36 64 1.0
1.0 0 1 1 0 0 5 9 60 60 3 3 1
```

Note that *gpranglo/hi* is set to 3, to use all three notes in the decoration field. The same three decoration notes decorate each of the substructure nodes. In this case, all the notes match except for the Eb/E-natural. Thus only the third node sounds different. Using the '1' option for *centring* did NOT result in the notes of the decoration being transposed to build on (above) each of the substructure nodes. All the nodes are decorated with the *same chord* because those are the only (3) notes given in the decoration field in Mode **3** in which only the notes in the defined harmonic set are used. If we provide more notes, in fact the pitches needed to place a triad on each node, we find that this is indeed what happens. Thus we can hear that TEXTURE DECORATED does use each node as a tonal centre for each decoration on it.

Example 2b
Example 2b illustrates constructing a decoration field that will enable building different **triads** on the substructure nodes. The note data file *ndfdec2b.txt* is as follows. Note that all start times are still 0 – the decoration start times come from the node itself.

```
60
#8
 0.0 1 60 0 0
 2.0 1 63 0 0
 4.0 1 67 0 0
 6.0 1 70 0 0
 8.0 1 74 0 0
10.0 1 70 0 0
12.0 1 67 0 0
14.0 1 63 0 0
#7
0.0 1 60 0 0
0.0 1 63 0 0
0.0 1 67 0 0
0.0 1 70 0 0
0.0 1 74 0 0
0.0 1 77 0 0
0.0 1 81 0 0
```

POST-DECORATED Example 2b (Preset/Patch **decor2b/ decorex2b**):
... minoutdur (28) skiptime (2.0)
sndf (1) sndl (1) ming (36) maxg (64) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (5) gpsizehi (9) gppaklo (60) gppakhi (60) gpranglo (3) gpranghi (3)
centring (1)

```
texture postdecor 3 horn decorex2b ndfdec2b.txt 20 2.0 1 1 36 64 1.0
1.0 0 1 1 0 0 5 9 60 60 3 3 1
```

Now we have different decorations (all of which are triads) coming in on each time-specified node. We have also used *centring* = 1 so that the node is the root (bottom note) of each decoration.

The key to achieving decorations restricted to triadic formations on the various nodes lies in the design of the harmonic field/set. The set used here moves up steadily in 3rds so that *all adjacent members* of the set remain a 3rd apart (major or minor): C - Eb - G - Bb - D' - F' - A'. The highest node is D' (74) so that the F' and A' are available to form the triad. Thus, when the triads are built on each of the nodes, we get the following triads: Cm, Ebmaj, Gm, Bbmaj, and Dm.

Note that if we tried to have major and minor versions of the same triad, this would introduce adjacent semitones into the field, e.g., C - Eb - E-natural - G. The program can only select *grprange* adjacent events, so in this case the first group would use C - Eb - E-natural and not form a triad.

Mode **4** used with this note data file produces the same results because no times are specified in the harmonic field/set. It is quite possible to have changing fields, in which case the *grprange* adjacent notes would change their harmonic character each time a new field came in. Modes **1** and **2** introduce note-events at different octaves, as permitted by the overall pitch range.

Example 2c
The next example illustrates the same process, this time with a note data file designed to produce dyads (pairs of notes around a single interval). Here the pitches in the decoration field are selected so that they will harmonise with the nodal tones. Then they are all set to begin at time zero, and **we rely on *gpranglo/hi* (set to 2) to select pairs of notes from this field.**

The idea of the example is to build dyads on each of the substructure nodes. We use *centring* = 1 to get the decoration to start on and lie above the substructure node. This time all the nodes of the substructure are decorated. *Ndfdec2c.txt* follows, and we are running it first in Mode **3**.

```
60
#4
0.0 1 55 0 0
2.0 1 58 0 0
4.0 1 62 0 0
6.0 1 65 0 0
#8
0.0 1 55 0 0
0.0 1 58 0 0
0.0 1 62 0 0
0.0 1 65 0 0
0.0 1 67 0 0
0.0 1 70 0 0
0.0 1 74 0 0
0.0 1 77 0 0
```

POST-DECORATED Example 2c (Preset/Patch **decor2c/ decorex2c**):
... minoutdur (20) skiptime (2.0)
sndf (1) sndl (1) ming (36) maxg (64) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (5) gpsizehi (5) gppaklo (60) gppakhi (60) gpranglo (2) gpranghi (2)
centring (1)

```
texture postdecor 3 horn decorex2c ndfdec2c.txt 20 2.0 1 1 36 64 1.0
1.0 0 1 1 0 0 5 5 60 60 2 2 1
```

Now this is important: We set *gpranglo/hi* to 2 to specify that we should use pairs of notes from the decoration field. What we hear, then, are the notes of the dyads selected from the decoration field and only these notes played in random order above each of the respective nodes. The decoration field has to be carefully constructed so that the desired harmonic effect is achieved.

Mode **4** used with the same data produces the same results. Modes **1** and **2** introduce octave notes, as permitted by the overall pitch range.

Building on these foundations, we can achieve more varied and 'plastic' results by employing ranges and time-varying breakpoint files for the various parameters. But recall what was observed above, that the effect of the times in breakpoint files seems constrained by the start times of the nodes: pending further experimentation, it appears that the parameter value in force when a node begins is what is used: the breakpoint times do not override the node times and therefore cannot bring the parameter value in at a breakpoint time which differs from a node time.

Example 3
Example 3 illustrates a more flexible use of TEXTURE DECORATED. The note data file *ndfdec3.txt* defines five nodes each 5 seconds apart. These are then decorated from a field of 15 notes, with a time-varying set of values for *gprangehi*.

```
[ndfdec3.txt]
60
#5
0  1 60 0 0
5  1 64 0 0
10 1 67 0 0
15 1 70 0 0
20 1 60 0 0
#15
0 1 49 0 0
0 1 52 0 0
0 1 54 0 0
0 1 55 0 0
0 1 58 0 0
0 1 60 0 0
0 1 61 0 0
0 1 64 0 0
0 1 66 0 0
0 1 67 0 0
0 1 70 0 0
0 1 72 0 0
0 1 73 0 0
0 1 76 0 0
0 1 78 0 0
```

The breakpoint file for *gprangehi* is *dx3gprhi.brk*:

```
0  8
5  10
10 15
15 12
20 11
```

Note that the times match those for the 'line' nodes in the note data file. When we hear the resultant sound, we can also observe that each decoration is centred on the node pitch – i.e., they move up and down in pitch according to the underlying node. The group pitch range low is set at a constant 2, so the range of decoration field notes used in each burst will vary between 2 and 8, 2 and 10, 2 and 15, etc.

The *gpsize* is set to give a relatively low number of events: from 10 to 25. The *gppak* range is fast, but varied: between 40 and 125 ms. The overall effect is little bursts of note events, with gaps inbetween. These gaps are those between the end of the decoration and the start of the next node – not *skiptime*, which occurs at the end of the whole sequence. We are using POSTDECOR so that the decorations begin with the node.

POST-DECORATED Example 3 (Preset/Patch **decor3/ decorex3**):
... minoutdur (20) skiptime (1)
sndf (1) sndl (1) ming (36) maxg (64) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (10) gpsizehi (25) gppaklo (40) gppakhi (125) gpranglo (2) gpranghi (dx3gprhi.brk) centring (0)

```
texture postdecor 3 horn decorex3 ndfdec3.txt 20 1 1 1 36 64 1.0 1.0
0 1 1 0 0 10 25 40 125 2 dx3gprhi.brk 0
```

In Example 3, *skiptime* is in fact never used because the node times are spread out to fill the whole *outdur*, so the node sequence doesn't get a chance to repeat.

End of TEXTURE DECORATED / PREDECOR / POSTDECOR

TEXTURE MOTIFS – A texture of fully defined motifs attached to pitches selected at random from a pitch range or from a Harmonic Field/Set (MOTIFS: only first note of motif constrained, MOTIFSIN: all notes of motif constrained); one or more input sounds

Usage

texture motifs|motifsin mode *infile [infile2 ..] outfile notedata outdur packing scatter tgrid sndfirst sndlast mingain maxgain minpich maxpich phgrid grpspace gpsprange amprise contour multlo multhi* [**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*] [**-w**] [**-d**] [**-i**]

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 [not used by MOTIFSIN]

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line; a pitch value must be given for each *infile* (separated by a space). 60=original pitch.

2. In Mode **5** this is followed by a NOTELIST defining a motif, specified like this:
   - **#***N* (where *N* is the number of pitches in the motif).
   - a motif specified thus: *time* (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *sustain* (secs).
     Different start times must be given, and they must must increase; all 0's for chords are not allowed.
     This is all that is needed in Mode 5. There may be more than one motif defined in this way.

3. ALTERNATIVELY: In Modes **1 – 4**, a harmonic field/set notelist must be defined BEFORE the motifs are listed. This is in the format:

- **#***N* (where *N* is the number of pitches in the notelist).
- a harmonic field specified thus: *time* (secs) *infile_no pitch* (MIDI) *amp* (MIDI) *sustain* (secs).
  In the harmonic set/field(s) *amp* and *sustain* are inactive fields. Also, in the harmonic field/set definition, *time* can be all zeros in Modes **1** and **3**, whereas in Modes **2** and **4** different times are given when the field's pitch contents change at specified times.
- **The motif(s) themselves must now be defined, exactly as above.**

4. **In the motifs, *time*, *amp* and *sustain* and all active fields**. Thus the motif is able to be very precisely defined.

5. Any number of motifs may be defined in this way, which is also true in Mode **5**.

Form: **MPV/M/–/–**		Form: **MPV/HF-S/M/–**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
#*N*	No. of lines to follow	**#***N*	No. of lines to follow
time instr pitch amp dur	List of Motifs	*time instr pitch amp dur*	List of Harmonic Field/Set
		#*N*	No. of lines to follow
		time instr pitch amp dur	List of Motifs

outdur – minimum duration of the *outfile*
packing – (average) time between motif onsets
scatter – randomisation of event onsets (Range: 0 to 10)
tgrid – minimum step in milliseconds for a quantised time grid for motif start times (Default: 0)
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
minpich, maxpich – minimum and maximum pitch (MIDI note value)
phgrid – a timegrid in milliseconds applying WITHIN the motifs (Range: 0.0 to 1000.0)
gpspace – spatialisation of event-groups (Range: 0 to 5; Default: 1)

0 no change
1 scattered (Default)
2 motifs will move **towards** where the event is
3 motifs move **away from** where the event is
4 motifs follow the texture motion from Left to Right
5 motifs follow the texture motion from Right to Left

gpsprange – spatial range of event-groups (Range: 0 to 1; Default: 1)
amprise – amplitude change within motifs (Range: 0 to 127; Default: 0)
contour – amplitude contour of groups (Range: 0 to 6; Default: 0)

0 mixture of the other types (Default)
1 crescendo
2 flat (no change)
3 decrescendo
4 crescendo or flat
5 crescendo or decrescendo
6 decrescendo or flat

multlo, multhi – smallest & largest multiplier of total input duration of motif
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values
-d – motif notes all have the same duration as the ornamented note
-i – motifs are not confined to the instrument of the ornamented note (Default: same note)

Understanding the TEXTURE MOTIFS Process

*TEXTURE MOTIFS creates fully defined figures like the ornaments of TEXTURE ORNATE, but does not attach them to a user-defined Nodal Substructure. It attaches them either to pitches randomly selected from a pitch range, or from the pitches defined in a Harmonic Field/Set, again making a random selection of which pitches to use. There are interesting and useful differences between Mode **5** and Modes **1-4** and between MOTIFS and MOTIFSIN.*

The note data file

In Mode **5** there is no Harmonic Field/Set, and so the first section of the note data file is used for the motif itself, in which all fields are active. There can be more than one motif by adding another **#***N* and the motif note events. The pitches for each note event are selected at random from the range defined by *minpich* - *maxpich* (the pitch range parameters).

In the other Modes, the Harmonic Field/Set comes first, followed by the motif(s). In the Harmonic Field/Set, the *time* field uses zeros for Modes **1** and **3** (the motif, with its own internal timing, repeats after *skiptime* seconds, and the pitches are selected at random from the field). Changing times (ascending) are used for Modes **2** and **4** (only the pitches given for a certain time will be used during that time period – Example 4 illustrates this).

The difference in how the motif notes are attached by MOTIFS and MOTIFSIN

MOTIFS – Only the first note of the motif is 'forced onto the harmonic Field/Set'. This means that only the first note of the motif need be specified in the field – all the other notes of the motif are accurately transposed to follow on from there. You hear the first note of the motifs moving about in the field, and the rest of the motif flowing nicely from it, with nothing missed out. This feature can be musically useful when intense repetitions of a (whole) motif are desired. This was not the case with DECORATED or ORNATE, where the Harmonic Field/Set had to contain all the notes which might be used when the decoration or ornament was transposed.

MOTIFSIN – In this case, **all** the notes of the motif are 'forced' onto the field (MOTIFSIN does not have a Mode **5**). The result of this is that all the other notes in the motif will come from the field. The software is very clever about this, and nothing is left out when a transposed motif calls for a pitch which isn't in the field – but the motif is altered in the process. What happens is that there are transpositions and repetitions of the motif pitches in order to recast the motif to use exclusively the pitches of the field. This muddles the shape of the motif quite a bit, which can be musically useful when you want to develop a texture using a motif which transforms in unpredictable ways.

Musical Applications

All the examples for TEXTURE MOTIFS can be made by running *motifexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *motifdel.bat*.

EXAMPLE 1

Our first example illustrates Mode **5** by playing a defined motif on pitches randomly selected from a pitch range. There is therefore no Harmonic Field/Set, so the note data file *ndfmot1.txt* gets straight to the definition of the motif itself:

```
60
#10                         (10-event motif)
0.0 1 60 70 0.3
0.1 1 61 50 0.3
0.2 1 66 50 0.3
0.3 1 67 50 0.3
0.4 1 66 70 0.3
0.5 1 67 60 0.3
0.6 1 70 60 0.3
0.7 1 66 60 0.3
0.8 1 72 80 0.3
0.9 1 66 50 0.3
```

MOTIFS Example 1 (Preset/Patch **motif1/motifex1**):
... outdur (12) packing (1) scatter (0) tgrid (0)
snd1st (1) sndlast (1) ming (30) maxg (90)
minp (48) maxp (84) phgrid (0) grpspace (1) gpsprange (1)
amprise (0) contour (0) multlo (1) multhi (1)
[-aatten -pposition -sspread -rseed -w -d -i]

```
texture motifs 5 marimba motifex1 ndfmot1.txt 12 1 0 0 1 1 30 90 48
84 0 1 1 0 0 1 1
```

What we hear is the defined motif repeating every second, beginning on different pitches selected from the pitch range 48 to 84 (3 octaves). By using the *seed* option, you can produce different *but reproduceable* outputs each time, if you want to explore different random selections from the pitch range. You can add more motifs in the note data file, and the program will select among them as it builds the output soundfile.

EXAMPLES 2a/b/c

These three examples make use of Mode **3**. The note data file *ndfmot2.txt* starts by defining a Harmonic Field/Set and then defines a quick-note motif which lasts for 1 second. *Skiptime* is set to 1 second, so the repeats (at different transpositions) follow on without a break.

```
60
#9                          (Harmonic Field/Set)
0.0 1 60 0 0
0.0 1 61 0 0
0.0 1 63 0 0
0.0 1 64 0 0
0.0 1 66 0 0
0.0 1 67 0 0
0.0 1 68 0 0
0.0 1 70 0 0
0.0 1 72 0 0
#10                         (10-event motif)
0.0 1 60 70 0.3
0.1 1 61 50 0.3
0.2 1 66 50 0.3
0.3 1 67 50 0.3
0.4 1 66 50 0.3
0.5 1 67 70 0.3
0.6 1 70 50 0.3
0.7 1 66 50 0.3
0.8 1 72 80 0.3
0.9 1 66 60 0.3
```

MOTIFS Examples 2a/b/c (Preset/Patch **motif2a, b, &c /motifex2a, b, &c** – 2b uses MOTIFSIN: harmonic):
... outdur (12) packing (1) scatter (0) tgrid (0)
snd1st (1) sndlast (1) ming (30) maxg (90)
minp (48) maxp (84) phgrid (0) grpspace (1) gpsprange (1)
amprise (0) contour (0) multlo (0.5) multhi (2)
[-aatten -pposition -sspread -rseed -w -d -i]

```
texture motifs 3 marimba motifex2a ndfmot2.txt 12 1 0 0 1 1 30 90 48
84 0 1 1 0 0 0.5 2
texture motifsin 3 marimba motifex2b ndfmot2.txt 12 1 0 0 1 1 30 90
48 84 0 1 1 0 0 0.5 2
texture motifs 3 marimba motifex2c ndfmot2.txt 12 0.25 0 0 1 1 30 90
48 84 0 1 1 0 0 0.5 2
```

What we hear:

- We hear a defined motif play, at varying tempos, on a random selection from the notes of a defined harmonic set. The first note of the motif is taken (at random) from the field, and (all) the rest of the motif follows on as is expected (i.e., transposed).
- The 'b' version uses MOTIFSIN with the same parameters, and we hear the motif being warped as it is recast to use *only* the pitches of the defined harmonic set. Some pitches may be omitted or repeated as a result of the warping. **Note how this differs from the use of Mode 3 in MOTIFS.**
- The 'c' version creates overlaps with a skiptime of 0.25.

Further work done in 2001 sheds a little more light on the above examples. It is important to realise that in both the 'a' and the 'b' versions, the start note is drawn *at random* from the harmonic field or set: that is, the motifs will repeat with transpositions. If there are overlaps, the motif and field/set need to be coordinated so that desired results are obtained.

I was seeking to have the motif repeat, **always beginning on the *same* pitch**, the pitch defined as the start-pitch of the motif itself, with or without overlaps. This can be done, I found, by defining the harmonic set, defining the motif with pitch and velocity (important to bring out rhythmic groups), and **then giving the start pitch of the motif as both the lower *and* upper pitch range values**. I expected it to play the motif by repeating only one pitch, but in fact all the pitches of the motif are used.

TMOTIFS Preset Examples 2a and 2b *appear* to be doing this (starting on the same pitch), but this is not in fact happening. It appears to be happening because both the harmonic set and the motif are constructed with intervals of a minor-3rd, so it all harmonises no matter which start pitch of the harmonic set is used. One must use the same technique described above (both low and high pitch range = the start pitch of the motif) for the motif to repeat starting each time on the same pitch. A motif structured with different intervals would reveal this. (AE)

EXAMPLES 3a/b/c

We now introduce two inputs (marimba & horn) and two motifs. The second motif uses longer note values than the first. The marimba is slightly lower than the horn in pitch, so we lower its reference MIDI note value to 59. The note data file *ndfmot3.txt* below defines the Harmonic Field/Set first, and then the two motifs. Again the times in the field are all zero for use with Mode **3** and *skiptime* determines the time between repetitions of the motif.

```
59 60
#9                          (Harmonic Field/Set)
0.0 1 60 0 0
0.0 1 61 0 0
0.0 1 63 0 0
0.0 1 64 0 0
0.0 1 66 0 0
0.0 1 67 0 0
0.0 1 68 0 0
0.0 1 70 0 0
0.0 1 72 0 0
#10                         (motif 1)
0.0 1 60 70 0.3
0.1 1 61 50 0.3
0.2 1 66 50 0.3
0.3 1 67 50 0.3
0.4 1 66 50 0.3
0.5 1 67 70 0.3
0.6 1 70 50 0.3
0.7 1 66 50 0.3
0.8 1 72 80 0.3
0.9 1 66 60 0.3
#3                          (motif 2)
0.0  1 60 40 1.5
0.34 1 67 45 1.5
0.67 1 72 50 1.5
```

MOTIFS Examples 3a/b/c (Preset/Patch **motif3a, b, &c /motifex3a, b, &c**)
... outdur (12) packing (1) scatter (0) tgrid (0)
snd1st (1) sndlast (1) ming (30) maxg (90)
minp (48) maxp (84) phgrid (0) grpspace (1) gpsprange (1)
amprise (0) contour (0) multlo (0.5) multhi (2)
[-aatten -pposition -sspread -rseed -w -d -i]

```
texture motifs 3 marimba horn motifex3a ndfmot3.txt 12 1.0 0 0 1 2 30
90 48 84 0 1 1 0 0 0.5 2
texture motifs 3 marimba horn motifex3b ndfmot3.txt 12 0.5 0 0 1 2 30
90 48 84 0 1 1 0 0 1 1
texture motifs 1 marimba horn motifex3c ndfmot3.txt 12 0.5 0 0 1 2 30
90 48 84 0 1 1 0 0 1 1
```

We hear:

- Our 'a' version plays the two motifs in varying orders and tempos, with the instrument playing each motif selected at random. The duration field of the second motif specifies 1.5 sec., which doesn't have any audible effect on the marimba sound, but when the horn plays, the sound continues. overlapping the entry of the next motif, even though each motif starts regularly at the start of each second (*skiptime*) is one, and each motif is 1 second long.

- The 'b' version takes away the tempo change (*multlo* and *multhi* are restored to 1) and introduces a 1/2 sec overlap. We begin to hear a bit of counterpoint as the two different rhythms overlap each other, sometimes with the same and sometimes with different instruments.

- The 'c' version repeats the previous command with Mode 1, which opens up different octaves, further enriching the texture in a way which seems to allow more motivic repetitions.

EXAMPLE 4

The final example for MOTIFS calls upon Mode 4 to illustrate how the start pitches of the motifs can be controlled by timing information in the Harmonic Field/Set. Only C-60 is allowed for the first two seconds, then only pitches C-61 and C-62 can be used (to attach the motifs to) during the next two seconds, etc. *Ndfmot4.txt* contains these alteraions. The tempo is increased (0.75) and a little overlap put in (*skiptime* = 0.5) to allow more entries/repetitions of the motifs within each time period.

```
59 60
#9                          (Harmonic Field/Set - changing times)
0.0 1 60 0 0
2.0 1 61 0 0
2.0 1 63 0 0
4.0 1 64 0 0
4.0 1 66 0 0
4.0 1 67 0 0
6.0 1 68 0 0
6.0 1 70 0 0
6.0 1 72 0 0
#10                         (motif 1)
0.0 1 60 70 0.3
0.1 1 61 50 0.3
0.2 1 66 50 0.3
0.3 1 67 50 0.3
0.4 1 66 50 0.3
0.5 1 67 70 0.3
0.6 1 70 50 0.3
0.7 1 66 50 0.3
0.8 1 72 80 0.3
0.9 1 66 60 0.3
#3                          (motif 2)
0.0  1 60 40 1.5
0.34 1 67 45 1.5
0.67 1 72 50 1.5
```

MOTIFS Example 4 (Preset/Patch **motif4/motifex4**):
... outdur (12) packing (0.5) scatter (0) tgrid (0)
snd1st (1) sndlast (2) ming (30) maxg (90)
minp (48) maxp (84) phgrid (0) grpspace (1) gpsprange (1)
amprise (0) contour (0) multlo (0.75) multhi (0.75)
[-aatten -pposition -sspread -rseed -w -d -i]

```
texture motifs 4 marimba horn motifex4 ndfmot4.txt 12 0.5 0 0 1 2 30
90 60 72 0 1 1 0 0 0.75 0.75
```

We hear a fast moving texture of motifs with a lot of repetition *on the same or nearby pitches*, the whole texture moving upwards in pitch every two seconds. This provides a mechanism for designing overall pitch movement in a motif-based texture.

End of TEXTURE MOTIFS / MOTIFSIN

TEXTURE ORNATE|PREORNATE|POSTORNATE – A texture of events with fully user-specified ornaments placed on a 'line', optionally with pitches restricted to a Harmonic Field/Set; one or more input sounds

Usage

texture ornate|preornate|postornate mode *infile [infile2 ..] outfile notedata outdur skiptime sndfirst sndlast mingain maxgain mindur maxdur*
phgrid grpspace gpsprange amprise contour
multlo, multhi
[**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*]
[**-w**] [**-d**] [**-i**] [**-h**] [**-e**]

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

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line; a pitch value must be given for each *infile* (separated by a space). 60=original pitch.
2. followed by a 'line' substructure NOTELIST, specified thus:
   - **#***N* (where *N* is the number of pitches in the notelist).
   - In all Modes, this is followed by *N* lines to define the nodal substructure on which the ornaments will be placed. These are in the format *time* (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *dur* (secs). *Amp* and *dur* are inactive fields in TEXTURE ORNATE in the node substructure section.
   - different start times must be given, and they must must increase; all 0's for chords are not allowed.
3. In Modes **1–4** this is followed by *N* lines to define the harmonic field/set in the format: *time* (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *dur* (secs). *Starttime*, *Amp* and *dur* are inactive fields in TEXTURE ORNATE in the harmonic field/set section. **This section is omitted for Mode 5**.
4. Now the ornament itself must be defined. Again, this is in the form *time* (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *dur* (secs) – **in which *time*, *amp* and *dur* and all active fields.** Thus the ornament is able to be very precisely defined.

Form: **MPV/NS/O/–**		Form: **MPV/NS/HF-S/O**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
#*N*	No. of lines to follow	***#****N*	No. of lines to follow
time instr pitch amp dur	List of 'line' (Nodal Substructure)	*time instr pitch amp dur*	List of 'line' (Nodal Substructure)
#*N*	No. of lines to follow	***#****N*	No. of lines to follow
time instr pitch amp dur	List of Ornaments	*time instr pitch amp dur*	List of Harmonic Field/Set
		#*N*	No. of lines to follow
		time instr pitch amp dur	List of Ornaments

outdur – minimum duration of the *outfile*
skiptime – time between repeats of motif-to-ornament in *notedata*, i.e., between runs of the 'line' substructure notelist
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
mindur, maxdur – minimum and maximum duration of events in texture
phgrid – a timegrid in milliseconds applying WITHIN the ornaments (Range: 0.0 to 1000.0)
gpspace – spatialisation of event-groups (Range: 0 to 5; Default: 1)

0 no change
1 scattered (Default)
2 ornaments will move **towards** where the event is
3 ornaments move **away from** where the event is
4 ornaments follow the texture motion from Left to Right
5 ornaments follow the texture motion from Right to Left

gpsprange – spatial range of event-groups (Range: 0 to 1; Default: 1)
amprise – amplitude change within ornaments (Range: 0 to 127; Default: 0)
contour – amplitude contour of groups (Range: 0 to 6; Default: 0)

0 mixture of the other types (Default)
1 crescendo
2 flat (no change)
3 decrescendo
4 crescendo or flat
5 crescendo or decrescendo
6 decrescendo or flat

multlo, multhi – smallest & largest multiplier of total input duration of motif
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values
-d – ornament notes all have the same duration as the ornamented note

-i – ornaments are not confined to the instrument of the ornamented note (Default: same note)
-h – ornaments on **highest** note of any chord (Default: on first note listed)
-e – ornaments on all the notes of any chord

Understanding the TEXTURE ORNATE Process

The salient feature in TEXTURE ORNATE is that an ornament is attached to an underlying substructure node. The ornament is a precisely defined musical figure: the start time, amplitude and duration fields in the ornament definition part of the note data file are all active. Individual ornament note durations may be longer than the time until the next ornament note start time, which makes fades and legato effects possible. Amplitude values will be important to shape the ornament rhythmically in an audible way.

All the examples for TEXTURE ORNATE can be made by running *ornatexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *ornatdel.bat*.

MODE 5

The ornament has specified start times, amplitude and durations for each of its notes. A 'turn' (upper and lower neighbors around a centre) could look like this, a quintuplet over the time of a crotchet:

```
#5
0.0 1 60 84 0.3
0.2 1 62 64 0.3
0.4 1 60 68 0.3
0.5 1 58 72 0.3
0.8 1 60 76 0.3
```

In Mode **5** this is all that is needed. This ornament will be automatically transposed in both time and pitch so that it begins on each of the nodes of the substructure melody line defined in the first part of the note data file. Because no harmonic field is defined, the full chromatic set is used by default. Thus the complete file *ndforn1.txt* could look like this:

```
60
#4                      (node substructure)
0.0 1 60 0 0
2.0 1 67 0 0
4.0 1 65 0 0
6.0 1 62 0 0
#5                      (ornament definition)
0.0 1 60 84 0.3
0.2 1 62 64 0.3
0.4 1 60 68 0.3
0.5 1 58 72 0.3
0.8 1 60 76 0.3
```

ORNATE Example 1 (Preset/Patch **ornate1/ornatex1**):
... minoutdur (12) skiptime (2)
sndf (1) sndl (2) ming (30) maxg (90) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
multlo (1) multhi (1)

```
texture postornate 5 marimba ornatex1 ndforn1.txt 12 2 1 1 30 90 1.0
1.0 0 1 1 0 0 1 1
```

The **tempo and time-relationship** between repeats of the ornament offers many possibilities. In the above file, the ornament lasts 1 second, and there are 2 seconds between substructure nodes. Thus there will be gap of 1 second before the next ornamented node. The speed (tempo) *of the ornaments* can be altered with *multlo/hi*: 1 is no change (e.g., crotchet/quarter-note = 60), 0.5 doubles the speed (e.g., crotchet/quarter-note = 120) etc. **However, this does not change the time *between* the ornaments.**

Although you can enter ornament timings with any values, I (AE) find it useful to use 1 second (crotchet/quarter-note = 60) as a reference. Then standard values for durations, such as 0.5 for ½ sec (quaver/ eighth-note), can be used, maintaining a conventional musical understanding of beats. This is normally how it is done in *Csound*, for example, with a tempo control to alter the actual rate of flow. And here in TEXTURE, the formula to calculate a tempo ratio (multiplier) for the ornaments will be **60 divided by the desired tempo**. E.g.,

- 60/72 = 5/6 = 0.833
- 60/120 = 1/2 = 0.5
- 60/90 = 2/3 = 0.667
- 60/84 = 5/7 = 0.714
- 60/76 = 15/19 = 0.789
- etc.

Skiptime is the time between runs of the whole node substructure. **It begins at the start time of the last substructure node**. For no overlap, the **rule of thumb** is:

> **'the last ornament start time *plus* the length of time after this until the next ornament is to begin'.**

The actual duration of this note (in the *duration* field of the note data file) may be longer/shorter than this to achieve a legato/staccato effect.

In the above example the total ornament time would be 0.8 + 0.2 to make 1 full second so that it will stay 'on the beat'. Thus, for the whole sequence to repeat seamlessly without a break (or an overlap), *skiptime* would be 1, 1 being the full time we want for the ornament. You can specify any *outdur*, so the sequence repeats until *outdur* is reached. *Skiptimes* less than 1 will, therefore, cause overlaps. We shall explore this in greater detail in Example 3 below.

Modes 1 to 4

You therefore have a great deal of control over the design of your ornaments. In these Modes, you can define a **harmonic field or set** so that only these pitches will be used. Note that you have to take care that all the pitches are available to enable the ornament to transpose to all nodes. If you hear any missing or incorrectly repeated ornament notes, the cause is usually a pitch missing in the harmonic field/set. The transposition within the program is done by simple addition, not by indexing, which accounts for this restriction.

It is possible to have **more than one ornament**. These are defined in the same way (starting with the *#N* separator). The start time of each ornament in the note data file will be zero. They are then attached to nodes at the times when the nodes are set to begin. The program will select from the list of ornaments in random order. If you want this random order to vary on different runs of the program, *but be reproduceable*, use the *seed* flag [**-r**] with specific values greater than 1.

More than one input sound file may be used. Provide a MIDI pitch for each of them in the note data file and adjust *last snd-in-list to use* accordingly. **You can, in *all* the Texture programs, make the value of the first- and last-sound-to-use vary over time (using a breakpoint file) so you can sweep through a list of input sound files as the texture progresses**. Different ornaments will use different sounds. If you specify the **Scatter decor instrs** flag [**-i**], the various notes within an ornament will randomly use different sounds. (Specifying specific instruments for specific ornament notes in the note data file has no effect, and specifying instruments for substructure nodes badly confuses the program, although it doesn't complain).

Remember that it will often be very important to shape the ornament with the **amplitude** field. The notes will just run on together without any aural definition unless accents, crescendos and decrescendos are introduced.

Musical Applications

We shall present three examples here, all of which use Mode **3** so that the full note data file is used (all three sections).

ORNATE Example 2

ORNATE Example 2 defines a harmonic set which enables a little figure to be accurately transposed onto two substructure nodes: C-60 and G-67. The note data file is *ndforn2.txt* as follows:

```
60 59
#2                          (the nodal substructure)
0.0 1 60 0 0
2.0 1 67 0 0
#8                          (the harmonic set definition)
0.0 1 60 0 0
0.0 1 62 0 0
0.0 1 63 0 0
0.0 1 65 0 0
0.0 1 67 0 0
0.0 1 69 0 0
0.0 1 70 0 0
0.0 1 72 0 0
#8                          (the ornament definition)
0.0    1 60 30 0.5
0.25   1 62 30 0.5
0.5    1 63 30 0.5
0.8333 1 62 30 0.5
1.0    1 63 30 0.5
1.25   1 65 30 0.5
1.5    1 63 30 0.5
1.75   1 62 30 0.5
```

Two input soundfiles are used.

ORNATE Example 2 (Preset/Patch **ornate2/ornatex2**):
... minoutdur (12) skiptime (2)
sndf (1) sndl (2) ming (30) maxg (90) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
multlo (1) multhi (1)

```
texture postornate 3 marimba horn ornatex2 ndforn2.txt 12 2 1 2 30 90
1.0 1.0 0 1 1 0 0 1 1
```

What we hear is a series of repetitions of the same ornament, placed alternately on the C and the G nodes. Sometimes the whole ornament is played by the horn and sometimes by the marimba – a random selection of the instrument is being made, but once made, it plays the whole ornament. Were we to use the **-i** flag, this random selection of instrument would be made *for each note of the ornament*, an altogether different result not unlike a Medieval hocket. If *instr_no* is specified for the various ornament notes, this has no effect on these results, with or without the **-i** flag.

> The actual pitch level of the ornaments could be inaccurate if the two source sounds were actually at different pitches and the same MIDI pitches in the note data file were given for each of them. If accuracy is important, you might be able to adjust one of the the MIDI pitch values given in the note data file (remember, you don't have to use whole numbers), or you could retune one of the source sounds.

We should also observe that the various repetitions follow on at regular intervals. The nodes are spaced at two seconds, the ornament takes 2 seconds, and *skiptime* is set at two seconds, which relates to the period between runs of the *whole sequence of the nodal substructure*. If we made *skiptime* 3 seconds, we would hear two ornaments, one on C and one on G, and then a pause of 1 second. Why 1 second? This is important: because the *skiptime* begins at the start time of the last node of the substructure. The ornament lasts 2 seconds, so there will be one more second of silence before the sequence repeats.

ORNATE Example 3

This example creates several different ornaments and attaches them to a stepwise rising nodal substructure. Again, the ornaments are selected in random order. Again, the harmonic set definition has to have all the notes it needs to play all the ornaments transposed to each of the nodes. This implies a compositional task to consider how to create ornaments which will follow one another in the manner you wish when they occur in various orders at various transposition levels.

The now much longer note data file *ndforn3.txt* is as follows:

```
60
#5                          (node substructure)
0.0  1 60 0 0
3.0  1 62 0 0
6.0  1 63 0 0
9.0  1 65 0 0
12.0 1 67 0 0
#17                         (harmonic set definition)
0.0 1 60 0 0
0.0 1 62 0 0
0.0 1 63 0 0
0.0 1 64 0 0
0.0 1 65 0 0
```

```
0.0 1 66 0 0
0.0 1 67 0 0
0.0 1 68 0 0
0.0 1 69 0 0
0.0 1 70 0 0
0.0 1 71 0 0
0.0 1 72 0 0
0.0 1 73 0 0
0.0 1 74 0 0
0.0 1 75 0 0
0.0 1 76 0 0
0.0 1 77 0 0
#8                          (ornament 1)
0.0    1 60 60 0.5
0.25   1 62 55 0.5
0.5    1 63 65 0.5
0.8333 1 62 55 0.5
1.0    1 63 70 0.5
1.25   1 65 65 0.5
1.5    1 63 60 0.5
1.75   1 62 55 0.5
#8                          (ornament 2)
0.0    1 60 70 0.5
0.2    1 63 72 0.5
0.4    1 62 74 0.5
0.6    1 65 76 0.5
0.8    1 63 78 0.5
1.0    1 66 90 0.5
1.5    1 67 85 0.5
1.75   1 66 85 0.5
#6                          (ornament 3)
0.0    1 60 60 0.5
0.34   1 63 50 0.5
0.67   1 67 50 0.5
1.00   1 62 60 0.5
1.25   1 62 45 0.5
1.50   1 66 50 0.5
#6                          (ornament 4)
0.0    1 60 50 0.5
0.25   1 67 60 0.5
0.75   1 60 40 0.5
1.00   1 65 70 0.5
1.50   1 65 70 0.5
1.75   1 62 65 0.5
#14                         (ornament 5)
0.0   1 60 40 0.5
0.125 1 62 45 0.5
0.25  1 63 50 0.5
0.375 1 65 55 0.5
0.5   1 67 60 0.5
0.625 1 65 55 0.5
0.75  1 63 50 0.5
0.875 1 62 45 0.5
1.0   1 63 70 0.5
1.17  1 65 65 0.5
1.33  1 67 70 0.5
1.5   1 65 65 0.5
1.66  1 63 70 0.5
1.83  1 62 70 0.5
```

To help read it and write out the ornaments in musical notation, here's a quick review of key duration values (and **Note Chart** lists MIDI note vales):

- 1.0 = 1 crotchet (quarter note)
- 0.75 = a dotted quaver (dotted eighth note)
- 0.5 = 1 quaver (eighth note)
- 0.25 = 1 semiquaver (16th note)
- 0.125 = 1 demisemiquaver (32nd note)
- 0.33 = 1 quaver in a triplet (.33 + .33 + .34 in any order)
- 0.166 = 1 semiquaver in a triplet (.167 + .167 + .166 in any order)
- 0.66 = 1 crotchet in a triplet (.67 + .67 + .66 in any order)

Note that all the ornaments start at time zero. They take their actual start time from the node to which they become attached. Note that they are also transposed to that node. These are all the parameters in command line format:

ORNATE Example 3 (Preset/Patch **ornate3/ornatex3**):
... minoutdur (21) skiptime (4)
sndf (1) sndl () ming (30) maxg (90) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
multlo (0.5) multhi (1.2)

```
texture postornate 3 marimba ornatex3 ndforn3.txt 21 4 1 1 30 90 1 1
0 1 1 0 0 0.5 1.2
```

What we hear is a sequence of different ornaments, probably not in the order in which the appear in they note data file. You will also notice that each ornament is at a slightly different tempo. This is because the *mult* parameter has been given a range from less than 1 (0.5) to greater than 1 (1.2).

ORNATE Example 4

This example plays with phasing the same ornament with precise rhythmic control, which enables us to create specific interval relationships in the overlapping figures. To do this, we design a single ornament for the purpose, in this case a rising and falling scale. This is note data file *ndforn4.txt* with one node, a harmonic set and the scale ornament:

```
60
#1                          (single node)
0.0  1 60 0 0
#8                          (scale as harmonic set)
0.0  1 60 0 0
0.0  1 62 0 0
0.0  1 63 0 0
0.0  1 65 0 0
0.0  1 67 0 0
0.0  1 69 0 0
0.0  1 70 0 0
0.0  1 72 0 0
#16                         (ornament:  rising and falling scale)
0.0  1 60 50 0.3
0.25 1 62 55 0.3
0.5  1 63 60 0.3
0.75 1 65 65 0.3
1.0  1 67 70 0.3
1.25 1 69 75 0.3
1.5  1 70 80 0.3
```

```
1.75 1 72 85 0.3
2.0  1 72 85 0.3
2.25 1 70 85 0.3
2.5  1 69 85 0.3
2.75 1 67 85 0.3
3.0  1 65 85 0.3
3.25 1 63 85 0.3
3.5  1 62 85 0.3
3.75 1 60 85 0.5
```

ORNATE Example 4 (Preset/Patch **ornate4/ornatex4**):
... minoutdur (12) skiptime (0.5)
sndf (1) sndl (1) ming (30) maxg (90) mind (1.0) maxd (1.0)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
multlo (1) multhi (1)

```
texture postornate 3 marimba ornatex4 ndforn4.txt 21 0.5 1 1 30 90 1
1 0 1 1 0 0 1 1 -a0.75
```

What we hear is a dense texture of rising and falling scale pattern spaced at the interval of a third. The density of overlap has led to overload, so the attenuation parameter [**-a**] is also used. Fortunately, the program itself calculates the attenuation which may be needed and suggests a figure to the user.

There are two methods by which the overlaps can be achieved.

1. The most direct (as above) is to create only one node and adjust the skip time. Our ornament in this case is 4 seconds long, so a *skiptime* of 4 would cause regular repeats of the whole ornament on the beat without overlap or gap. When we reduce *skiptime* to 0.5, the ornament begins again at MIDI note 60 at the same time that the first one is playing MIDI note 63 (at 0.5 sec), thus creating the interval of a third. The original and repeated ornaments will proceed upwards in thirds. However, after another ½ second, another ornament will start, creating a whole texture of thirds out of the phased ornament. A *skiptime* of 0.25 will (with this ornament) produce 2^{nds} and 1.0 will result in 5^{ths}, and so it goes.

2. The second method involves creating more than one node, which necessarily have to be at different times. If in this case both were at MIDI note 60 and the second node began at 0.5, with *skiptime* set to 4.0 (full ornament duration), we will hear the ornament begin again after half a second and produce thirds. No additional ornament repeats will come in, so we will hear the second one end by itself, but the whole sequence will repeat after 4 seconds until the specified *outdur* is reached. There will be a slight overlap at the repeats of the whole node sequence because of the second ornament starting later. This method gives more control over density and the pitch level of each repeat.

If we want to make use of the tempo control as well (*mult*), we might also want to adjust *skiptime* to reflect the new tempo in order to preserve the same time relationships. Otherwise, the ornament(s) will play at different speeds, but the *skiptime* will stay the same, which will alter the timing relationship of the ornaments as the underlying node sequence repeats.

Preserving time relationships while using a time-varying tempo can be improved by multiplying the desired *skiptime* by the *mult* factor. When a tempo *range* is used things do seem to get out of step, but it is possible to keep things somewhat more in step when *breakpoint files* are used: the breakpoint times are matched in the *skiptime* file, and its values are multiplied by the corresponding *mult* factor.

I had hoped that the phased ornaments would keep perfectly in step while the tempo waxed and waned, but this does not appear to be happening, so there does seem to be some degree of offsetting taking place as well.

Here are some further experiments you can carry out with the timing of the ornament. The following are suggested alterations to the (command line) parameters as given for Example 3 in *ornexs.bat* with the note data file *ndforn4.txt* as above.

- **time-varying *skiptime*** – *oex3skip.brk*

```
0       4
12      0.5
```

  sets a *skiptime* of 4 at time 0 and 0.5 (creates the 3rds) at time 12. **We hear** a full run of the ornament by itself; the ornament begins again, and then the first few notes of the next run come in a little before the end of the first, and subsequent repeats move the overlap forward until the 0.5 time mark is reached.

- **time-varying tempo: accel** – both the low and high tempo parameters use the same breakpoint data: *oex3mulo/hi.brk*:

```
0       1
12      0.5
```

  Thus we have 1 (no change) at time zero and 0.5 (twice as fast) at time 12; *skiptime* is reset to 4 as above. **We hear** a gradual speedup during the course of the output soundfile as the ornaments repeat (without overlap).

- **time-varying tempo: Accel/decel** – Similarly, moving from 1 at time 0 to 0.5 at time 6 and back to 1 and time 12 will result in an accel/decel over the course of the output soundfile: revised *oex3mulo/hi*:

```
0       1
6       0.5
12      1
```

  We hear the ornaments move faster-slower, but also notice that the time between the ornaments appears to alter. This paradoxically is because the skip time is constant. What is happening is that the faster ornaments take less time and end sooner, but the next run is still set to begin after 4 seconds, so we have longer and shorter periods of silence, although the actual time between repeats is staying the same.

- **combining time-varying tempo with overlaps set by the *skiptime***: revised *oex3skip.brk*:

```
0       4
6       0.5
12      4
```

  Now **we hear** the ornaments overlap in a complex way (notes are not in sync) as the time-varying *skiptime* gradually moves the overlap forward from 4 seconds and back again.

- **skip times adjusted by tempo ratio** – To get a better (not not perfect) preservation of ornament overlap relationships while changing tempo, we multiply the skip time values by the tempo factors. Let's see what happens when we do this with the two previous experiments:

```
[oex3skip.brk * oex3mulo/hi = new skiptime]
0   4    *   1     =    4
6   0.5  *   0.5   =    0.25
12  4    *   1     =    4
```

  Now the *skiptime* is reduced more in step with the tempo. **We hear** not a perfect synchronisation, but considerably closer temporal relationships than before.

- **sustained tones within ornaments** – Try using the horn.mp3 source and adding a 2 second C-60 at the start (bottom) of the scale and a 2 second C-72 at the top. This introduces a contrapuntal dimension which further illustrates potential applications, esp. within the context of music based on harmonic fields.

- **mixed rhythms** – Ornaments with more varied rhythmic contents open up the possibility of a more strongly shaped gestural and motivic music.

End of TEXTURE ORNATE / PREORNATE / POSTORNATE

TEXTURE TIMED – A texture with events constrained to a rhythmic template and pitches selected at random from a pitch range or a Harmonic Field/Set; one or more input sounds

Usage

texture timed mode *infile [infile2...] outfile notedata outdur skiptime sndfirst sndlast mingain maxgain mindur maxdur minpich maxpich* [**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*] [**-w**]

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

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line (60=original pitch)
2. followed by a times NOTELIST, specified thus:
    - **#***N* (where *N* is the number of events in the notelist).
    - This is followed by *N* lines to define a rhythmic template in the format: *time* (secs) *infile_no pitch* (MIDI) *amp* (MIDI) *dur* (secs) **NB:** Only the times of the motif are active, although pitch and loudness must also be specified (arbitrary values are OK); durations may be 0.
    - the specified times within the motif must increase
3. followed (optionally) by another notelist to specify a harmonic field or set (not used in Mode **5**).
    - these begin with **#***N* (where *N* is the number of events in the notelist). This also acts as a 'separator', enabling the program to know where the timed motif ends and the harmonic field or set begins.
    - in the list of note events, all start times will be zero for a given harmonic field or set (Modes **1** and **3**).
    - for Modes **2** and **4** give later start times for the changing harmonic field(s) or set (s).

Form: **MPV/T/–/–**		Form: **MPV/T/HF-S/–**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
#N	No. of lines to follow	***#N***	No. of lines to follow
time instr pitch amp dur	List of times	*time instr pitch amp dur*	List of times
		#N	No. of lines to follow
		time instr pitch amp dur	List of Harmonic Field/Set

outdur – minimum duration of the *outfile*
skiptime – time between repetitions of timing motif in the note data file (from the end of one motif to the beginning of the next; NB: cannot produce overlaps with values shorter than the motif itself)
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
mindur, maxdur – minimum and maximum duration of events in texture
minpich, maxpich – minimum and maximum pitch (MIDI note value)
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values

Understanding the TEXTURE TIMED Process

TEXTURE TIMED repeats a series of notes in random order but locked into a defined rhythmic template.

All the examples for TEXTURE TIMED can be made by running *timedexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *timeddel.bat*.

The key to this process lies in realising that the first part of the note data file consists of note events in which **only the onset timing is relevant** (the other three fields need something in them, however, 1 for the *instr_no* and usually zeros for the other fields). **You are therefore specifying a *rhythmic* template** which can be filled with any sequence of pitches.

This template will draw upon the parameter pitch range in random fashion in Mode **5**. For example, *ndftim1.txt* contains only:

```
60
#6                (rhythmic template definition)
0.00 1 0 0 0
0.34 1 0 0 0
0.67 1 0 0 0
1.00 1 0 0 0
1.50 1 0 0 0
1.75 1 0 0 0
```

TIMED Example 1 (Preset/Patch **timed1/timedex1**):
sndfirst (1) sndlast (1) mingain (24) maxgain (84)
mindur (0.2) maxdur (1.0) minpich (48) maxpich (84)

```
texture timed 5 marimba timedex1 ndftim1.txt 12 2.0 1 1 24 84 0.2 1.0
48 84
```

We hear the wide pitch range from which the notes for the rhythmic motif are selected. We also notice that *skiptime* is different. Here *skiptime* is the time **between statements of the rhythmic motif/template**. Thus we hear a long gap of two seconds between repeats because *skiptime* = 2.0 sec. Thus *skiptime* in TEXTURE TIMED begins *after the end of the motif*.

This is **not** like the *skiptime* in TEXTURE ORNAMENT, where *skiptime* is the time between repeats of the whole Nodal Substructure. *Skiptime* begins at the *start time of the last Node*. Thus, in TEXTURE ORNAMENT a 2 second ornament with a 2 second *skiptime* leads to a repeat without a gap. Also, where a *skiptime* less than the ornament duration results in overlaps. Here in TEXTURE TIMED the *skiptime* is the time between repetitions of the whole rhythmic template definition and there is no mechanism for creating overlapping rhythmic motifs (times less than 0 are not allowed).

In Modes **1-4**, the notes are randomly selected from only the pitches in a specified Harmonic Field/Set. In the latter case, you need an additional notelist in the note data file.

EXAMPLE 2. The following example (*ndftim2.txt*) makes clear how to do this (note the absence of pitch data in the list of times, and the identical (0.0) start times in the harmonic field/set note list):

```
60
#5                (rhythmic template definition)
0.00 1 0 0 0
0.25 1 0 0 0
0.75 1 0 0 0
1.00 1 0 0 0
1.50 1 0 0 0
#6                (harmonic field/set definition)
0.0 1 60 0 0      (omitted in Mode 5)
0.0 1 62 0 0
0.0 1 65 0 0
0.0 1 67 0 0
0.0 1 70 0 0
0.0 1 72 0 0
```

The above note data file is run with the following set of parameters, using Mode **3**.

TIMED Example 2 (Preset/Patch **timed2/timedex2**):
... outdur (12) skiptime (2.0)
sndfirst (1) sndlast (1) mingain (24) maxgain (84)
mindur (0.2) maxdur (1.0) minpich (48) maxpich (84)

```
texture timed 3 marimba timedex2 ndftim2.txt 12 2.0 1 1 24 84 0.2 1.0
48 84
```

EXAMPLE 3 - quick note gestures on a harmonic grid.
The 'b' example uses Mode **1** to open the texture out to other octaves.

ndftim3.txt

```
        60
        #17             (rhythmic template)
        0.00 1 0 0 0
        0.05 1 0 0 0
        0.10 1 0 0 0
        0.15 1 0 0 0
        0.20 1 0 0 0
        0.25 1 0 0 0
        0.30 1 0 0 0
        0.35 1 0 0 0
        0.40 1 0 0 0
        0.45 1 0 0 0
        0.50 1 0 0 0
        1.00 1 0 0 0
        1.50 1 0 0 0
        1.60 1 0 0 0
        1.70 1 0 0 0
        1.80 1 0 0 0
        1.90 1 0 0 0
        #5              (Harmonic Field/Set)
        0.0 1 48 0 0
        0.0 1 50 0 0
        0.0 1 53 0 0
        0.0 1 55 0 0
        0.0 1 58 0 0
```

TIMED Example 3a/b (Preset/Patch **timed3a, &b/timedex3a, &b**):
... outdur (12) skiptime (0.1)
sndfirst (1) sndlast (1) mingain (40) maxgain (80)
mindur (0.4) maxdur (1.0) minpich (48) maxpich (84)

```
texture timed 3 marimba timedex3a ndftim3.txt 12 0.1 1 1 40 80 0.4
1.0 48 84
texture timed 1 marimba timedex3b ndftim3.txt 12 0.1 1 1 40 80 0.4
1.0 48 84
```

We hear the rapid flows constrained to the C-D-F-G-Bb harmonic grid. In the absence of amplitude shaping, the ear inevitably begins to hear the longer notes at time 0.5 and 1.0 as the main foci, with the fast 5 .1 and faster 10 0.05 durations flowing around them. The 'b' example using Mode **1** opens out the octaves and creates a richer, warmer texture.

Musical Applications

TEXTURE TIMED requires a very specific set of note event timings, and yet makes its note selections randomly. It is therefore it is well suited to to slower and very distinctive rhythmic ideas.

The random reworking of the note selections can be a bit too obvious, however, so, to mitigate this, you might consider using longer note durations, a fairly large and sonorous harmonic field or set, and possibly several input soundfiles.

Alternatively, a vigorous texture of fast, repeated rhythms can be obtained with very closely placed timings and a *skiptime* equal to the length you want the note of the last rhythmic motif note event to have. Then the next motif will begin with a gap. If *skiptime* is near zero (min *skiptime* is 0.000002), there is no time allowance for the last note event of the motif, and the last note of one motif and the first of the next motif will be virtually simultaneous.

End of TEXTURE TIMED

TEXTURE TGROUPED – A texture with the onsets of separate internally shaped event groups constrained to a rhythmic template, with pitches drawn from a pitch range or a Harmonic Field/Set; one or more input sounds

Usage

texture tgrouped mode *infile [infile2 ..] outfile notedata outdur skip*
sndfirst sndlast mingain maxgain mindur maxdur minpich maxpich
phgrid grpspace gpsprange amprise contour
gpsizelo gpsizehi gppacklo gppackhi gpranglo gpranghi
[**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*]
[**-w**] [**-d**] [**-i**]

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

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line (60=original pitch)
2. followed by a times NOTELIST, specified thus:
   - **#***N* (where *N* is the number of events in the notelist).
   - This is followed by *N* lines to define a rhythmic template in the format:
     time (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *dur* (secs) **NB:** Only the times in the template are active, although pitch and loudness must also be specified (arbitrary values are OK); durations may be 0.
   - times in the template must increase
3. followed (optionally) by another notelist to specify a harmonic field or set (not used in Mode **5**):
   - these begin with **#***N* (where *N* is the number of events in the notelist). This also acts as a 'separator', enabling the program to know where the time template ends and the harmonic field or set begins.
   - in the list of note events, all start times will be zero for a given harmonic field or set (Modes **1** and **3**).
   - for Modes **2** and **4** give later start times for the changing harmonic field(s) or set (s).

Form: **MPV/T/–/–**		Form: **MPV/T/HF-S/–**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
#N	No. of lines to follow	***#N***	No. of lines to follow
time instr pitch amp dur	List of times	*time instr pitch amp dur*	List of times
		#N	No. of lines to follow
		time instr pitch amp dur	List of Harmonic Field/Set

outdur – minimum duration of the *outfile*
skip – time between repeats of timing motif in *notedata*, i.e., between runs of the 'line' substructure notelist
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
mindur, maxdur – minimum and maximum duration of events in texture
minpich, maxpich – minimum and maximum pitch (MIDI note value)
phgrid – a timegrid in milliseconds applying WITHIN the groups (Range: 0.0 to 1000.0)
gpspace – spatialisation of event-groups (Range: 0 to 5; Default: 1)

0 no change
1 scattered (Default)
2 groups will move **towards** where the event is
3 motifs move **away from** where the event is
4 groups follow the texture motion from Left to Right
5 groups follow the texture motion from Right to Left

gpsprange – spatial range of event-groups (Range: 0 to 1; Default: 1)
amprise – amplitude change within groups (Range: 0 to 127; Default: 0)
contour – amplitude contour of groups (Range: 0 to 6; Default: 0)

0 mixture of the other types (Default)
1 crescendo
2 flat (no change)
3 decrescendo
4 crescendo or flat
5 crescendo or decrescendo
6 decrescendo or flat

gpsizelo, gpsizehi – smallest & largest numbers of events in the groups
gppacklo, gppackhi – shortest & longest time between event-onsets in the groups
gpranglo, gpranghi – minimum & maximum pitch range of the groups OR, for harmonic field/sets, the range of notes within the field to use
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values

-d – fixed timestep between group notes
-i – each group is not each confined to a fixed instrument (Default: fixed)

Understanding the TEXTURE TGROUPED Process

TEXTURE TGROUPS is a natural (and wonderful) development of the rhythmic template idea in TEXTURE TIMED. In this case, the template times replace the packing parameter as the time-onsets of each group. Thus the groups can not only follow one another at regular time onsets, but these onsets can be have a defined rhythm. The groups themselves are shaped just as they are in TEXTURE GROUPED.

Once the idea is clear that the time-onsets of each group follow the rhythmic template defined in the first part of the note data file, understanding TGROUPED follows easily. We shall illustrate this by using the parameters in Mode **5** to make the rhythmic template audible, and then 'populate' it with groups.

All the examples for TEXTURE TGROUPED can be made by running *tgrouexs.bat* from the Console prompt. The soundfiles produced can be deleted with *tgroudel.bat*.

TGROUPED Example 1a

The first example sets out the rhythmic template in the note data file *ndftgr1.txt*, which is in fact an exact copy of *ndftim1.txt* used with TEXTURE TIMED. This now familiar rhythm is made completely audible by having a 'group' of only one event (*gpsizelo*, *gpsizehi*, *gpranglo* and *gpranghi* are all = 1). Here is the data:

```
[ndftgr1.txt]
60
#5
0.00 1 0 0 0
0.25 1 0 0 0
0.75 1 0 0 0
1.00 1 0 0 0
1.50 1 0 0 0
```

TGROUPED Example 1a (Preset/Patch **tgrouped1a/tgrouex1**):
... minoutdur skiptime
snd1st sndlast mingain maxgain mindur maxdur minpch maxpch
phgrid grpspace gpsprange amprise contour
gpsizelo gpsizehi gppaklo gppakhi gppranglo gppranghi
[-aatten -ppos -ssprd -rseed -w]

```
texture tgrouped 5 marimba tgrouex1a ndftgr1.txt 12 2.0 1 1 30 60 0.3
0.6 48 84 0 1 1 0 0 1 1 100 200 1 1
```

We hear the defined rhythm over two beats very clearly, because only one note is playing each of the rhythmic durations (the *grpsize* range is 1). The pitches are widely spaced because selected (at random) from a 3-octave pitch range. There is a two second pause (*skiptime*) between repeats of the rhythmic template.

TGROUPED Example 1b

Using the same note data file, we now increase the *grpsize* range to 3, using 3 for both low and high. Thus all the pitches which previously had 1 note on them, will now have three.

TGROUPED Example 1b (Preset/Patch **tgrouped1b/tgrouex1**):
(In *Sound Loom* use **tgrouex1**, making the parameter changes described below.)
... minoutdur skiptime
snd1st sndlast mingain maxgain mindur maxdur minpch maxpch
phgrid grpspace gpsprange amprise contour
gpsizelo gpsizehi gppaklo gppakhi gppranglo gppranghi
[-aatten -ppos -ssprd -rseed -w]

```
texture tgrouped 5 marimba tgrouex1b ndftgr1.txt 12 2.0 1 1 40 80 0.4
1.0 48 84 0 1 1 0 0 3 3 166 167 1 1
```

Group size high and low are reset to 3. The group packing is set between 166 and 167 milliseconds. In the context of this musical example, this corresponds to the duration of a semiquaver (16th note) triplet. We also make the example a little louder by changing *min, max event gain* to 40 and 80. **We hear** the triplets on each note – but notice that all the triplets have the same overall duration of one quaver (8th note). Thus the triplets which begin on semiquaver durations of the timing template will overlap with those which begin a semiquaver later.

TGROUPED Example 1c

Now we make things more variable, with the number of notes in each group ranging from 5 to 10 (*grpsize*), the packing between group note events ranging between 50 and 100 ms (*grppak*) and the number of pitches taken from the pitch range widening from 1 in the above 'a' and 'b' examples to a range of 3 to 10 (*gpranghi*).

TGROUPED Example 1c(Preset/Patch **tgrouped1c/tgrouex1**):
(In *Sound Loom* use **tgrouex1**, making the parameter changes described above.)
... minoutdur skiptime
snd1st sndlast mingain maxgain mindur maxdur minpch maxpch
phgrid grpspace gpsprange amprise contour
gpsizelo gpsizehi gppaklo gppakhi gppranglo gppranghi
[-aatten -ppos -ssprd -rseed -w]

```
texture tgrouped 5 marimba tgrouex1c ndftgr1.txt 12 2.0 1 1 40 80 0.4
1.0 48 84 0 1 1 0 0 5 10 50 100 3 10
```

We hear a much more fluid texture of overlapping events. While we may sense some variation in density, one side effect of the ranges used is that the onsets of the rhythmic template can no longer be easily perceived.

TGROUPED Example 1d

The final version of this parameter set simply reduces the *skiptime* to ½ a second. This draws the repeats much closer together, thus increasing the overlap of the groups as they play themselves out, even though the rhythmic templates themselves do not overlap.

TGROUPED Example 1d(Preset/Patch **tgrouped1d/tgrouex1**):
(In *Sound Loom* use **tgrouex1**, making the parameter changes described above.)
... minoutdur skiptime
snd1st sndlast mingain maxgain mindur maxdur minpch maxpch
phgrid grpspace gpsprange amprise contour
gpsizelo gpsizehi gppaklo gppakhi gppranglo gppranghi
[-aatten -ppos -ssprd -rseed -w]

```
texture tgrouped 5 marimba tgrouex1d ndftgr1.txt 12 0.5 1 1 40 80 0.4
1.0 48 84 0 1 1 0 0 5 10 50 100 3 10
```

We hear a rapid, fluid texture, in which the random element has become prominent.

Musical Applications

We now have a fairly good idea of what TIMED GROUPS is all about. We can try for a different range of effects by making use of harmonic fields or sets to constrain the random dimension. *Ndftgr2.txt* adds a harmonic field to a new and simpler timegrid.

```
60
#4
0.00 1 0 0 0
0.50 1 0 0 0
3.00 1 0 0 0
4.50 1 0 0 0
#7
0.0  1 48 0 0
0.0  1 53 0 0
0.0  1 58 0 0
0.0  1 63 0 0
0.0  1 68 0 0
0.0  1 73 0 0
0.0  1 78 0 0
```

TGROUPED Example 2(Preset/Patch **tgrouped2/tgrouex2**):
... minoutdur skiptime (0.5)
snd1st (1) sndlast (1) mingain (40) maxgain (70) mindur (0.25) maxdur (0.75)
minpch (48) maxpch (84)
phgrid (0) grpspace (1) gpsprange (1) amprise (0) contour (0)
gpsizelo (7) gpsizehi (28) gppaklo (75) gppakhi (150) gppranglo (1) gppranghi (7)
[-aatten -ppos -ssprd -rseed -w]

```
texture tgrouped 3 marimba tgrouex2 ndftgr2.txt 21 0.5 1 1 40 70 0.25
0.75 48 84 0 1 1 0 0 7 28 75 150 1 7
```

We hear repeated notes, arpeggios on perfect 4^{ths}, something that sounds like permutations on 2 or 3 pitches, some dyads (2-note chords), varying speeds. Let's look more closely at the various settings:

- A new time template spreads out the onsets.

- Mode **3** is used so that only the pitches defined in the harmonic set are used. Arpeggios along stacked 4^{ths} are frequent. If Mode **1** is used, it is interesting to note that the inclusion of different octaves also has the effect of enabling adjacent pitches to be used, enriching the texture, but reducing the clarity of the defined harmonic grid.

- The *skiptime* of 0.5 relates to the time template. The last time is at 4.5 (sec). Adding 0.5 to this = 5.0, so that the next sequence will begin 'on the beat' at 5 sec.

- The note duration range of 0.25 to 0.75 concerns how much of the input soundfile to allow to play, which produces shorter and longer note values, the latter allowing in a bit of resonance. Note duration ('sustain') cannot be longer than the input soundfile.

- The groupsize range moves between 7 and 28 notes per group. This will result in some groups which end well within before the next timegrid onset, and others which will still be playing out their notes when the next timegrid onset comes: i.e., overlaps will occur, in this case producing dyads. These dyads may be perfect 4^{ths}, but other intervals may occur. It all depends on the note selected for the start of the group.

- The faster and slower pacing of the notes in the groups results from the internal group packing range of 75 to 150 ms.

- The groups vary nicely between repeated notes and arpeggios because the group pitch range goes from 1 to 7 notes of the harmonic field.

End of TEXTURE TGROUPED

TEXTURE TMOTIFS/MOTIFSIN – A texture with the onsets of fully user-specified motifs constrained to a rhythmic grid and attached to pitches drawn from a pitch range or from a Harmonic Field/Set; one or more input sounds

Usage

texture tmotifs/tmotifsin mode *infile [infile2 ..] outfile notedata outdur skip*
sndfirst sndlast mingain maxgain minpich maxpich
phgrid grpspace gpsprange amprise contour multlo multhi
[**-a***atten*] [**-p***position*] [**-s***spread*] [**-r***seed*]
[**-w**] [**-d**] [**-i**]

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 [not used by TMOTIFSIN]

Parameters

infile – input soundfile to use as source material
infile2 ... – optional soundfile(s) to use as additional inputs
outfile – output soundfile
notedata – **textfile, containing:**

1. assumed MIDI 'pitch' of each input sound, specified on the 1st line (60=original pitch)
2. followed by a times NOTELIST, specified thus:
   - **#***N* (where *N* is the number of pitches in the notelist).
   - This is followed by *N* lines in the format:
     time (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *dur* (secs). Only *time* is active.
     Pitch, *amp* and *dur* are inactive fields in TEXTURE TMOTIFS.
   - different times must be given, and they must increase; all 0's for chords are not allowed.
3. Now you need an additional set of lines for the motif definition.
   - these are also introduced by the number of lines (**#***N*, which acts as a separator).
   - This is followed by *N* lines in the format:
     time (secs) *infile_no* *pitch* (MIDI) *amp* (MIDI) *dur* (secs). All fields are active to define the motif.
4. Optionally there may be a Harmonic Field/Set definition in the usual format, omitted in Mode **5**. If omitted, the motif definition occupies the second section of the note data file.

Form: **MPV/T/M/–**		Form: **MPV/T/HF-S/M**	
Mode 5 format	**Comments**	**Modes 1-4 format**	**Comments**
60	MIDI pitch value(s)	60	MIDI pitch value(s)
#*N*	No. of lines to follow	***#****N*	No. of lines to follow
time instr pitch amp dur	List of times	*time instr pitch amp dur*	List of times
#*N*	No. of lines to follow	***#****N*	No. of lines to follow
time instr pitch amp dur	List of Motifs	*time instr pitch amp dur*	List of Harmonic Field/Set
		#*N*	No. of lines to follow
		time instr pitch amp dur	List of Motifs

outdur – minimum duration of the *outfile*
skip – time between repeats of timing motif in *notedata*, i.e., between runs of the 'line' substructure notelist
sndfirst, sndlast – first and last soundfiles to use from a list of soundfiles for input (Range: 1 to the number of sounds)
mingain, maxgain – minimum and maximum level of input sounds (Range: 1 to 127; Default: 64 and 64)
minpich, maxpich – minimum and maximum pitch (MIDI note value)
phgrid – a timegrid in milliseconds applying WITHIN the motifs (Range: 0.0 to 1000.0)
gpspace – spatialisation of event-groups (Range: 0 to 5; Default: 1)

0 no change
1 scattered (Default)
2 motifs will move **towards** where the event is
3 motifs move **away from** where the event is
4 motifs follow the texture motion from Left to Right
5 motifs follow the texture motion from Right to Left

gpsprange – spatial range of event-groups (Range: 0 to 1; Default: 1)
amprise – amplitude change within motifs (Range: 0 to 127; Default: 0)
contour – amplitude contour of groups (Range: 0 to 6; Default: 0)

0 mixture of the other types (Default)
1 crescendo
2 flat (no change)
3 decrescendo
4 crescendo or flat
5 crescendo or decrescendo
6 decrescendo or flat

multlo, multhi – smallest & largest multiplier of total input duration of motif
-a*atten* – overall attenuation of the output
-p*position* – centre of sound output image (Range: 0 to 1, where 0 is Left and 1 is Right; Default: 0.5)
-s*spread* – spatial spread of texture events (Range: 0 to 1, where 1 is full spread)
-r*seed* – the same *seed* number will produce the same result on rerun (Default: 0, where 0 is different result each time)
-w – always play whole input sound, ignoring duration values

-d – motif notes all have the same duration as the timing note
-i – motif is not each confined to a fixed instrument (Default: fixed)

Understanding the TEXTURE TMOTIFS Process

*TEXTURE TMOTIFS enables us to create fully defined motifs which begin to play according to the times in a rhythmic template. These will draw their pitches either from a pitch range (Mode **5**) or from a Harmonic Field/Set (Modes **1 - 4**).*

In TGROUPED we found that the rhythmic template repeated without very much definition because we could not set the amplitude at specific time points. TMOTIFS enables us to create fully defined musical figures, thus providing a useful spectrum in the 'timed' group of functions from random to fully specified.

> **All the examples for TEXTURE TMOTIFS can be made by running *tmotiexs.bat* from the DOS prompt. The soundfiles produced can be deleted with *tmotidel.bat*.**

Our first example, then, simply creates a 6-note figure which repeats on a rhythmic template of compressing durations: 3-2-1-0.5 seconds, giving us the onset template of onsets at 0, 3, 5, 6, and 6.5 seconds. *Ndftmo1.txt* sets this template and an easily recognisable musical figure:

```
60
#5
0.0 1 0 0 0
3.0 1 0 0 0
5.0 1 0 0 0
6.0 1 0 0 0
6.5 1 0 0 0
#6
0.000 1 63 70 0.4
0.167 1 62 65 0.3
0.334 1 60 60 0.3
0.500 1 62 65 0.4
0.667 1 60 60 0.3
0.834 1 59 55 0.3
```

TMOTIFS Example 1 (Preset/Patch **tmotif1/tmotiex1**):
... outdur (21) skiptime (3.5)
snd1st (1) sndlast (1) mingain (40) maxgain (80)
minpich (48) maxpich (84)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
multlo (1) multhi (1)
[-aatten -ppos -ssprd -rseed -w -d -i]

```
texture tmotifs 5 marimba tmotiex1 ndftmo1.txt 21 3.5 1 1 40 80 48 84
0 1 1 0 0 1 1
```

We hear a dynamically shaped semi-quaver sextuplet motif get closer together and, because each motif is 1 second long, there is an overlap of ½ second at 6.5 seconds. Longer note durations in the note data file (up to the length of the input soundfile) will produce more resonance due to a greater note event overlap within the motifs.

Let's look at the time between repeats very closely, as it is musically useful to understand this accurately. The *skiptime* parameter is the time between repetitions of the rhythmic template, which *Sound Loom* refers to as the 'pause between line repeats'. **This is calculated from the *onset* of the last motif, i.e., the last of the timing template nodes.** The program does not take into account how long the motif is, so this *skiptime* pause does *not* begin at the end of the last motif, but rather at its beginning. (If it did begin at the end, this would prevent any overlapping of the last motif and the start of the next repetition of the template.)

The last motif in the Example 1 template begins at 6.5, lasts for 1 second and ends at 7.5. The 3.5 second *skiptime* pause therefore starts at 6.5 and ends at 10.0. Thus the repeat begins at 10 seconds. Suggestion: When counting this out while listening to the output soundfile, remember to start the count at 0 as in the note data file so that your count will match the numbers in the file.

Thus we could see that a *skiptime* pause of 0.5 seconds would cause the template to begin again at 7.0 seconds, overlapping the last motif by ½ second. Try it! With longer and more harmonically oriented motifs, this overlap potential can become musically significant.

The motif duration multiplier *multlo/hi* alters the speed of the motifs, but not of the timing template. A range from 0.5 (twice as fast) to 2.0 (twice as slow) introduces more variety of tempo into the output. Also note that it changes the overlaps considerably.

A second example introduces the use of a Harmonic Field/Set and seeks to play with the overlaps which may occur. *Ndftmo2.txt* creates a timegrid equally spaced at ½ sec. and revises the motif to form a 3 second arpeggio on various thirds. The harmonic set comprising pitches which belong to the motif itself leads to transpositions which create intervals of 3rd as the 3 second motifs overlap on a ½ second rhythmic template.

```
60
#5
0.0 1 0 0 0
0.5 1 0 0 0
1.0 1 0 0 0
1.5 1 0 0 0
2.0 1 0 0 0
#9
0.0 1 52 60 0
0.0 1 55 60 0
0.0 1 58 60 0
0.0 1 61 60 0
0.0 1 63 60 0
0.0 1 66 60 0
0.0 1 68 60 0
0.0 1 72 60 0
0.0 1 75 60 0
#18
0.000 1 52 70 0.4
0.167 1 55 65 0.3
0.334 1 58 60 0.3
0.500 1 55 65 0.4
0.667 1 58 60 0.3
0.834 1 61 55 0.3
1.000 1 58 70 0.4
```

```
1.167 1 61 65 0.3
1.334 1 63 60 0.3
1.500 1 61 65 0.4
1.667 1 63 60 0.3
1.834 1 66 55 0.3
2.000 1 63 70 0.4
2.167 1 66 65 0.3
2.334 1 68 60 0.3
2.500 1 66 65 0.4
2.667 1 68 60 0.3
2.834 1 72 55 0.3
```

TMOTIFS Example 2a/b (Preset/Patch **tmotif2a, &b/ tmotiex2**):
(In *Sound Loom* use **tmotiex2**, making the parameter changes described below to create the 'b' example.)
... outdur (21) skiptime (0.5)
snd1st (1) sndlast (1) mingain (40) maxgain (80)
minpich (48) maxpich (84)
phgrid (0) gpspace (1) gpsprange (1) amprise (0) contour (0)
multlo (1) multhi (1)
[-aatten -ppos -ssprd -rseed -w -d -i]

```
texture tmotifs 3 marimba tmotiex2a ndftmo2.txt 21 0.5 1 1 40 80 48
84 0 1 1 0 0 1 1
texture tmotifs 3 marimba tmotiex2b ndftmo2.txt 21 0.75 1 1 40 80 48
84 0 1 1 0 0 1 1
```

We hear the motif rising repeatedly without a break. The *skiptime* of 0.5 sec. added to the final time node of 2.0 sec means that the time pattern repeats starting at 2.5 sec. Thus the ½ second time intervals is smoothly maintained. The similarity of motif and field in harmonic design leads to the interval of a third sounding as a simultaneity as the motifs overlap. The 'b' example makes only one change: 0.25 is added to the *skiptime*. This ¼ second offset starts to produce semiquaver (16th note) offsets when the rhythmic template repeats.

As with MOTIFS and MOTIFSIN, the harmonic field or set is used in two different ways:

- MOTIFS: just the first note of the motif is placed on the harmonic grid, with all the pitches of the motif itself being accurately transposed from that reference point;
- MOTIFSIN: all the notes of the motif are constrained to the harmonic grid, warping the motif to do so.

Also see some further observations about starting off motifs on the same pitch each time.

Musical Applications

The range of possibilities in the TEXTURE set provides facilities for remarkably varied musical situations. In particular, there is a balance between random selection and fully defined musical figures. Always there are constraints within which random selections are made, such as parameter ranges and harmonic fields or sets, and all of these can vary through time. In addition you are not constrained to use the pitches of the tempered scale – fractional MIDI values will give you any tuning you desire. Thus, while it is possible to duplicate MIDI sequencer type fully defined pitch and rhythmic relationships, it is also possible to use the algorithmic potential of this software to move beyond towards freer forms of texture design, designs often very

suitable for sounds of a more 'natural' character, i.e., with limited pitch content.

For example, you may move from a dense apparently unpitched texture, (using a very small packing time and a wide pitch range, but with no harmonic field specified – Mode **5**) then gradually narrow the pitch range to (close to) a single pitch. In this way the unpitched band becomes gradually focused on a pitch (provided your source material has some pitch content). You can also take any sound of complex quality (e.g. speech) and by creating a very dense texture by using a very small packing time, 'white out' the texture, producing a band of undifferentiated noise. By varying the packing time you can then pass between the noise band and a texture of voices.

Thus the DECORATED group defines a 'line' (nodal substructure), but the decorations themselves are formed by (constrained) random selections. Complementing this is the ORNATE group, which enables you to place fully defined figures ('ornaments') on this 'line', with the option to constrain the pitches of the figure(s) to a harmonic grid.

Similarly, MOTIFS makes fully defined motifs possible, but the 'line' feature is removed. Thus the motifs will begin either on a pitch selected at random from a pitch range, or selected at random from a harmonic grid. In MOTIFS only the first pitch of the motif is 'forced' onto this grid, and in MOTIFSIN, all the pitches of the motif are forced onto the grid, warping the figure if necessary to do so.

TMOTIFS/TMOTIFSIN complements the 'line' feature of DECORATED and ORNATE with a **timing grid**, so that the motifs can be made to begin at specified times. This opens up possibilities such as controlled overlaps, variations in density, and interval control. The relationship between the timing grid, the intervals in the harmonic grid, the intervals in the motif itself, the time between the onset of the last motif on the grid and the rerun of the grid pattern (*skiptime*), and the tempo control (the motif duration multiplier *multlo/hi*) – all affect the overall result and provide a wonderfully flexible environment for designing musical textures.

Furthermore, as we are working in the context of sound transformation software, it will be normal to apply the designs to inharmonic input soundfiles with little or no clearly defined pitch content. In this case the timing and pitch contours control 'washes' of sound and timbral colour.

End of TEXTURE TMOTIFS / TMOTIFSIN

Last Updated 14 July 2015
Documentation: Archer Endrich, revised R. Fraser