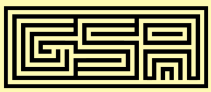
Yann Ics

Journal of
Generative Sonic Art

2014 – 2024



Foreword

About this edition

This journal is a witness to a decade of work on music with a conceptual approach. This is far from perfect, and probably, on some points, outdated, even obsolete, but I have the weakness to believe this is worth reading and sharing.

For any request, feel free to contact the author:

by.cmsc@gmail.com

The first part of this book regroups articles as chapters fitting my research interest in music analysis and synthesis in terms of algorithms.

Most of my works are interconnected in this writing as a development, a continuation, or an experiment. A distinct work also connected to this writing is the *Neuromuse3* project (N3), which aims to develop algorithms in terms of neuroscience and cognition in a 'musical' context. The documentation is available online:

https://github.com/yannics/Neuromuse3/blob/master/n3.pdf

Common Lisp libraries required and designed for this work:

→ *cl-gsa* notably mentioned in the chapters *Melody to Tone* and *Music Data Score*.

https://github.com/yannics/cl-gsa

→ *cl-cycle* mentioned in the chapters *Electronic background sketch* and *Proportional canon as an event climax*.

https://github.com/yannics/cl-cycle

The second and the third parts refer to the *praxis* as concepts and as experiments *in situ*. The SuperCollider `gsa.quark` is required.

https://github.com/yannics/GSA

For any reference to this writing, please cite in the press:
Yann Ics [2014/2024]. *Journal of Generative Sonic Art*. [online] Available at: https://github.com/yannics/GSA/blob/master/gsa.pdf.

Third-party softwares

Some third-party softwares are required for some of these works.

Praat

Praat is a free software for the analysis of speech in phonetics. It was designed, and continues to be developed, by Paul Boersma and David Weenink of the University of Amsterdam.

http://www.fon.hum.uva.nl/praat

Praat is used for the analysis of the sound file. Praat can be scripted and it can be called in a Bash script. The Praat manual displays all the needed information concerning the algorithms and the parameters used in Praat.

SBCL

SBCL means Steel Bank Common Lisp. It is a free software and a mostly-conforming implementation of the ANSI Common Lisp standard.

http://www.sbcl.org

Common Lisp is a very efficient and flexible programming language specifically and historically dedicated to Artificial Intelligence. SBCL can be called in a Bash script.

Morphologie

Morphologie is an OpenMusic and PWGL library implemented in Common Lisp. Morphologie is a set of functions of analysis, recognition, classification and reconstitution of symbolic and digital sequences, developed by Jacopo Baboni-Schilingi and Frédéric Voisin at the IRCAM in 1997.

http://www.baboni-schilingi.com/index.php/research

In this work, I adapted some functions for a 'pure' Common Lisp use.

Midi

This is a Common Lisp library for parsing MIDI (Musical Instrument Digital Interface) file format files and representing MIDI events.
http://www.doc.gold.ac.uk/isms/lisp/midi

In the chapter *Music Data Score*, section *Score*, the midi package is required as a dependency of *cl-gsa*.

FluidSynth

FluidSynth is a free software synthesizer and can be used as a command line shell.
http://www.fluidsynth.org

With the package `fluid-soundfont-gm`, `FluidSynth` allows converting midi file to a sound file – see chapter *Music Data Score*, section *Convert midi file to MDS*.

SuperCollider

SuperCollider is a programming language for real-time audio synthesis and algorithmic composition.
http://supercollider.github.io

And also,

SoX

SoX (Sound eXchange) is a cross-platform command line utility that allows manipulating audio files.
http://sox.sourceforge.net

The associated command line SoXI (Sound eXchange Information) allows the displaying of sound file metadata.

Gnuplot

Gnuplot is a portable command-line driven graphing utility as an interactive plotting program.
http://www.gnuplot.info

Graphviz

Graphviz is open source graph visualization software using graph layout programs, such as Neato – see note 3 on page 54.
http://graphviz.org

Contents

List of Algorithms

List of Figures

Part I

GENERATING AND ANALYZING DATA
FOR MUSICAL PURPOSE

Chapter 1

Documentation of the executable script *enkode*

2014 – 2021

This application is a command line and falls within a developmental process as an *alpha version* within experimental research.

1.1 Presentation

enkode is an executable script Bash and it can be used as a command line under Unix or Linux. This executable consists of the analysis of a signal defined by a sound file according to some modalities that will be described here.

This script was designed more specifically for undetermined pitch percussion music. Then, the modalities concern the analytic discrimination in terms of duration (sample segmentation), relative pitches (defined by $f0$ and the centroid), and dynamics (as loudness and low-pass filtered event loudness as 'loudbass') in such a way each segment is analyzed as an array event. Thus, *enkode* generates data on the *stdout* for structural or formal analysis and sound synthesis as parameters.

This is an experimental procedure designed to illustrate one way to extract relevant characteristics of a sonic segment.

enkode needs two main programs. The first one is Praat devoted to the analysis of sound files, and the second one is the Lisp compiler SBCL for data processing. The Shell script *enkode* ensures the mediation.

1.2 Event segmentation

1.2.1 Dynamic profile

Beforehand, we use the analysis of Praat software that has the particularity to process each analysis with a simple script. This script generates initially cochleagram analysis based on the perception of sound by the ear. This allows considering the same encoding of the inner ear to the brain. This translates into a bark scale[1] ranged in 24 critical bands of hearing and the perceived sound level expressed in the phone unit. The phonie is a weighted expression of the sound level according to an equal loudness curve (see figure 1.1), which reflects the sensitivity of the human auditory system. The equal loudness curves were empirically established in 1933 by Fletcher and Munson and revised in 1956 by Robinson and Dadson.

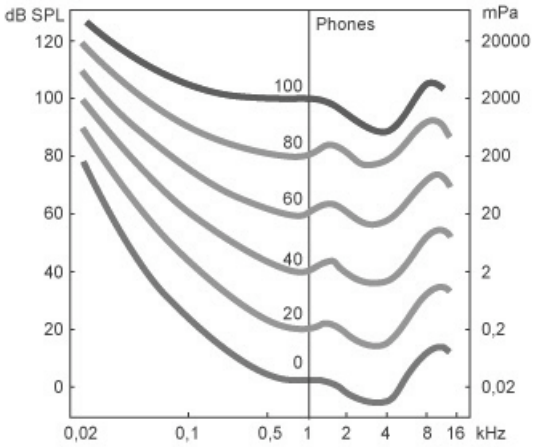


Figure 1.1: Equal loudness curves.

To complete this representation of our sound perception, the phones of analysis Praat are converted into sones[2] such as:

$$sones = 2^{\frac{phones-40}{10}}$$

Thus, by adding the loudness of each frame (designating the timbre profile) we get a relevant dynamic profile.

1.2.2 TextGrid

The TextGrid is a connected sequence of labeled intervals, with boundaries in between. Currently, these intervals correspond to an event.

[1] The Bark scale is a psycho-acoustical scale proposed by Eberhard Zwicker in 1961.
[2] The sone unit can estimate a sound twice as loud by a double value of loudness.

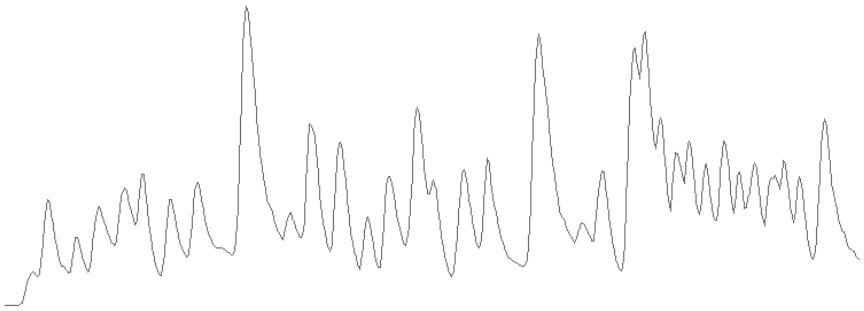


Figure 1.2: Dynamic profile of a sample computed by Praat. [→ 1.1]

From figure 1.2, we have to make a segmentation to discriminate each event inside the sample in order to write an appropriate TextGrid file that we will use for further analysis in Praat. For that, we have to select each peak and each valley from the profile and fill in the following conditions in the case where:

- If the sample starts or finishes on a peak – for example, due to an inappropriate 'cutting' – it will be ignored.

- The differential gap under a threshold loudness between consecutive peak and valley or valley and peak implies their removal. This gap can be set with `--loudness-diff-threshold` – the default value is the mean value according to the logarithmic scale of the variations peak/valley and valley/peak.

- All peak levels lower than a minimal loudness is deleted. This minimal value of loudness as peak can be set with `--loudness-min-threshold`. The default value is the minimum value of a peak after the filtering of `--loudness-diff-threshold`.

- All valley's level upper than a maximal loudness is deleted. This maximal value of loudness as valley can be set with `--loudness-max-threshold`. The default value is the maximum value of a valley after the filtering of `--loudness-min-threshold`.

- All events whose duration is less than the value of `--min-duration` – 0.05 second by default – will be merged with the next event, except for the last which will be merged in this atypical case with the previous event.

- All events whose duration is more than the value of `--max-duration` – 10 seconds by default – will be truncated to fit within this duration.

1.3 Extracting the required values

Now, with the sample and its associate TextGrid, a script Praat allows getting values for each event as duration, $f0$, centroid, loudness, and 'loudbass'.

1.3.1 Duration

The duration of each event is estimated from the Δt between two valleys of the profile as described above. The duration of the last peak is estimated from the last valley in relation to the total duration of the sample.

1.3.2 *f0*

From each segment, we deduct the timbre profile by smoothing spectrum by the cepstrum method[3].

The value of $F0$ in Praat is estimated to 500 Hz and can be set with the option `--smooth-frequency`.

Then, only the first peak or partial from the smoothed profile is retained as $f0$.

1.3.3 Centroid

The relative pitch is estimated – currently in the case of an inharmonic sound – by the centroid of the timbre profile as spectrum[4] for a given sound event expressed in Hertz.

The spectral centroid represents the frequency center of gravity of a signal. The center of gravity is the average of f over the entire frequency domain, weighted by the power spectrum.

The centroid is defined as follows:

$$f_c = \frac{\int_0^\infty f P(f) df}{\int_0^\infty P(f) df}$$

with $P(f)$ as the power spectrum.

[3] **Smoothing spectrum by the cepstrum method**.
The signal can be viewed as a superposition of a short wave vibration with the 'period' of $F0$ and long wave vibrations representing the course of the transmission function.
Then, smoothing means low-pass filtering the signal in such a way that the present short-wave vibration with the 'period' of $F0$ is removed and the envelope remains. If $F0$ is known, a band-stop filter can be used instead of the low-pass, i.e., a filter that blocks only the undesired oscillations and their immediate vicinity, but lets everything else pass.

[4] The spectrum is a representation of a sound as either power or pressure as a function of frequency.

1.3.4 Loudness

The loudness of a sound event is expressed in sone unit.

Knowing that the loudness is defined for each frame such as:

$$L = \int 2^{(e(f)-40)} df$$

with $e(f)$ as the excitation in phon unit.

I state that for a continuous sound, the 'amplitude' feeling decreases over time by habituation. The decreasing function depends on the context and the duration of the sound object. The first is not quantifiable and the second is moreover depending on the shape of the sound object. Therefore, intuitively and empirically as an experimental evaluation, the mean loudness of a given sound object is estimated as follows:

$$\overline{L} = \frac{\sum_{i=1}^{n}(n-i+1)L_{(i-1)dt}}{\sum_{i=1}^{n} i}$$

1.3.5 Bass loudness

In order to isolate the salience of bass frequencies as presence, we have to realize a filtering type low pass filter (figure 1.3) and then get the loudness in the same way as seen previously with the loudness.

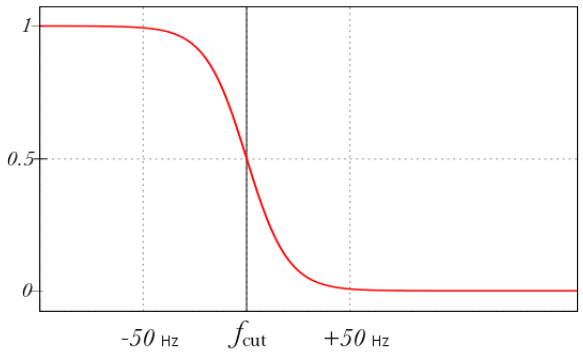


Figure 1.3: Low Pass Filter showing the width (fixed to +/- 50 Hz) of the region between pass and stop according to the cut-off frequency.

The cut-off frequency can be set with the option `--cutoff-frequency` (default value is 100 Hz).

1.4 Discrimination in classes

The values generated by the previous analysis are destined to be discriminated into n classes in order to get a numerical score, mainly for structural analysis and as synthesis parameters. This is done by a recursive discrimination based on the mean of the overall values.

Then, with the option `-I`, `--as-int`, the result is a data list of positive integers with by line an event, and by column respectively the class number of duration, $f0$, centroid, loudness and 'loudbass'. Also, the result can be converted as a thrifty code[5] with the option `-T`, `--as-tc`, or as a gray code[6] with the option `-G`, `--as-gc`.

All these options take as argument a positive number which is applied as a number of recursivity for all different parameters, or a list of positive numbers as the number of recursivity with a cardinal equal to 5, that is to say, one number by values of events. Each class as a positive integer means the value of an attractor. This ordered 'alist' is written on the 5 first lines of *<fileName>.info*.

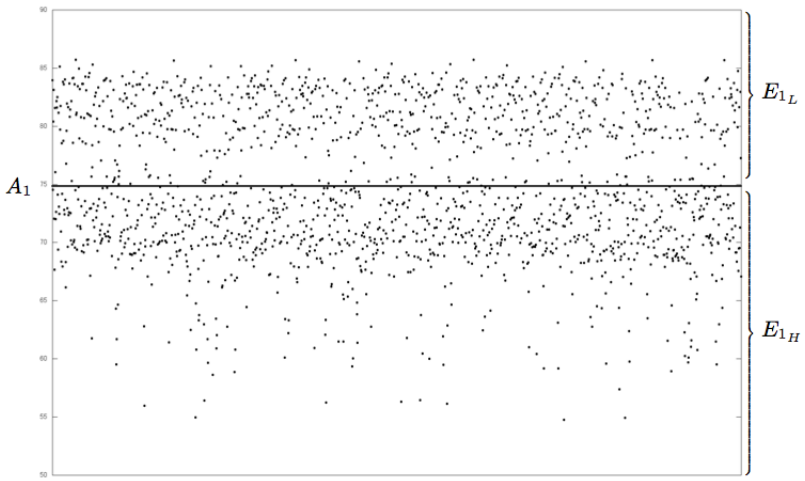


Figure 1.4: [Coefficient of recursivity: 1] In this example is represented by a cluster of bass loudness. The attractor A_1 is the arithmetic mean of all points, creating in this way two subsets: E_{1_L} and E_{1_H}.

[5]The thrifty code consists of writing one piece of information on one bit of n digits. This involves discrimination of the order of $card(A_n) = n$.

[6]Gray code – according to the Bell Labs researcher Frank Gray who introduced the term of *reflected binary code* in 1947 – is an ordering of the binary numeral system such that two successive values differ in only one bit.

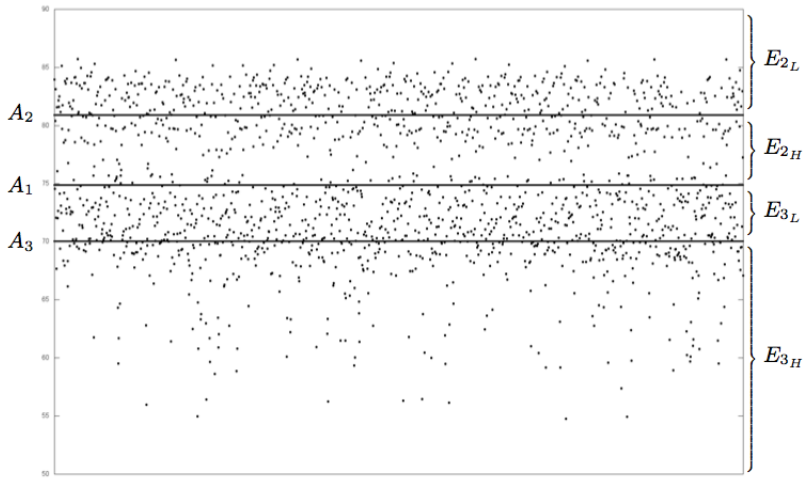


Figure 1.5: [Coefficient of recursivity: 2] From the two subsets of figure 1.4 (E_{1_L} and E_{1_H}), we determine in the same manner two new attractors (respectively A_2 and A_3) generating in this way two subsets for each attractor, respectively E_{2_L}, E_{2_H}, E_{3_L}, and E_{3_H}.

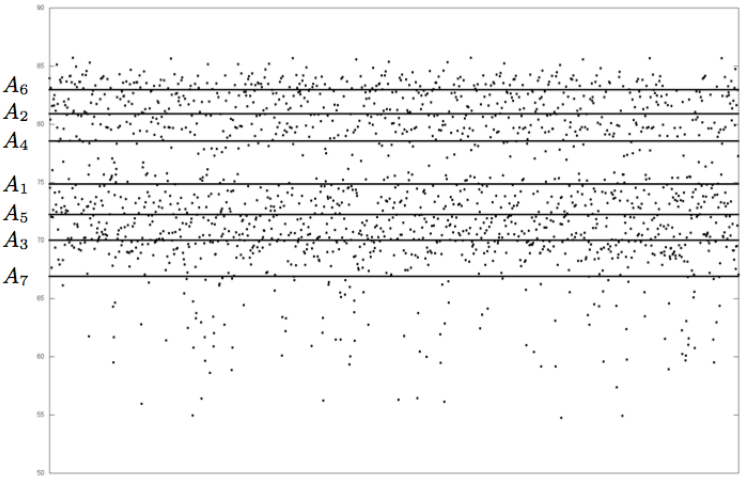


Figure 1.6: [Coefficient of recursivity: 3] Just repeat the process described above (see Figures 1.4 and 1.5) to obtain 4 new attractors (A_4, A_5, A_6 and A_7) calculated from subsets of Figure 1.5. This gives us a total of 7 attractors (or classes).

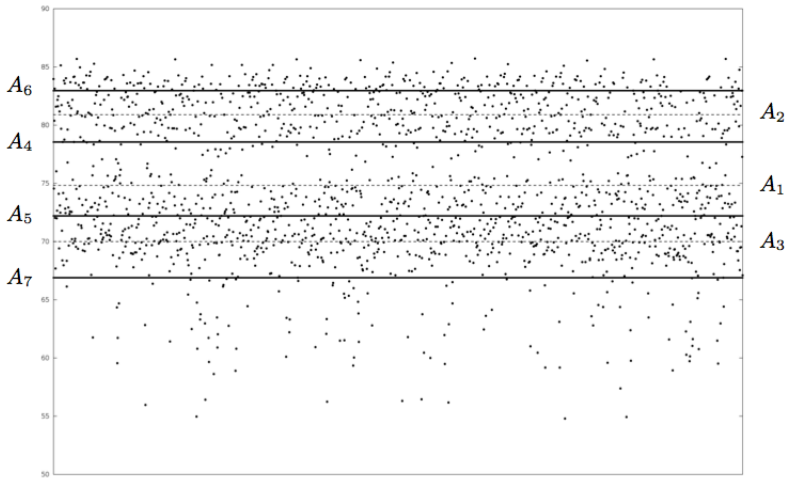


Figure 1.7: When the argument is a float number – for example with a value of 2.5 – the number of attractors is equal to a number of attractors with a coefficient of recursivity of 3 (ceiling value of 2.5) – see figure 1.6 – minus the number of attractors with a coefficient of recursivity of 2 (floor value of 2.5) – see figure 1.6. Thus, this gives us the 4 following attractors: A_4, A_5, A_6, and A_7.

This way of discriminating data consists of determining the number of classes relative to attractors. These attractors are computed by the arithmetic mean of the data in terms of the set according to a coefficient of recursivity determining the number of attractors. When the argument n is an integer, the number of attractors is equal to $2^n - 1$, and when the argument n is a float number, the number of attractors is equal to $2^{\lfloor n \rfloor}$.

See the description of this algorithm in figures 1.4 to 1.7.

Note that the recursivity stops when all data are captured by an attractor. Then, the number of discriminations can be smaller than the number of expected attractors defined by the value of the argument.

1.5 Command line use

1.5.1 Install *enkode*

- Create a personal bin directory – for instance `$HOME/bin`

- Copy *enkode* in this folder.
- Add the following to file ∼/.profile
  `export PATH=$HOME/bin:$PATH`
- To install man page:

```
$ sudo mkdir /usr/local/share/man/man1
$ sudo cp enkode.1 /usr/local/share/man/man1/
# plus on LINUX
$ sudo mandb
```

1.5.2 Using *enkode*

- Preliminary analysis

```
$ enkode -p test.wav
NumberOfEvents              119
--loudness-min-thres        14.955614       < 18.768398
--loudness-max-thres        46.377018       > 39.519974
--loudness-diff-thres       3.8155203       < 3.8345852 >
    3.7744398
MinDiffLoudness             0.018797874
MaxDiffLoudness             48.29043
```

This table allows us to adjust the different threshold values according to their initial value and their next efficient value. Most of the time, these values have to be adjusted empirically according to the accuracy required.

Sometimes, the discriminative algorithm is not accurate or efficient enough. In this case, the segmentation can be edited or created in Praat. Then, it is recommended to 'reframe' the TextGrid, that is to say, labeled the first tier as an ordered integer. This can be done with the script `enkode.praat` – see section 1.7 on page 26.

- Default behavior:

```
$ enkode test.wav
0.28 228.790283203125 235.7666500379242
    8.830720512009613 1.7966619273740976
0.10999999999999999 193.798828125 165.9317394706834
    5.758170441172713 1.9914783596510206
0.15999999999999998 506.0302734375 492.1692215465892
    10.396071452511483 1.6476902981421457
0.12 349.91455078125 374.6610256304264
    10.416102008324566 1.6657044245205024
0.14999996000000004 204.5654296875 206.9951008846426
    9.59662211074838 1.9097877289745604
...
```

- Using some options:

```
$ enkode --as-gc=3 test.wav
1 0 1 0 0 1 0 0 1 0 0 1 1 1 1
0 0 1 0 0 1 0 0 1 0 0 1 1 0 0
0 1 0 1 0 0 1 1 1 0 1 1 0 0 1
0 1 1 1 1 0 0 1 0 0 1 1 0 1 1
0 1 0 0 0 1 0 0 1 0 1 1 1 0 1
...
$ cd ~/Documents/enkode/test/ | ls
test.info
test.profile
test.raw
test.TextGrid
$ head -5 test.info
0.10512812 0.13341777 0.16100016 0.18058825 0.25173908
    0.2737499 0.3035293
230.04222 261.6105 304.03033 336.9426 406.43915
    465.34964 542.8634
240.16635 299.2008 349.80164 402.11923 474.9509 526.0028
     585.22284
8.017678 10.201452 12.064082 14.078046 16.145906
    18.439219 20.902403
1.6448121 1.6699406 1.7021359 1.7619185 1.8279897
    1.9078826 2.0029936
```

- Error behavior:

```
$ enkode -I '(2 3.5 6)' test.wav
... error during process, check error in ~/Documents/
    enkode/error.log ...
```

1.6 Download

August 30, 2024

enkode 7.1 alpha released.

https://github.com/yannics/enkode

1.7 *Post-scriptum*

The script `enkode.praat` allows to work directly on Praat with the TextGrid of the user. The script does the same as *enkode* except the *Event segmentation* and the *Discrimination in classes*, as well as the 'roughness' analysis (see next section) requiring lisp computation. Optionally, the TextGrid can be 'reframed'.

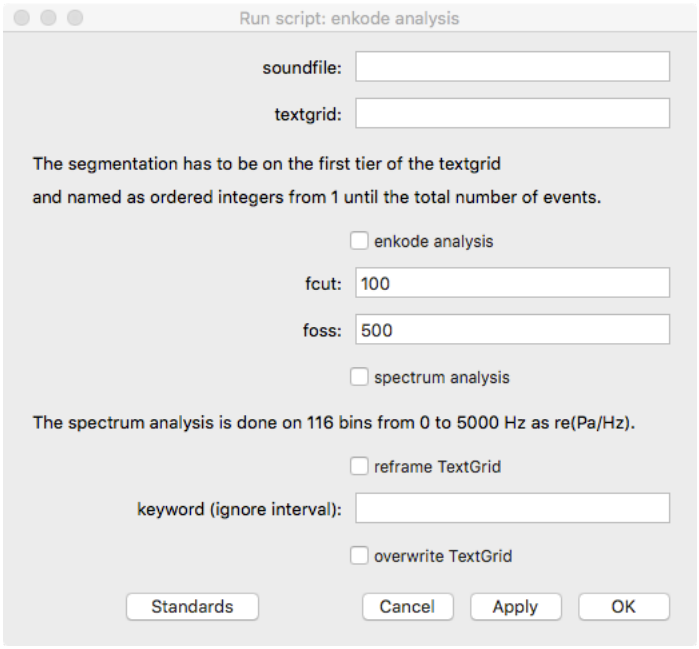


Figure 1.8: The ScriptEditor window of the script `enkode.praat`.

1.8 *Addendum*

Roughness

An additional option aims to estimate some kind of roughness. Note that is comprehensively empirical and does not take into account the psycho-acoustic phenomenon, but allows certain relevance in certain situations for sound discrimination for instance. This consists of estimating an emergent frequency from the peaks of the loudness analysis. An accurate time step is required and set at 0.001s by default, that is to say, a factor of 0.1 of the analysis time step of *enkode*. The result is raw data and has to be interpreted knowing that the roughness is identified as such and as a subjective perception in the range of 15 Hz (below this value, we are talking about *vibrato* or beat) to 300 Hz (beyond that the effect disappears) with a maximum at 70 Hz[7]. Each value is evaluated as the average of the first derivative of the peak curve and correlated with reliability as the mean value of the standard deviation of the first derivative, which is normalized in order to subtract this value to one as a percentage of confidence.

[7] This can be interpreted as follow :
$roughness = e^{-0.00355(midinote-37.175)^2}$
with $midinote = 12 \cdot log_2(frequency \times 0.00227272727) + 69$.
To know more about roughness in psychoacoustic, see appendice page 169.

Chapter 2

Melody to Tone

2015 – 2019

2.1 Presentation

This article aims to focus on some tools to interpret a melody to a unique tone with its own identity. The main idea is to analyze an audio sample or a spectrum dataset (generated for instance with the command line *enkode* with the option `--spectrum`), with or without the correlated score in order to generate a data profile in terms of 'sorting melody' and energy destined to be used as synthesis parameters.

These tools are available in the Common Lisp library named *cl-gsa*.

2.2 Analysis

2.2.1 Segmentation

Segmentation can be required to extract the harmonic profile or the spectrum of an event of a given sound file. Then the main idea consists of segmenting the audio file in order to get as many audio files as axiomatic events (notes, chords, or others) forming the melody for analysis. In practice, it exists some algorithms to realize it, but all use a specific segmentation with a teleological aim, and the relevance is relative. So, each melody will be segmented in an empirical manner with its own tools.

For instance, the command line *enkode* can realize a segmentation as a TextGrid – usable with the software Praat – according to a significative differential loudness and some threshold values defined by the value of the options:

- `--loudness-diff-threshold`
- `--loudness-min-threshold`

- `--loudness-max-threshold`

2.2.2 Harmonic profile

The harmonic profile is an ordered list of weights according to the harmonic series. Then the first item is the weight of the root as $f0$, the second item the weight of the first harmonic as $f1$, and so on.

This can be done on one specific sample according to the spectrum analysis of the software Praat and scripted as follows:

```
form Spectrum analysis
    sentence soundfile ...
    positive range ...
endform
Read from file... 'soundfile$'
current_sound$ = selected$ ("Sound")
filedelete 'defaultDirectory$'/'current_sound$'.spectrum
filedelete 'defaultDirectory$'/'current_sound$'.bw
To Spectrum: "yes"
step=1
repeat
    res = Get real value in bin: 'step'
    fileappend 'defaultDirectory$'/'current_sound$'.spectrum
         'res' 'newline$'
    freq=Get frequency from bin number: 'step'
    step=step+1
until 'freq' > 'range'
bw=Get bin width
fileappend 'defaultDirectory$'/'current_sound$'.bw 'bw'
select all
Remove
```

This script generates the *spectrum* file of the sample and the bandwidth value on the *bw* file. Then, the computation of the harmonic profile is done as follows:

```
CL-USER> (require 'cl-gsa)
CL-USER> (in-package :cl-gsa)
CL-GSA> (harm-profile (mapcar #'car (read-file
    "~/piano.spectrum")) 13 (caar (read-file "~/piano.bw")))
Fmax = 984.375 Hz.
Fundamental = 984.375 Hz.
(0.005333674 7.5771334e-4 0.0011644672 1.3338796e-4
    7.3244237e-6 7.2097446e-6 2.3828345e-6 4.374403e-6
    4.064216e-6 1.18368014e-4 4.6414575e-6 5.292382e-6
    2.4746847e-5)
```

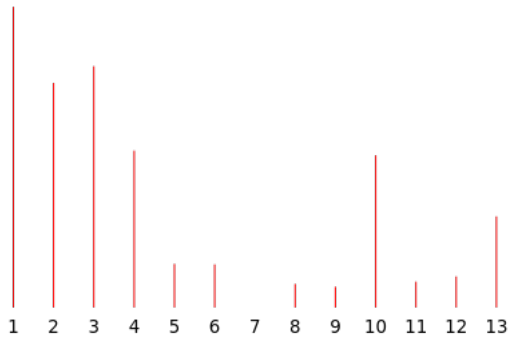


Figure 2.1: Harmonic profile of the piano from a given sample. Note that the graphic representations use a logarithm scale on the y-axis. [→ 1.2]

2.2.3 Sorting melody

as harmonic

The principle is to prioritize the notes of a given melody – expressed in midi or hertz – according to their respective harmonics profile as weight list(s) with an approximation set by the nearest division of the whole tone. This is done independently of the melody itself and this depends rather on the number of occurrences of the notes.

The result gives an ordered list of notes according to their importance – expressed as weight – in terms of resonance generated by the interaction of their respective harmonics profiles according to a deliberate approximation.

as spectrum

This can be done also from a spectrum analysis. In this case, the analysis can be done according to the harmonic series fitting the range of the spectrum, or on the whole spectrum by adding all events values by bin. The result is an ordered bandwidth index as the root harmonics or as the peaks – or partials – of the spectral profiles summation.

Here is a short description of the procedure according to the harmonic series:

- First, according to the range of the spectrum from 0 to `*f-range*` (5000 Hz by default) and the number of bins `*f-bin*` (116 by default) as bandwidth defined by `*f-range*` divided by `*f-bin*`, we list all the possible 'harmonic' series by bin (done with the function `all-ser`).

- Then, we collect the greater value defined by the summation of the value of each bin width involved divided by the number of bin widths involved for each event (done with the function `mean-sum`).

- Then, the summation of all harmonic series selected, by column (in other words by bin) is done with the function `summation-hors-tps`. The result is the weighted 'sorting melody' according to the indices of the frequency bandwidth.

Thus, the melodic profile of the sample is estimated according to the sorting melody as indices of frequency bandwidth.

The spectrum analysis can be done for each event of a sound file with the command line *enkode* according to the values defined by the options `-spectrum` and `--smooth-frequency`.

2.2.4 Energy profile

Now it exists a powerful tool called `energy-prof-morph-analysis` – renamed `energy-profile` in *cl-gsa* – from the library Morphologie developed at the IRCAM [Voisin et al., 1999]. The function `energy-prof-morph-analysis` is applied on the result of the function `new-old-analysis` which is described in the article *Morphological Analysis* [Aralla, 2002].

However, to resume and for the record, let's illustrate these algorithmic processes with the following sorting melody as the symbolic list `(a b c b d e f)` for instance.

The first step is to realize the analysis of contrast described in Paolo Aralla's article [op. cit.].

> *The analysis of contrasts, which is the function at the heart of the Morphologie Library developed by Jacopo Baboni Schilingi and Frederic Voisin, identifies the occurrences within any sequence of events.*
> *Such analysis is of quantitative type, and has considerable development potentialities towards a qualitative description of the processes that put in relation morphologic structure of the message, mnemonic perceptive activity, and psychic response.*
> *The hierarchies that the analysis of contrasts describes become qualitatively pertinent to the mnemonic activity.*
> *We have called New/Old Analysis the function that describes the newness level of an event in relation to the context in which it appears.*
> *The importance of such a function is crucial because it describes from the point of view of the psychic response the different newness level of the single event in the time.*
> *The steps to define New/Old Analysis are three:*
> *1 - Measurement of the distances:*

it allows quantifying the local distance between the different events in relation to their first appearance in the time.

```
(defun distances (sequence)
  (mapcar #' (lambda (x) (x->dx x)) (Contrasts-all-lev sequence)))
```

> *The Analysis of the Contrasts, formulated by Hervé Rivière and Frédéric Voisin, and implemented in the OpenMusic Morphologie Library, is a model able to describe the becoming of the form in the time.*
> *It points out the hierarchic relation created by the temporal sequence of the events: in fact, for the mnemonic activity, each event is a datum point for every following event and a datum point for the previous ones.*
> *The numerical transcription carried out through the Analysis of Contrasts describes the entry order of the events in the time.*
> *We could define the numerical transcription created using the analysis of contrasts as a morphological structure of the entry order of the events.*
> *From this starting point it is possible to identify the presence of internal patterns and analyze their potential capacity to describe and re-establish the form in its original status.*
> *example:*
> `(contrasts-all-lev '(a d f g f)) ----> ((1 2 3 4 3) (1 2 3 2) (1 2 1) (1 2))`
> (Documentation of `contrasts-all-lev`)

2. Attribution of different weights to the datum points:
This step is crucial because it strengthens the global hierarchy among the various analysis levels in relation to the time parameter.

```
(defun weights (sequence)
  (mapcar #' (lambda (x) (apply '+ x))
    (Contrasts-all-lev sequence)))
```

3. Application of weights to the distances:
this further step is just the application of different weights - obtained considering every time one of the events a datum point (global parameter, ex. nr. 3) - to the distances between the various contiguous events (local parameter).

```
(defun Contrasts-lev.1*weights (sequence)
  (mapcar #' (lambda (x y) (om* y x))
    (distances sequence) (weights sequence)))
;--------
(defun Contrasts-all-lev*weights (sequence)
  (reverse (mapcar #' (lambda (xx) (apply '+ xx))
    (mat-trans (mapcar #' (lambda (x) (reverse x)) (Contrasts-lev.1 *
      weights sequence))))))
```

> *A theoretical problem we have faced is the relation between the object we have analyzed and the previous and following events. Any events chain perceived as belonging to a whole and complete organism stays anyway in relation with the previous and following sequential chain.*
>
> *In the case of the performance of a music piece, the silence acts as a frame of the structure, and, being a frame, it becomes an organic element of the structure analyzed.*
>
> *It is worth underlining that even in case of extrapolation, like in the here quoted examples (a thematic fragment, a subject of a fugue, etc.), the object is perceived as a unit, and therefore the silence places it in a well defined mental space.*
>
> (Documentation of `new-old-analysis`)

start = symbol-silence-start
stop = symbol-silence-end

start	a	b	c	b	d	e	f	*stop*	
1	2	3	4	3	5	6	7	8	**39**
	1	2	3	2	4	5	6	7	**30**
		1	2	1	3	4	5	6	**22**
			1	2	3	4	5	6	**21**
				1	2	3	4	5	**15**
					1	2	3	4	**10**
						1	2	3	**6**
							1	2	**3**

Then we get all dx by row which will be multiplied by the previous summation.

1	1	1	-1	2	1	1	1	**39**
	1	1	-1	2	1	1	1	**30**
		1	-1	2	1	1	1	**22**
			1	1	1	1	1	**21**
				1	1	1	1	**15**
					1	1	1	**10**
						1	1	**6**
							1	**3**

Then we make the sum of each column.

39	39	39	-39	78	39	39	39
	30	30	-30	60	30	30	30
		22	-22	22	22	22	22
			21	21	21	21	21
				15	15	15	15
					10	10	10
						6	6
							3
39	**69**	**91**	**-70**	**218**	**137**	**143**	**146**

This 'temporary' result `(39 69 91 -70 218 137 143 146)` corresponds to the analysis of contrasts called `new-old-analysis` in the Morphologie library.

> *The step that allows transforming the new-old-analysis function into a model able to simulate the psychic response of the perceptive act to the morphologic structure occurs using three functions.*
> *Then, to this, the three functions apply to allow to define the energy profile.*
> *1. In the first passage, the transformation into absolute abs value contains all the relations with reference to the first element of the chain.*
> *At this point, the data do not represent the aging degree of the events anymore, but they are mere distance (it does not matter if they are old or new, they are to be intended nearly as physical distance between the various data stored in space/memory) related to a virtual point zero (a kind of possible present)*
> *2. In the second passage, the use of the local derivative, implemented in OpenMusic under the name of x–¿dx, the contiguous relations are again pointed out, and the distance identified in the first passage is assimilated to the energy needed to cover the contiguous distances in space/memory*
> *3. Finally, the transformation into absolute abs value, because of the transformation of the distances into energy, brings all the data back to positive values.*
> (Documentation of `energy-prof-morph-analysis`)

	0	39	69	91	-70	218	137	143
abs	0	39	69	91	70	218	137	143
$x \rightarrow dx$	39	30	22	-21	148	-81	6	
abs	**39**	**30**	**22**	**21**	**148**	**81**	**6**	

Note that this kind of analysis is strictly symbolic, focusing on the structure of contrast, and all symbols of the list are initially only referred to themselves. In our case, symbols can refer to the pitches, intervals, and durations, along with others.

2.3 Instantiation

The library *cl-gsa* allows three types of data: `:midi` as a list of midi notes or as a list of chords as a list of midi notes; `:freq` as a list of frequencies or as a list of clusters as a list of frequencies; `:spectrum` as a list of spectrum profiles defined as events.

The sorting melody and the energy profile can be used independently according to the modalities of the user. The main function `m2tab` allows to compute both at the same time and optionally offers the possibility to write the result in a file defined by its path.

2.3.1 From a partition

For instance, according to the partition of figure 2.2, the sorting melody and the energy profile are done according to the following steps:

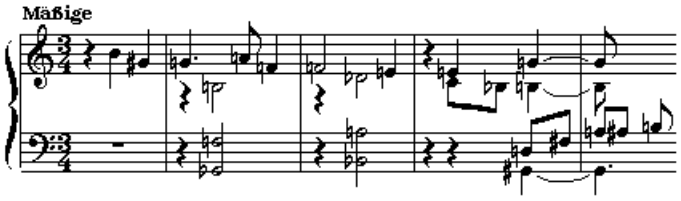


Figure 2.2: Arnold Schoenberg: opening phrase of *Drei Klavierstucke* for solo piano, opus 11.

1. Convert the partition to a midi score according to the following syntax:

```
CL-GSA> (defparameter *score* '((71) (68) (67) (59 53
    42) (69) (65) (65) (61 57 46) (64) (60 64) (58) (67
    59 50 42) (54) (57) (58) (59)))
```

2. The harmonic profile is given by the computation done in figure 2.1.

```
CL-GSA> (defparameter *harmonic-profile-piano*
    '(0.005333674  7.5771334e-4  0.0011644672 1.3338796
    e-4 7.3244237e-6  7.2097446e-6  2.3828345e-6
    4.374403e-6 4.064216e-6  1.18368014e-4  4.6414575e-6
      5.292382e-6 2.4746847e-5))
```

3. Then, the main function `m2tab` displays the result with by line the frequency of the note involved in the initial melody, the weight of the sorting melody (or in other words the energy of the signal), the weight of the energy profile as the 'formal' energy and the midi note. The last argument allows to write the result according to a given path name.

```
CL-GSA> (m2tab (sort-melody :midi *score* :harm-weight
   *harmonic-profile-piano* :approx 8) "~/test")
246.94165   0.11377126      0.087647595     59.0
349.22824   0.0894084       0.0849228       65.0
92.498604   0.08421194      0.014191644     42.0
233.08186   0.08116744      0.20640327      58.0
220.0       0.07584624      0.09196185      57.0
...
```

The approximation concerns the accuracy of the harmonic match.

2.3.2 From a sound file

It is possible to work directly from a sound file. The command line *enkode* with the option `--spectrum` allows for segmenting the sample as spectrum events list, which can be used either as harmonic or as spectrum in order to compute the sorting melody and the energy profile.

1. Segment the sound file with *enkode*:

```
$ enkode --spectrum --loudness-diff-thres=1.48 test.mp3
    > test.spectrum
```

2. Then, the main function `m2tab` displays the result with by line the top frequency as the indices multiplied by the bandwidth value (5000/116 in this case), the 'presence' as the energy of the signal, the 'formal' energy and the index of the salient frequency bandwidth.

```
CL-GSA> (m2tab (sort-melody :spectrum (read-file
   "~/test.spectrum") :partial 3) "~/test")
387.93103   0.4931115       0.022011383     8
344.82758   0.20561041      0.020645158     7
431.0345    0.11075174      0.05981024      9
1508.6206   0.03852629      0.03248577      34
1594.8275   0.021509826     0.039165083     36
...
```

2.4 Synthesis

The synthesizer I chose reproduces the 'sustain' of a piano. I scaled the level from 0.01 to 0.1 and the energy (that I transposed in terms of 'grain' that is to say a kind of 'texturation') from 300 to 3000 Hz. The synthesis is realized with the software SuperCollider.

```
// inspired by the synthetic piano patch (James McCartney)
    originally for SC2, 1998.
(
SynthDef(\op11M2T, { |bus=0, pitch=60, amp=0.1, grain=3000|
        var detune, delayTime, noise, out;
        out = Mix.ar(Array.fill(3, { arg i;
                // detune strings, calculate delay time :
                detune = #[-0.05, 0, 0.04].at(i);
                delayTime = 1 / (pitch + detune).midicps;
                // each string gets own exciter :
                noise = LFNoise2.ar(grain, 0.1); // grain =
                    3000 Hz is the reference
                CombL.ar(noise,
                // used as a string resonator
                        delayTime,
                        // max delay time
                        delayTime,
                        // actual delay time
                        6
                        // decay time of string
                        )
        }));
        Out.ar(bus, Splay.ar(out*amp)*EnvGen.ar(Env.linen
            (0.5,10,3.0),doneAction:2))}).add
)

~data = FileReader.read("~/test.m2t".standardizePath).
    collect({|line| line.collect({|val| val.asFloat})})

(instrument: \resPiano, pitch: ~data.flop[0].cpsmidi, amp:
   ~data.flop[1].normalize(0.01, 0.1), grain:
   ~data.flop[2].normalize(300, 3000)).play;
```

2.5 Discussion

I exposed the basics of some *cl-gsa* tools in the context of 'converting' a melody to a unique tone (M2T) as timbre. And of course, the possibilities of interpretation are infinite, but these tools allow managing this kind of transposition – that is to say the 'transformation' of a melody to a unique tone – with relevance and open a substantial field of creativity.

In any case, this process is destructive in a way that it is not possible to retrieve the original melody from the tone profile. In other words, a melody has one tone profile according to one process, and a tone profile with the same process can be generated with different melodies.

Chapter 3

Music Data Score

2016 – 2021

3.1 Purpose

This work allows creating a bridge between data as a musical object or as a traditional score notation, and some algorithmic environment in terms of interpretation, analysis, and synthesis such as neural network *Neuromuse3* (N3) and SuperCollider (SC). Then, this takes the form of a text file, which is interpreted as a data list, referring to the settings defining each sonic object. This text file can be read for instance as a learning dataset in the N3 context or as a SC array in the context of *Streams, Patterns, and Events* and *Sequencing with Routines and Tasks*.

The aims of this article are a proposition of formatting what I call a Music Data Score (MDS) from a partition, analysis data, and midi file – using the Common Lisp library *cl-gsa* –, and a guideline of how to use it in the N3 and SC context.

3.2 Writing Music Data Score

The main idea is to list in time a set of parameters as an event with at least duration as the first argument, and according to a specific encoding as raw data or class number relating to any kind of object. In any case, the number of parameters depends on the initial accuracy, and on the teleological object.

The last line is dedicated to the number of grouping as a number of parameters defining an event. It can be added on the same line – according to the user's needs – for instance the tempo, the number of duration relative to the tempo, the structure as the number of repetitions, or according to the order number of the phrases and so on.

3.2.1 Traditional score

In this case, we group the data as a musical phrase or by measure or by voices with at least the duration and the pitch. It is possible to add any more information such as the level for instance, or any kind of data describing the event. Note that the number of dimensions is specified on the last line.

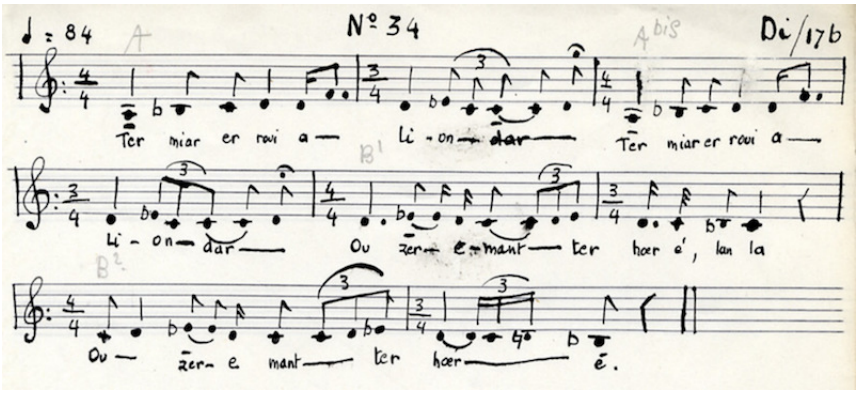


Figure 3.1: Partition of the Breton song *Tèr merh er roué a Liandar* [Marcel-Dubois et al., 1939].

From a previous work called `HEXO` (see 8.2 on page 93), the original score in figure 3.1 is encoded according to each bar of the song on 2 lines as follows:

```
24 12 12 24 6 18
57 58 60 62 62 65
24 8 8 20 12+rrand(3,6)
62 63 60 60 62
24 12 12 24 6 18
57 58 60 62 62 65
24 8 8 20 12+rrand(3,6)
62 63 60 60 62
36 18 6 20 8 8
62 63 62 60 62 63
9 3 12 24 24
62 60 58 60 0
12 24 24 6 12 8 8 8
60 62 63 62 60 60 62 63
28 4 4 12 24
62 60 59 58 0
2 84 24 9
```

- The first line represents the durations as integers according to an irreducible minimum value applied for the whole score.

- The second line represents the midi notes; the zero represents silence; the negative midi note represents a tie to the previous one as an absolute value.
- Some 'extra' text can be added depending on the context as for instance in the present case a fermata in the SuperCollider context – coded `n+rrand(a,b)` – which is evaluated between `n+a` and `n+b`.

The last line indicates for the first argument the dimension of the events – with the command line *enkode* (see chapter 1 on page 17) this number is 5 – and optionally the tempo, the number of the irreducible minimum value to fit the whole note of the tempo, and the structure of the piece (can be for instance a number of repetition – 9 in the example of the Breton song – for the whole score or a structure such as ABBAC).

3.2.2 Analysis data

The data file has to be ordered in order to list all respective parameters for each event as a row or as a column.

Note that a matrix transposition can be required with the instance method `flop` in SuperCollider context – or with the Bash function *Transpose* in appendice 2 on page 143.

From a sound file, the command line *enkode* according to an automatic segmentation, generate a list of analysis data by event as duration, $f0$, centroid, loudness and 'loudbass'. According to the option, the result is the raw data or a classification by classes (see 1.4 on page 22).

3.2.3 Reading MDS in SC

```
// Read file as array
~score = FileReader.readInterpret("".resolveRelative +/+
    "dat.score", true, true);
 // -------------------------
~infoLine = ~score.pop;
~grouping = ~infoLine[0];
~score = ~score.clump(~grouping);
// concatenate all `tracks' as musical phrases
~score=~score.collect({|a| a.flop}).flatten(1).flop;
 // -------------------------

// From the cmd line enkode:
// ~score[0] ---> duration
// ~score[1] ---> f0
// ~score[2] ---> centroid
// ~score[3] ---> loudness
// ~score[4] ---> loudbass
```

3.3 Convert midi file to MDS

For any formatted midi file with the extension .mid or .midi there are 2 ways to realize the midi conversion to MDS according to the interpretation it is possible to do either as a sound file or as a data file.

1. The conversion midi to a sound file formatted as WAV for instance – in order to compute the MDS as described previously with the command line *enkode* – can be done with the command line `fluidsynth` which requires the SoundFont file `FluidR3_GM.sf2` from the package `fluid-soundfont-gm`.

```
fluidsynth -F outfile.wav /usr/share/sounds/sf2/
    FluidR3_GM.sf2 infile.mid
```

2. The conversion midi to MDS can be done as raw data – see *Outline of the Standard MIDI File Structure* in appendix 4 on page 161 – using tools of the Common Lisp library *cl-gsa* which is described in the two next sections *Duration* and *Score*. Note that this library requires in this case the package MIDI.

3.3.1 Duration

For each track of the midi file.

In time

The duration in time is the difference between the value of the `Note_on` and the value of the `Note_off` or the same value `Note_on` with the level set to zero.

In the case of the multi-voices part, there is a bias in terms of fitting the duration between two notes or two chords. If the duration is too long, then it is clipped at the beginning of the next event. Needless to say, this bias can be avoided if the score is written with only one voice by track. If the duration stops before the next event and if the remaining duration is significant[1], a silence is set in between. Note that a silence is set to zero as a midi value.

This is done as follows:
Let t be the position in time of a *chord* as a group of notes and d the duration.
Let A_i be the dataset $[t_i, d_i, chord]$ at the position i.
Assume that $\delta_i = t_{i+1} - t_i$ and $\Delta_i = \delta_i - d_i$.
Then,
if $\Delta_i = 0 \rightarrow A_i$
if $\Delta_i > 0 \rightarrow A_i + [t_i + d_i, \delta_i - d_i, rest]$
if $\Delta_i < 0 \rightarrow [t_i, \delta_i, chord]$

[1] This is another bias about the remaining silence. The latest does not have to be under the value of the sixty-fourth, note by default according to the division value of the midi score.

Out time

All the durations computed just below are reduced in order to get the minimal value as an integer. This is done by calculating the greatest common divisor using the method of prime factorizations.

3.3.2 Score

In order to illustrate how to manage the midi conversion to the MDS, here are some code snippets in the context of N3 and SC using the score Kjølhiea from the performances `K540` (see 8.3 on page 95) which was exported as a midi file.

In this case, the scope of the possible division according to the time division of the midi file is set to `(1 2 4 8)`, that is to say respectively the quarter note, the eighth note, the sixteenth note, and the thirty-second note.

```
CL-GSA> (setf *scope* '(1 2 4 8))
```

to N3

In the N3 context, the aim is to format the MDS as an irreducible sonic object in order to be interpreted as an '*infon*', that is to say, a clique that can be learned as such. Here, the data has to be ordered according to the network involved, matching each track or voice with its respective SOM (Self-organizing Map as a neural network called MLT in N3).

In this way, the MDS is transposed as a matrix with the relative duration followed by the midi note for each voice of the midi score.

```
CL-GSA> (midi2mds (add-tie (scoring-midi "~/Kjolheia.mid"))
   :to 'N3)
((2 54 47 62 62) (2 59 62 59 66) (2 59 66 61 66)
 (2 59 62 54 66) (2 62 64 59 62) (2 59 62 47 67)
 (1 59 61 64 66) (1 -59 -61 62 -66) (1 59 66 61 67)
 (1 -59 -66 62 -67) (2 66 47 55 64) (2 59 62 59 67)...
```

Note that the function `add-tie` is applied for each track of the score.

to SC

For the performances `K540`, the midi file Kjølhiea has been converted to a MDS file as follows:

```
(midi2mds (mix-track (scoring-midi "~/Kjolheia.mid"))
   :out "~/Kjolheia" :to 'SC)
```

Note that the function `mix-track` allows mixing n tracks or voices to one.

The resulting score is formatted as a couple of lines respectively duration and midi note(s) as an array by track as follows:

```
2 2 2 2 2 2 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1...
[62,47,54] [66,62,59] [61,66,59] [66,54,62,59] [59,64,6...
2
```

Then, we can interpret the MDS in the SuperCollider context according to the section 3.2.3 on page 41. Mind setting a duration factor according to the tempo required.

Thus, the score is interpreted as an array and therefore can be read inside a stream in the Routine or part of Pattern – using `Pbind` for instance.

3.4 Discussion

This proposition is an attempt for 'normalization' for fixed scores as a dataset for 'non-real-time synthesis'. Indeed, MDS retains a partition which allows any kind of structural or formal analysis *de facto a posteriori* and algorithmically, on the understanding that does not solve the discrimination issue of the sonic object as an irreducible and signifier axiom[2].

Finally, it is also convenient as a file that can be read in different contexts such as *Neuromuse3* or SuperCollider to name but two which have been outlined in this article, and easy to write from traditional score or midi file, even to generate from or as raw data.

Also, this format allows convenient possibilities of conversion such as any object-oriented programming (OOP) or SC *Score* in *Non-Realtime Synthesis* (NRT).

[2]For instance the counterpoint in terms of relationship between different musical layers, or the sonic object as an emergent phenomenon and perceived as such, which can be subjective and hence dependant of the teleological object.

Chapter 4

Analytical modeling

2018 – 2019

4.1 Introduction

The main idea is to describe a sound object by analyzing its constituent elements in order to reveal structure characteristics, and use it in terms of composition. These analysis procedures depend on and are based on the segmentation of the musical object in terms of symbolic elements.

Axiomatic discrimination in terms of the irreducible segmentation of an object is an unanswered question in term of the philosophy of science. At least, there is no definitive answer, and we have to deal with the best solution, which remains often unsatisfying, especially when we are talking about human sciences. So, the discrimination is empirical and depends essentially on the analytic and cultural context, and which level of abstraction we are working on [Snyder, 2001], in other words, the axiomatization depends on what one seeks and how one seeks it.

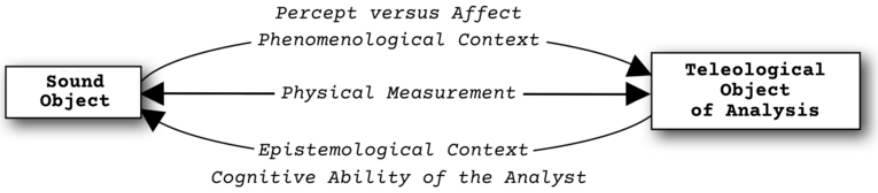


Figure 4.1: Synaptic diagram showing relationships of the sound object relative to the analyst.

Then, Encoding presupposes an interpretation in terms of *percept* – including *affect* – of the sound phenomenon considered, such as the teleological object of analysis in the musical context induces a qualitative or a quantitative axiomatization. The second can be derived from the first by empirically associating numerical values with a quality recognized as such, validating the relevance of the observation. The qualitative character depends on the phenomenological context and the cognitive ability of the analyst implying an *in situ* relationship – even an interaction – between the sound object and the analytical design of the model (see diagram on figure 4.1). Note that the cognitive ability of the analyst fits into the *épistémè* [Foucault, 1966].

The type of encoding is then decisive and depends on the analytic perspective considered.

This is one proposition for automatic analysis as modeling in a musical context and from a sound file in terms of difference and resemblance. Some steps are inevitably empirical, and the modeling is somehow more about computation optimization as resolutions of combinatorial issues.

This work falls within the margin of the project *Neuromuse3* and it has been inspired by the research report *Morphologie* [Voisin et al., 1999].

4.2 Encoding

The encoding consists to identify useful and isolable component – with algorithmic or symbolic discriminative processes – as the axiomatic of the studied system. This is always done in the teleological perspective of analysis and synthesis. Note that the musical object represents an abstractive level from the axiom to the whole object as a system.

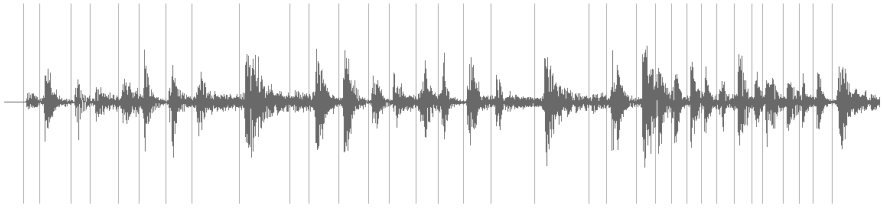


Figure 4.2: The waveform of the 5 first seconds of the sample with its associated segmentation according to the TextGrid generated with *enkode*. [→ 1.3]

***enkode* analysis**

The aim of this encoding is to normalize data from a sound file using the analytic tools of the software Praat managed by the command line *enkode*, defined as a multidimensional array of 5 dimensions. This array contains the duration, $f0$, centroid, loudness and 'loudbass'.

```
$ enkode --as-int=3 test.mp3 > test.dat
```

4.3 Symbolisation

Hierarchical clustering

The hierarchical clustering of the previous multidimensional data is built inside the artificial neural network *Neuromuse3* context – called CAH – according to the output of the neurons as events. The agglomerative process uses the Euclidean distance.

```
CL-USER> (require 'N3)
CL-USER> (in-package :N3)
N3> (defvar *DATA* (remove-duplicates (read-file "test.dat")
     :test #'equalp))
N3> (create-mlt 'test (length (car *DATA*)) (length *DATA*)
     :carte #'rnd-map)
N3> (loop for neuron in (neurons-list test)
     for dat in *DATA* do (setf (output neuron) dat))
N3> (dendrogram test 3 :and-data t)
;; The second argument is the aggregation type :
;; Ward's method in this case
-934.9051+
```

The function `dendrogram` generates a data file with the number of nodes according to the trimming distance associated with the minimum distance of the parent node and the sum of the intra-class inertia of the children nodes of the parent node.

Now, the idea is to get the optimum number of classes according to the distance from the parent node and the intra-class inertia. There are no rules, therefore the choice is empirical and estimated with the degree of accuracy analysis required. All it needs to know is that the distance has to be maximum and the inertia minimum.

However, in *Neuromuse3* context, the optimization can be done according to the number of the current *'fanaux'* in order to be more accurate by increasing their number.

In this case, this is a segmentation of 5 classes – referred to as A, B, C, D, and E – which is retained as result.

```
N3> (alpha-seq test -934.9051+ 5 (read-file "test.dat"))
(C C C A A C C D D A C B C A A E C D D A C A A B B B B B E
 B B B D A C A A E B B B B B E E B B B A C D E A A E A A C
 B B B B E A A A A B B A D A B C C A C E C D D A C D B A E C
 D A A A B B E A A E B A D A E D E A A E A D D C C B E A B B
 C E B D A B D A A E B A A A B B E A A B C D B B)
```

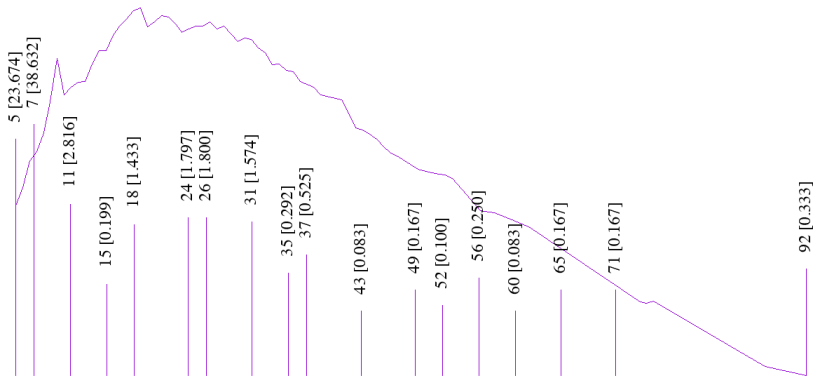


Figure 4.3: The curve is the sum of the intra-class inertia by trimming. The lines are the peaks of the curve of minimum distance from the parent node (as the second number in the square bracket, the first number is the number of classes at this point). Note that in this graph, the impulse segments are displayed in a logarithmic scale for better readability.

4.4 Contrastive analysis

Segmentation by marker

The contrastive analysis consists of segmenting an array of symbols according to a marker defined by the number of occurrences of this marker as the smallest sub-structure, or in other words according to the number of repetitions and for a short sequence which is more focused by the brain as a relevant marker to memorize. This is done recursively until all symbols are different. The side effect of this algorithm is, in the case of strict equality between different occurrences of sequences, the choice is done according to the sorting algorithm of the lisp implementation, in which we retain the first item. The function `structure-s` takes as argument the symbolic sequence previously computed.

The two last mergings affect sequentially – that is to say on the whole sequence – two adjacent items respectively if the first is equal to the head of the second (for instance the sequence `AB ABC` becomes `ABABC`), and if the second is equal to the tail of the first (for instance the sequence `ABC BC` becomes `ABCBC`).

```
N3> (structure-s (alpha-seq test -934.9051+ 5
      (read-file "test.dat")))
(CCC AACC DDACBC AAEC DDACAA BBBBBB EBB BDAC AAEBB BBB EE
 BBBBACDEAAEAA CBB BBEAA AABBAD ABCC ACEC DDACDBAE CDAA
 ABBEAAEBAD AE DEAA EAD DC CBE ABBC EBD ABDAA EBAA ABBEAABC
 DBB)
```

4.5 Paradigmatic analysis

Unrooted tree

The paradigmatic analysis allows us to observe typological variations within an object or a corpus.

There are no rules either for the paradigmatic discrimination, but an analysis by hierarchical clustering with the single linkage or the complete linkage as aggregation can offer some guidelines.

With this approach, we can accurately see the proximity between 'sub-structures' according to the current musical context. The main idea is to use the Levenshtein distance with some preliminary algorithms, which are respectively defined by the distance according to the local repetition and the distance between two bijective sequences as patterns a and b according to the decomposition into permutation cycle [Deléglise, 2010] – called σ – defined by $c = |O\sigma(x)|$, such as $\delta(a, b) = |lcm(c_a) - lcm(c_b)|$.

Let A and B be sub-sequences, the distance between A and B is computed as follow :

1. Remove common local duplicate(s) such as $A \to A'$ and $B \to B'$
   Then the 'repetition distance' is $d_1 = |A \setminus A'| + |B \setminus B'|$

2. Remove patterns such as $A'' = A' \setminus C$ and $B'' = B' \setminus C$ with $C = A' \cap B'$
   Then the 'transposition distance' is applied to the pattern C as d_2

3. Apply Levenshtein distance between A'' and B'' as d_3

Then the total distance is $d_1 \times w_1 + d_2 \times w_2 + d_3 \times w_3$ with w_i as weight respectively 1/2, 1/2 and 1 by default.

The setting – that is to say the aggregation type (single or complete linkage) and the weight of each algorithm applied to estimate proximity – remains empirical but the investigation field is significantly reduced and rather intuitive to integrate this modeling into an automatic process.

The figures 4.4 and 4.5 point out the possible groupings with the single and the complete linkage methods. These figures were generated with the online application iTOL[1] – with the display mode set to the unrooted tree –, from their respective Newick files built from the contrastive analysis as a dendrogram. See appendice 4 on page 144 for an alternative representation and appendice 4 on page 165 for a description of the standard Newick tree format.

```
N3> (dendrogram '(CCC AACC DDACBC AAEC DDACAA BBBBBB EBB
    BDAC AAEBB BBB EE BBBBACDEAAEAA CBB BBEAA AABBAD ABCC
    ACEC DDACDBAE CDAA ABBEAAEBAD AE DEAA EAD DC CBE ABBC EBD
     ABDAA EBAA ABBEAABC DBB) 1|2)
```

[1] iTOL allows to display online phylogenetic trees from datasets:
https://itol.embl.de/

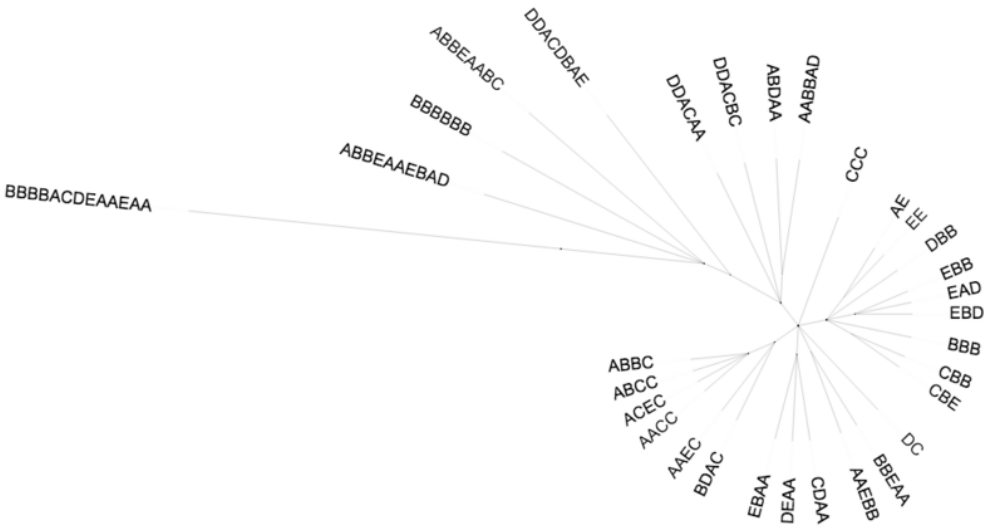


Figure 4.4: Single-linkage clustering.

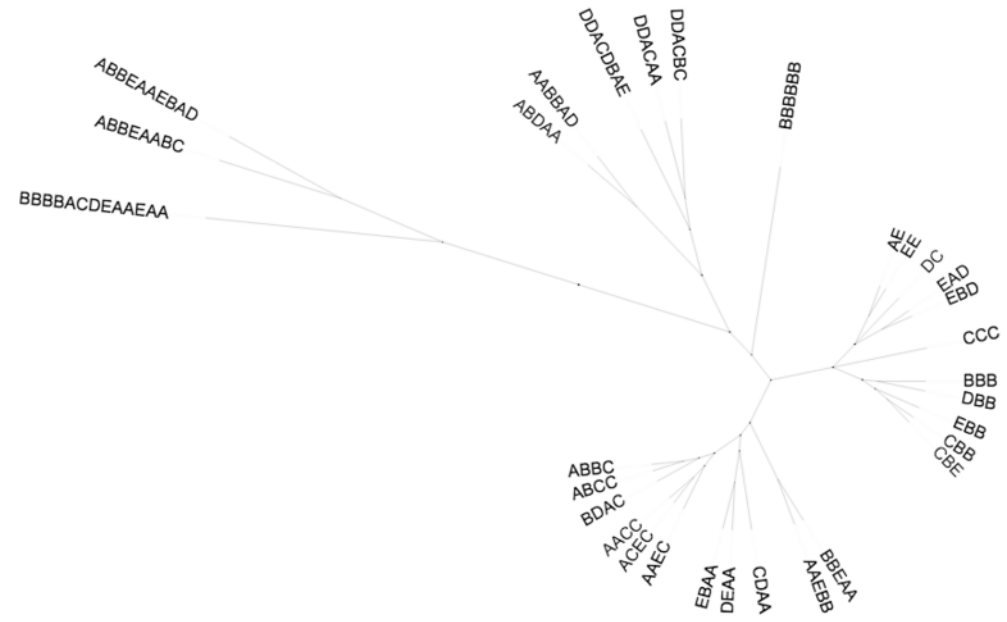


Figure 4.5: Complete-linkage clustering.

The single linkage tree can be divided into five paradigmatic fields, while the complete linkage tree offers the possibility to define more obviously four paradigmatic fields.

In any case, this is the teleologic object that determines the setting, both for the number of discrimination and the way the discrimination is done in terms of distance.

4.6 Systemic analysis

Clustering proximity

Defined as a set of relations that maintain elements between them allowing the constitution of a coherent system. Thus, form and structure are two interrelated notions that determine the immanent or transcendent view of the system.

The form can be expressed in terms of fractality from the micro-form to the macro-form, articulated according to structural modalities, or in terms of recurrence and repetition/variation [Analyse Musicale, 1990].

In some musical performances, some characteristics involve a morphogenesis point of view as a dynamic system. Morphogenesis is an 'in time' analytic system for observing formal variations according to identified structural processes. Indeed, some traditional musical events for instance are not pieces of music with a determined form, but rather a piece of music that evolves 'in time' according to the feelings and some codification in terms of proclivity involving each participant.

So, the systemic analysis will focus on the relationship between adjacent sub-structures defined by the contrastive analysis as derivative according to the distance defined for the paradigmatic analysis, and the process of successivity in terms of the probability of the elements constituting the sub-structures.

4.6.1 Derivative clustering

According to the distance between two adjacent sequences, the derivative clustering (see figure 4.6) consists of segmenting the whole sequence into parts 'in time' delimited by the mean distance.

```
;; Thus, the initial sequence is segmented as follow :
N3> (part-s '(CCC AACC DDACBC AAEC DDACAA BBBBBB EBB BDAC
    AAEBB BBB EE BBBBACDEAAEAA CBB BBEAA AABBAD ABCC ACEC
    DDACDBAE CDAA ABBEAAEBAD AE DEAA EAD DC CBE ABBC EBD
    ABDAA EBAA ABBEAABC DBB))
((CCC AACC DDACBC AAEC) (DDACAA BBBBBB) (EBB BDAC AAEBB BBB)
 (EE BBBBACDEAAEAA) (CBB BBEAA AABBAD ABCC) (ACEC DDACDBAE
  CDAA ABBEAAEBAD) (AE DEAA EAD DC CBE ABBC EBD ABDAA) (EBAA
  ABBEAABC))
```

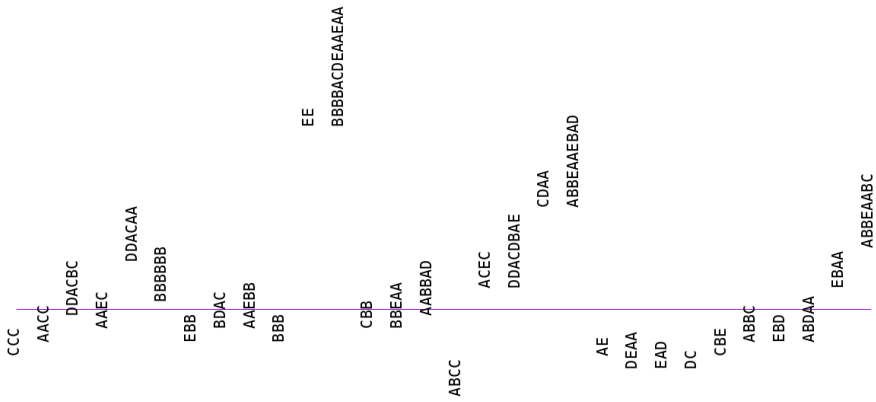


Figure 4.6: The first letter of each sequence marks the distance level – on the y axis – to the next sequence. The horizontal line is the mean distance involved in all combinations of two different sequences.

Note that the last sub-sequence `DBB` is omitted because there is no distance defined from this sequence, but this sequence is of course implicitly associated with the last sequence `ABBEAABC` as distance.

4.6.2 Developmental process

In this analysis, the approach consists of evaluating the probability of an event occurring according to n previous events. For instance, the probability of events succeeding the sub-sequence `CD` is :

```
N3> (next-event-probability '(C D) (alpha-seq test
    -934.9051+ 5) :result :verbose)
B => 28.571 %
A => 14.286 %
D => 42.857 %
E => 14.286 %
```

Mind that the sum of the probabilities is equal to 100 %, or very close according to some rounding error that might be caused by computer systems [Goldberg, 2001].

4.7 Resolution

During this article, we proceeded to a ‘deconstruction’ of a sample as a sound file – according to the discriminative analysis of *enkode* as events, and moreover as

symbols and as sub-structures and their relationship – with a view to or in the perspective of a 'reconstruction' according to for instance a formal grammar defined as a musical L-system [Manousakis, 2006], or a Markov chain, which could be weighted as a developmental process.

According to the transition probability matrix defined by the function `next-event-probability`, it is possible to experiment with a Markov chain as follows:

- let S be the initial sequence according to the alphabet $e = \{a, b, c, ...\}$ such as $e \subset S$
- let $P(e)$ be the probability of an occurrence e_n in S
- let w be the sub-sequence as the previous state and set with an initial element such as $w = P(e)$ or $w = e_n$ then the next event is $P(e|w)$
- if $P(e|w)$ does not exist or if $P(e_n|w) = 1^*$ with $|w| > 1$ then minimize $i \in]0, |w[i]| = 1]$ such as $\exists i \in \mathbb{N} : P(e|w[i]) \in e$ knowing that $w[i]$ is the position of and from the beginning of the reduced w as a tail sub-sequence, or in other words as a suffix.
- if $P(e|w) = 1$ with $|w| = 1$ then the next event is $P(e)$ in order to avoid an endless loop on a sub-sequence.

  * In this case and from this event, we have to solve the *max order problem* [Papadopoulos et al., 2014]. Indeed, the sequence generated will strictly be a copy of the initial sequence and does not allow any variation of the latest. This behavior is obviously not opportune in this context.

Also, even if this work is done on a portion of a piece of music or on the whole musical work, this analytic process is done *a posteriori* and more about a structural relationship – and according to the principle of immanence –, in other words 'out time', a bit like a background process in an artificial intelligence context – especially in reference to the brain activity during sleep for instance.

4.8 Discussion

The '*a posteriori*' structural analysis is naturally different from the idea that could be done in real time. During the temporal flux, several factors interfere:

- the marker delimiting two sub-sequences, this one can evolve over time and be different for each discrimination (the incoming information can change or consolidate the probabilities of the acquired);
- the concentration and the type of focusing – which can be versatile – during listening;

- and the passive of the subject, notably about his/her musical education and his/her own experience of the sound phenomenon.

It also exists some bias as what I call the discriminatory paradox. Indeed, the interpretation of a potential result, notably regarding multidimensional discrimination, can elude one character for the benefit of another. In other words, the discriminatory paradox is defined by the fact that it can exist inside a subset of two elements a and b whose distance – or value of dissimilarity – is superior to two elements a and c belonging respectively to two distinct subsets.

This illustrates the elusive character of an 'objective' analysis. In practice, this consists of minimizing these factors to reach a formalism proposing convincing modeling. This can be done with repeated listening of the work allowing a holistic analysis, or with an algorithmic analysis or a synthesis of different types of analysis but *a posteriori*. In both cases, time is required; and the result remains dependent on the axioms (or prerequisites) – i.e. the formalization step – and the teleological object – i.e. the modeling step.

4.9 *Addendum*

Minimal Spanning Tree

The Minimal Spanning Tree – noted MST – is a graph built from a given undirected tree – as a matrix of distances – that connects each node by the minimal distance (corresponding to the weight of an edge) without creating a cycle.

For this purpose, it exists at least three well-known algorithms as Borůvka's algorithm established by Otakar Borůvka in 1926 (also called Sollin's algorithm), Kruskal's algorithm established by Joseph Kruskal in 1956, and Prim's algorithm developed in 1930 by Czech mathematician Vojtěch Jarník and later rediscovered and republished by computer scientists Robert Clay Prim in 1957 and Edsger Wybe Dijkstra in 1959 (it is also sometimes called the Jarník's algorithm, Prim-Jarník algorithm, Prim-Dijkstra algorithm or the DJP algorithm).

The Common Lisp library *cl-mst* allows computing MST with one of the algorithms previously enumerated[2] and representing MST using the command line Neato[3].

https://github.com/yannics/cl-mst

The MST allows the clustering analysis, such as the classification of spectral forms of the birds singing [Voisin, 2011]. Also, some tools from the library *cl-mst* such as the degree of a node – interpreted musically in terms of structural articulation – or the path between two nodes – interpreted in terms of contrast or repetition/variation can offer some interesting analysis guidelines.

[2]These algorithms are described in the *README* file of *cl-mst*.

[3]Neato is one of the tools available in the open-source package Graphviz for drawing undirected graphs – using 'spring' models [Kamada et al., 1989] – specified in DOT language script.

The MST can be applied from the data file generated previously as *test.dat*, in order to get an overview of discriminative boundaries.

In the present context, the MST is applied according to the Euclidean distance between the events defined in *test.dat*.

```
CL-MST> (defparameter *distance-matrix*
  (let ((a (remove-duplicates (read-file "test.dat")
     :test #'equalp)) r)
    (dotimes (i (length a) (nreverse r))
      (loop for j from (1+ i) to (1- (length a)) do
           (push (list (nth i a) (nth j a)
                 (N3::euclidean (nth i a) (nth j a))) r)))))
CL-MST> (defparameter *color-map* (mapcar #'list (loop for s
    in '(C C C A A C C D D A C B C A A E C D D A C A A B B B
    B B B E B B B D A C A A E B B B B B E E B B B B A C D E
   A A E A A C B B B B E A A A A B B A D A B C C A C E C D D
    A C D B A E C D A A A B B E A A E B A D A E D E A A E A
   D D C C B E A B B C E B D A B D A A E B A A A B B E A A B
    C D B B) collect (cadr (assoc s '((A coral) (B
   chartreuse4) (C dodgerblue2) (D darkorchid1) (E sienna))
  :test #'equalp))) (read-file "test.dat")))
```

From this step, the choice of the algorithm depends on the analysis context, but in most cases, Borůvka's algorithm turns out the most suitable.

```
CL-MST> (neato (boruvka *distance-matrix*) :alpha t :len t
   :scale t :color *color-map*)
```

However, the MST in figure 4.7 on page 56 remains difficult to discriminate algorithmically in regard to the hierarchical clustering using Ward's method described in the chapter *Symbolisation*, which correspond to the different colors on the graph.

Also, the MST can be applied to the contrastive analysis in order to classify subsequences as an alternative paradigmatic analysis[4].

```
CL-MST> (defparameter *distance-matrix*
 (let ((a '(CCC AACC DDACBC AAEC DDACAA BBBBBB EBB BDAC
    AAEBB BBB EE BBBBACDEAAEAA CBB BBEAA AABBAD ABCC ACEC
    DDACDBAE CDAA ABBEAAEBAD AE DEAA EAD DC CBE ABBC EBD
    ABDAA EBAA ABBEAABC DBB)) r)
  (dotimes (i (length a) (nreverse r))
    (loop for j from (1+ i) to (1- (length a)) do
      (push (list (nth i a) (nth j a)
        (N3::structure-distance (nth i a) (nth j a))) r)))))
```

[4] A third-party Common Lisp library called *mk-string-metrics* can be useful to compute distance between two strings as Damerau-Levenshtein distance, Hamming distance, Jaccard similarity coefficient, Jaro distance, Jaro-Winkler distance, Levenshtein distance, Normalized Damerau-Levenshtein distance, Normalized Levenshtein distance and Overlap coefficient.
https://github.com/cbaggers/mk-string-metrics

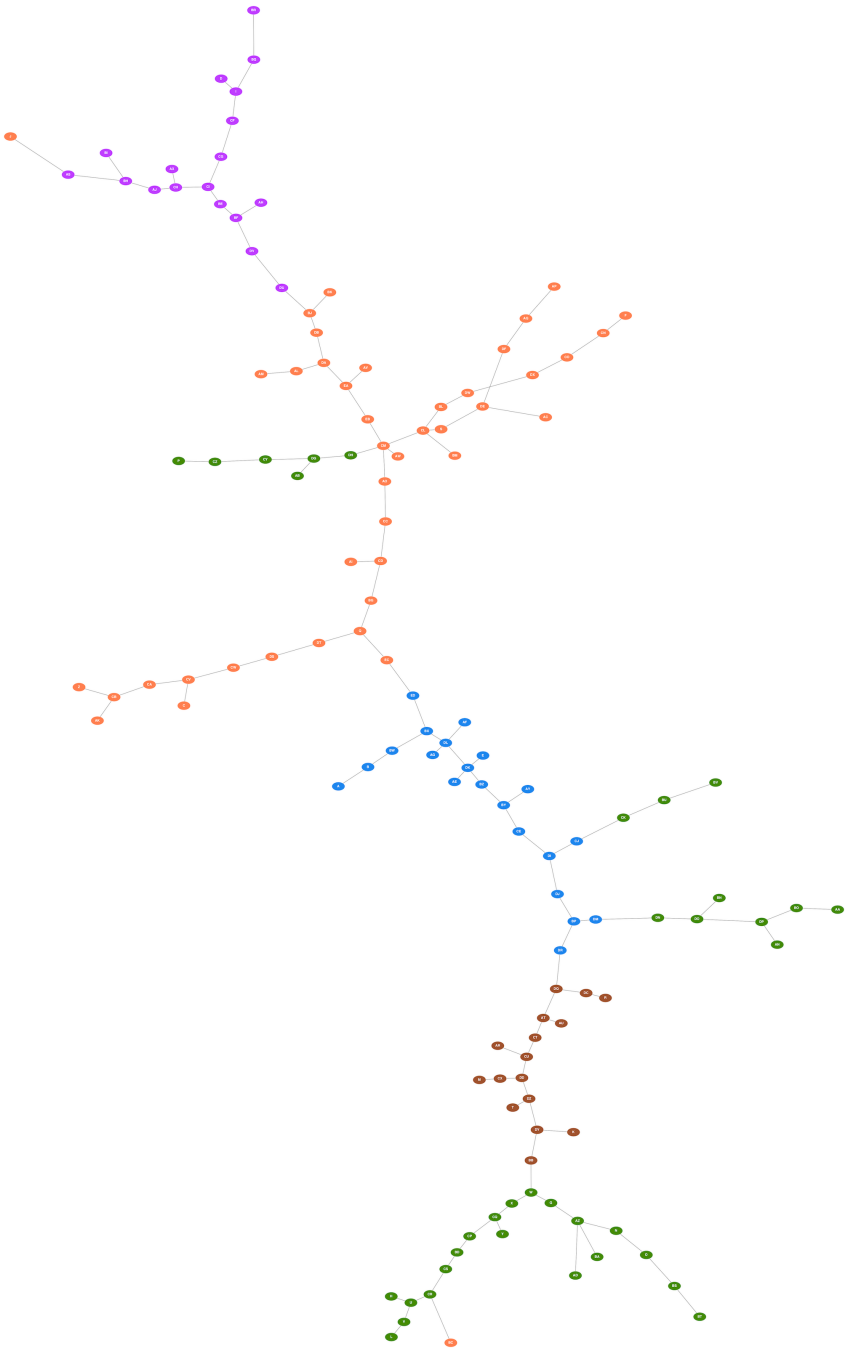


Figure 4.7: MST from *test.dat* using Borůvka's algorithm.

Here also, the choice of Borůvka's algorithm imposes itself.

```
CL-MST> (neato (boruvka *distance-matrix*))
```

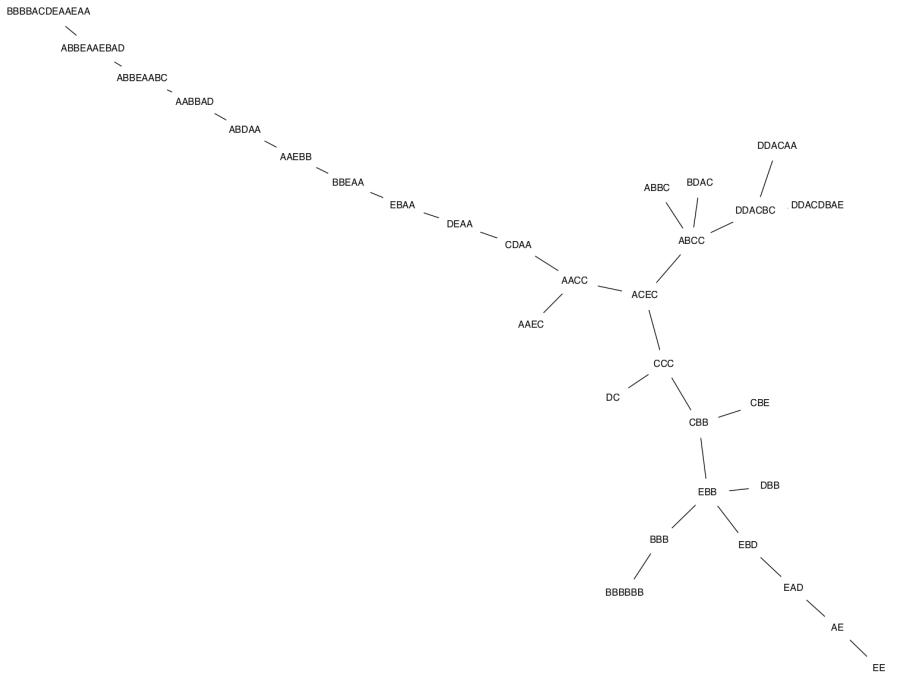


Figure 4.8: MST of the contrastive clustering using Borůvka's algorithm with the distance as described in the chapter *Paradigmatic analysis*.

On the other hand, the MST in figure 4.8 offers an interesting view[5] as structural relationships, and a different approach for paradigmatic discrimination.

For instance, we can consider three branches from the node ACEC. Or even more relevant, we can cut the tree in two parts between the nodes AACC and ACEC, according to the branch defined by the path from BBBBACDEAAEAA to AAEC, and the branch defined by the path from DDACDBAE to EE.

[5] Note for a better displaying of the graph, some attributes of the DOT file have been modified, as `overlap=compress` and `fontsize=12`.

Chapter 5

Motivic Recognition

2015 – 2020

5.1 Introduction

This article aims to propose an indexing method by estimating the amount of change or variation between two profiles according to a common motive. The profiles in question are symbolic in the sense that the numerical values can refer to any kind of data as melody, dynamic, along with others but in time which is expressed in terms of ordered durations – such as the formatting of the MDS described in the chapter *Music Data Score*. This approach focuses on the common rhythmic structure in relation to the first derivative applied to it.

The originality of the current approach in the case of a melodic line is to consider its profile only on the sync onsets in order to elude any kind of melodic variations as '*note de passage*', '*monnayage*', '*échappée*', '*broderie*', etc.

This work does not take into account the cognitive capacity of the listener or any database, nor the music expert's evaluation because there is no consensus within the musicologist communities to estimate the proximity of two melodic snippets, at least in terms of relevant parameters, excepts for traditional folk music for which the similarity focuses on tonalities (as interval ratio, ranking notes according to the tone, etc.) and/or rhythm in terms of downbeat among others, raising the quantization problem [Desain, 1992] in some non-trivial case. Indeed, for instance, two 'identical' melodies – that is to say the same pitches and the same rhythm – can be perceived very differently according to their respective articulation, tempo, and context. And *a contrario*, the feeling of two snippets expressed very differently – in terms of the opposition of timbres, ornamentation, rhythmic subdivisions, harmonic substitution, and so – can sound very close, even seen as the same basic melody [Hanna et al., 2008]. The border between

family resemblance and difference – even '*différance*'[Derrida, 1967][1] – is at the least blurry.

This proposition assumes its own bias and aims to be useful in certain circumstances, notably in the context of *Neuromuse3* in order to recognize snippets as input sequences and interact knowingly. This algorithm is therefore part of the package N3 – from version 3.0.9 – and hence is coded in Common Lisp language.

5.2 Description

The main function `differential-vector` is an 'experimental' proposition allowing us to estimate numerically the distance in terms of motivic proximity between two sequences. This distance is defined according to the cartesian coordinate system as a two-dimensional vector i.e. two criteria defined on the x-axis by the events concordance in relation with the length of the sequences in time as a motive and on the y-axis defined by this motivic profile concordance in terms of signs of the first derivative.

This implies some algorithmic processes that can be described as follows (note that all keys described below can be parameterized from the main function `differential-vector` and the uppercase words refer to functions as subprocesses):

A – Evaluating x.

(a) Computation of the coordinate x with the functions:

`DUR->ONSET`

Convert normalized ordered durations to onsets in time for each sequence according to one of the following modalities:

`:ended :ignore`

set by default, the last duration is ignored;

`:ended :first`

the last duration is taken into account as a cyclic pattern;

`:ended :last`

the last duration is taken into account with the first derivative equal to zero.

`SUBSEQ-THRES`

Match events from one sequence to the other according to a given threshold – key `:thres` – applied as a percentage of the minimal duration of the sequences involved (0.2 by default).

[1] This concept fits into Derrida's deconstruction and plays with on one hand the French homophony '*différence*/*différance*' referring in broader terms to an ontological difference and in another hand the polysemy of the french word '*différer*' (deferral in English) in which reality can only be understood as differential gaps hidden in cultural heritage in space and in time, leaving out or postponing any kind of 're-construction' or conclusion as an open problem.

`FILTERNOWAY`
Remove if needed onset duplicates on adjacent clusters.

(b) Get the percentage of the timing concordance according to the keys:

`:opt :mean`
set by default, divided the number of concordance by the average of cardinals of the two sequences;

`:opt :max`
divided the number of concordance by the maximal cardinal of the two sequences;

`:opt :min`
divided the number of concordance by the minimal cardinal of the two sequences.

B – Evaluating y.

(a) Computation of the coordinate y according to the rhythmic motive by associating for each selected onset their respective event values, then the function:

`LX->DX`
compute the first derivative of the motive by sequence according to the 'dominant value' of each cluster defining the motive estimated through the keys:

`:cluster :median`
set by default, retains the mean value of the bigger group in terms of first derivative signs (in case of equality this is the mean value of the union of these groups);

`:cluster :mean`
retains the mean value of the cluster;

`:cluster :maxima`
retains the maximal value of the cluster;

`:cluster :minima`
retains the minimal value of the cluster.

(b) Convert the first derivative as signs:

$+1$ positive slope,
-1 negative slope,
0 no slope;

(c) apply tolerance if set to `:yes` (`:no` by default) – that is to say the values $+1$ and 0 or -1 and 0 counts as common values;

(d) counts the common signs of the motives and get the mean value.

C – Values of the output (`:diff` as the level of difference, `:sim` as the level of similarity, `-norm` as the normalised euclidean norm of the vector, `-coord` as the coordinate vector and `-list` the norm with x and y):

`:result :diff-norm` (set by default)
$$\frac{\sqrt{(1-x)^2+(1-y)^2}}{\sqrt{2}}$$

`:result :diff-coord`
$$((1-x)\ (1-y))$$

`:result :sim-norm`
$$\frac{\sqrt{x^2+y^2}}{\sqrt{2}}$$

`:result :sim-coord`
$$(x\ y)$$

5.3 Evaluation

Let the following trivial example in figure 5.1 be an illustration of the behavior of the algorithm, notably the different possible evaluations in relation to the length of the sequences (key `:opt`) and the result in terms of proximity (key `:result`).

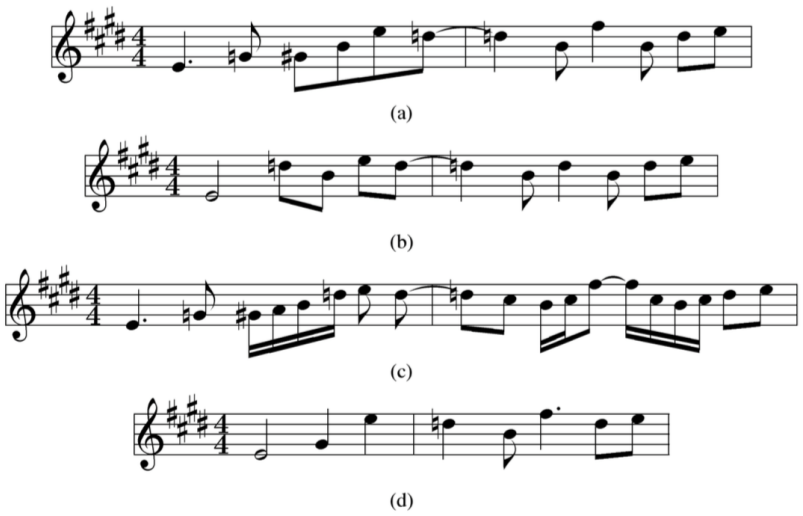


Figure 5.1: (a) Theme of The Beatles' song *Day Tripper* (b) Variation with same tonic E and same dominant B as Em tone (c) Variation with '*notes de passage*' (d) Variation same notes on downbeats [Hanna et al., 2008, p. 115].

Let the tables 5.1 and 5.2 be the results of the function `differential-vector` as the semi-matrix of distances applied to the four snippets of figure 5.1, such as the first number of each cell is the percentage of similarity or difference fol-

lowed by the coordinate vector between square brackets (rounded to two decimal places).

:diff	:opt :min	:opt :max	:opt :mean
ab	08 [0.00 - 0.11]	10 [0.09 - 0.11]	09 [0.05 - 0.11]
ac	00 [0.00 - 0.00]	25 [0.35 - 0.00]	15 [0.21 - 0.00]
ad	09 [0.12 - 0.00]	26 [0.36 - 0.00]	19 [0.26 - 0.00]
bc	08 [0.00 - 0.11]	30 [0.41 - 0.11]	20 [0.26 - 0.11]
bd	00 [0.00 - 0.00]	14 [0.20 - 0.00]	08 [0.11 - 0.00]
cd	09 [0.12 - 0.00]	42 [0.59 - 0.00]	31 [0.44 - 0.00]

Table 5.1: Evaluation of the semi-matrix of distances as difference.

:sim	:opt :min	:opt :max	:opt :mean
ab	95 [1.00 - 0.89]	90 [0.91 - 0.89]	92 [0.95 - 0.89]
ac	100 [1.00 - 1.00]	84 [0.65 - 1.00]	90 [0.79 - 1.00]
ad	94 [0.88 - 1.00]	84 [0.64 - 1.00]	88 [0.74 - 1.00]
bc	95 [1.00 - 0.89]	75 [0.59 - 0.89]	82 [0.74 - 0.89]
bd	100 [1.00 - 1.00]	91 [0.80 - 1.00]	95 [0.89 - 1.00]
cd	94 [0.88 - 1.00]	76 [0.41 - 1.00]	81 [0.56 - 1.00]

Table 5.2: Evaluation of the semi-matrix of distances as similarity.

These tables show a maximal difference of 42% with the key `:max` (snippets `cd` table 5.1) and 100% of similarity with the key `:min` (snippets `ac` and `bd` on table 5.2). For the last two cases, that means all notes of the shortest snippet match with the longest and the melodic profile follows the same shape. Knowing that the key `:mean` is a good compromise to moderate this bias.

Also, the D♯ of the snippet (d) does not match with others because of the delay of the onset, and the first B of the snippet (b) disrupt the melodic profile on the snippets (a) and (c) (y`:diff`$= 0.11$ and y`:sim`$= 0.89$).

Note that difference and similarity as normalized vectors (by dividing the norm of the vector by the square root of two) are not complementary values in that sense their sum is not always equal to 100%. More specifically, when the absolute value of x minus y tends toward one, then the summation of the normalized norm of the vectors, respectively as the difference distance D and as the similarity distance S, tends toward the square root of two:

$$\text{if } |x - y| \to 1 \text{ then } \frac{||\overrightarrow{D}||}{\sqrt{2}} + \frac{||\overrightarrow{S}||}{\sqrt{2}} \to \sqrt{2}$$

and furthermore

$$\text{if } |x - y| \to 0 \text{ then } \frac{||\overrightarrow{D}||}{\sqrt{2}} + \frac{||\overrightarrow{S}||}{\sqrt{2}} \to 1$$

5.4 Discussion

The relevance is relative but this algorithm – like others – gives some guidelines in paradigmatic terms as family likeness focusing on the event onsets sync proportionally and on the first derivative profile as the sign values.

Also, one of the possible applications of this motivic research for a given sequence allows – according to a given number of events as a pattern or directly a given pattern – to evaluate sequentially the amount and the relevance of redundancy of this pattern through the considered sequence until this sequence as the whole. Thus, this method can detect all 'fractality' in the sequence for a deliberate pattern (See *Discussion* in appendix 4 on page 147).

Part II

ELECTRONIC PART AS ALGORITHMIC CONCEPT

Chapter 6

Formal and structural sketches

2014 – 2019 (rev.2023)

Initially this work has been initiated as an International Musical Project aiming to share compositional strategy into one work.

This is some 'formal and structural sketches' that I propose.

These algorithms are intended to be used in the SuperCollider context. Some analysis might require the software Praat. The software Lilypond is also required in order to display scores. Also, some Common Lisp libraries as *cl-cycle*, *cl-gsa* and *Neuromuse3* can be useful to complete this work.

This writing is a work in progress, and therefore can evolve according to the experience *in situ* and new ideas.

A philosophical point

This compositional approach consists in adopting a transcendent point of view of musical creation. The structural models presented in algorithmic form allow an adaptability of the initial material as a musical idea to be developed according to its own inner structure.

These models are historically and analytically inscribed as musical forms such as sonata form, fugue, canon, etc ... according to some outlines allowing to reinvent themselves as the proportional canon, to develop as a fractal process and to create as the 'peak morphing' articulation.

Time-based algorithms are understood in terms of repetitions/variations. It goes from the strict repetition, if such a phenomenon could exist, to the most striking contrast in term of opposition. It remains to know what kind of opposition that can be done depending on the considered object.

6.1 *Electronic background sketch*

The electronic background consists of realizing a formal and structural direction according to some algorithmic process managed with a SuperCollider code. The material is computed in Common Lisp using the library *cl-gsa* from samples and/or scores (shortly this consists of 'transposing' a given melody to a weighted tone).

Then the direction consists of determining the duration of transformation between two events. An event is a combinatorics – see algorithm 1 ∼OCWR and OCWR box description – of frequential profiles. These combinatorics are included as set inside a combination called `symmetric-permutation` from the library *cl-cycle*. Each frequential profile is determined by an array of frequencies (as peaks), a respective array of amplitudes, and a respective array of bandwidths (interpreted as grain) plus a resonant filter. This last is divided into 3 bandwidths which are correlated in temporal directivity to the three first formants and their respective bandwidths and intensity from Praat analysis of the recorded voices according to an appropriate concatenation. This last is scaled in order to match the total duration of the score.

The transformation – that I call 'peak morphing' – consists of a linear interpolation between 2 points that match a minimal distance.

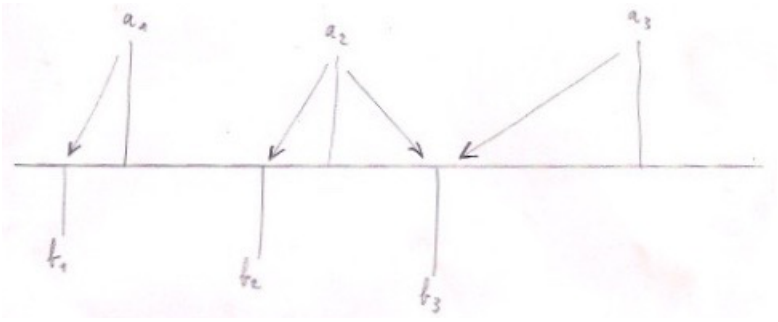


Figure 6.1: Peak morphing from a to b.

$a_1 \to b_1, a_2 \to (b_2, b_3), a_3 \to b_3$

The formal direction can be determined in terms of spatialization, with x as the left/right axis, y as the front/back axis, and z as the distance axis. The latter (see appendix 4 on page 145) will determine the presence level (or conversely the absence of sound depending on the audibility threshold). These coordinate will be effective inside the total duration D by a successive transformation duration d according to the equality $D = \sum_{i=1}^{\lambda} d_i$ with $\lambda = card(\mathcal{C}_n)$ where $n =$ the number of synthesis and $\mathcal{C}_n$ the set of combinatorics with n synthesis.

Algorithm 1 $\sim$OCWR $(list \mid R, dec)$

```
/* OCWR for Ordered Combination Without Repetition */
if list ≠ nil then
    R =[ list ]
    dec = |list|
if dec = 1 then
    return R
else
  for item in R do
    if |item| = dec then
      for position from 0 to |item| − 1 do
    R.add ( item\item [ position ] )
    ∼OCWR ( nil, R, dec − 1 )
```

Note that for $\mathcal{C}_n = \{C_1, C_2, \ldots, C_\lambda\}$ the 'peak morphing' duration d_i will be define from C_{i-1} to C_i. This means that we have to initiate the process with the combination C_0.

OCWR description

In other words the cardinal of OCWR is the number of k-combinations for all k minus the empty set defined by:

$$\sum_{0 \leq k \leq n} \binom{n}{k} = 2^n$$

For instance, with 3 elements there are 8 combinations (subsets) including the empty set:

$|\{\{\}; \{1\}; \{2\}; \{3\}; \{1,2\}; \{1,3\}; \{2,3\}; \{1,2,3\}\}| = 2^3 = 8$

Representing these subsets (in the same order) as base 2 numbers:

$0 - 000$
$1 - 001$
$2 - 010$
$3 - 100$
$4 - 011$
$5 - 101$
$6 - 110$
$7 - 111$

Thus, the formal direction of each parameter will be defined by (i) determined functions (algorithms), (ii) stochastic distribution and (iii) according to the instrumental score; (i), (ii) and (iii) can be correlated and interdependent as $d = f(i)$ in terms of structural direction such as $\sum_{i=1}^{\lambda} f(i)_i = D$.

Algorithm 2 ∼PEAKMORPHING (A, B)

args
A = initial profile
B = final profile

$result = [\,]$
for a in A **do**
$result.add([\,a, nearest(B, a)\,])$
for b in B **do**
$result.add([\,nearest(A, b), b\,])$
return *result*

In closing, let's see the detailed procedure called initialization for this soundscape.

1. Normalize the weighted profiles of the sample(s) and/or the score(s) generated by `m2tab` as an array.

2. Determinate combination

   (a) Set the algoritm 1 ∼OCWR according to the number of weighted profiles.

   (b) Compute symmetric permutation according to the length of OCWR regardless of $2^n - 1$ (minus one excluding the empty set).

   For instance with $n = 3$ as the number of profiles according to point 1., the symmetric permutation takes as argument a random permutation or a deliberate permutation – regarding the number of permutations for example – of `(0 1 2 3 4 5 6)` for `lst` and for `code-lst`.

   (c) Associate the combination of OCWR to the values of the flattened list of a symmetric permutation as indices applied to OCWR.

3. Group the profile(s) according to the previous combination – point 2.(c) – as indices of the array of profiles.

4. Apply the algorithm 2 as a 'peak morphing' distribution between two successive events according to the sequence defined in point 3.

5. Set the duration transformation according to a deliberate shape.

   For instance, a sinusoid allows one to stretch and compress a sequence of durations like an elastic time line.

6. Allocate into buffers the formantic analysis – see `formantAnalysis.sh` in appendix 3 on page 143 – realized with some given recorded voice(s) as respectively the intensities, the three first formants, and their associated bandwidths.

Then, the sequence (3) is played with some resonant noise according to the frequencies as approximate rates, articulated with the 'peak morphing' (4) – following the sequence of durations (5) – and filtered according to some resonances set by the formants (6).

6.2 *Proportional canon as an event climax*

Here are some introductory definitions:

Proportional canon

The canon in music consists of repeating a melodic phrase – or, by extension, a series of sound events – according to various types of formal modalities in a contrapuntal relationship. One of them – called proportional – imitates the melodic line by changing the rhythmic values by increasing or decreasing in a proportional relationship, such as the resulting is a polyrhythm according to some factors of the initial tempo.

Event climax

A climax depicts a progression as an arsic movement followed most of the time by a thetic movement. This progression results in an increasing information flow – may refer in this case to the density of events – to a peak emphasizing the maximum intensity of the climax, and then might decrease in a release time logic.

In this model, the proportion concerns as well as the duration of the sample than the position in time of the acme of the climax. Therefore, the algorithm consists of computing the duration of each voice according to a given ratio (or computed ratio via the durations of the first and the last voices), and their respective delay in order to match the acme of the climax. The position of the acme of the climax depends of the value of the ratio ρ (knowing that ρ has to be a value between 0 and 1 respectively excluded), which has an impact on the delay d such as (see figures 6.2 and 6.3):

$$d = \begin{cases} d^{+} & \text{if } \frac{1}{2} < \rho < 1 \\ d^{-} & \text{if } 0 < \rho < \frac{1}{2} \end{cases}$$

In this order, there are two ways to generate the proportional canon as an event climax, which consists of computing the durations and the delays of each voice according to the ratio ρ:

A - According to the figure 6.2, let A be the first voice B the last voice, with $\rho \in]0,1[$, then $B = A\rho$. The delay of the last voice equals $d = A\rho - B\rho$ or $d = A\rho(1-\rho)$. Thus, each intermediary voice will be a linear interpolation between the delays 0 and d, and between the durations A and B.

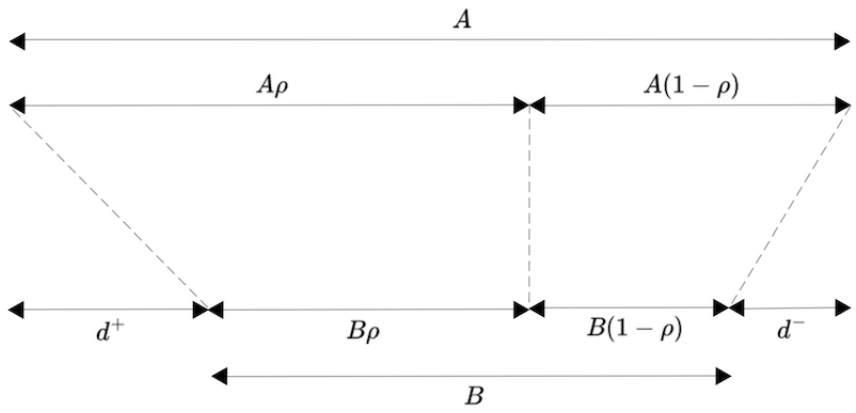


Figure 6.2: Illustration of the proportional canon with linear interpolation. In this example, the value ρ is between $\frac{1}{2}$ and 1.

B - Another way to realize the proportional canon is to apply the ratio ρ for each voice (see figure 6.3) according to a given duration for a given voice. Let T be the set of n durations (in other word n is the number of voices) according to the ratio ρ for a given duration d_x of the x^{th} voice, then

$$T = \bigcup_{i=1-x}^{n-x} d_x \rho^i$$

and let D be the set of the delays respectively assigned to T according to the total duration d_t such as

$$D = \bigcup_{i=1}^{n} d_t(\rho - \rho^i)$$

Note some outstanding values as $\rho = \frac{1}{2}$ for a perfect symmetry.

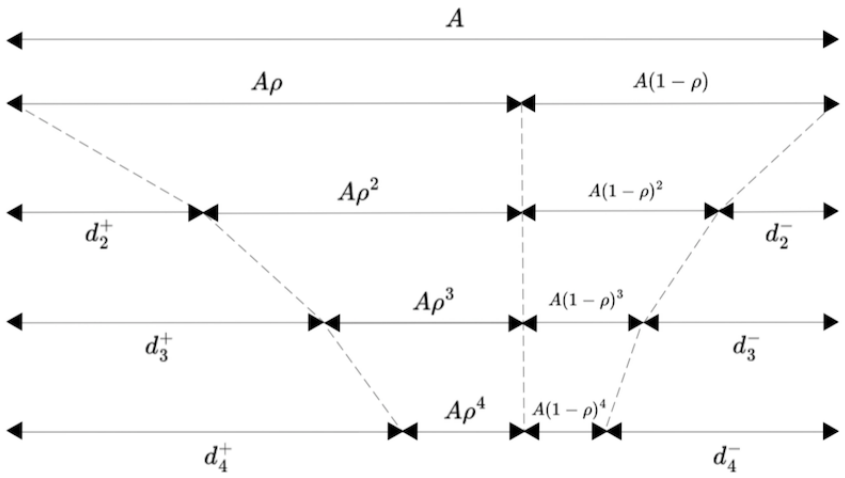


Figure 6.3: Illustration of the proportional canon on 4 voices with the ratio ρ applied recursively. In this example, the value ρ is also between $\frac{1}{2}$ and 1.

The construction of the initial rhythm can be intuitive (invention), referenced (quotation), and/or even created algorithmically. In the last case, it is possible to create rhythms with algorithmic processes such as those described with the libraries *cl-cycle*[1] and *cl-gsa*[2] inter alia.

From here, it is possible to synchronize the result according to a minimal note length as duration. Then, let C be the set of the durations of the canon (such as items of C is the set of durations of i voices) and m the duration of the minimal value such as

$$syncDur = \bigcup_{i \in C} \bigcup_{j \in i} \left[\frac{j}{m} \right] \cdot m$$

A last word concerning the melody and the counterpoint. The melody – or other as a *klangfarbenmelodie* for instance – can be considered according to some possible strategies.

The first one consists to attribute the same melody for each voice according to their respective 'tempo'. In this case, the 'counterpoint' consists to adjust the ratio ρ.

[1] The *cl-cycle* library is a set of algorithms that use the principle of cyclicality from a numerical sequence. The latter refers to an axiomatic symbolization by metaphorical transposition – or other – enabling the interpretation of a 'mathematical reality' in a 'musical reality' by correlation.

[2] The *cl-gsa* library can be used to analyze and interpret a melody to create a weighted frequency profile correlated with the melody. However, these tools can be diverted to create rhythms from some relevant weights generated by the analysis.

The second one consists to manipulate the melody by transformation and/or by constraints – or any kind of variation – on the melody itself, or on the melodico-rhythmic resultant of the canon into a dialectic counterpoint.

Algorithm 3 ∼PROPORTIONALCANON $(n,\ d_x,\ \rho,\ x)$

args

n = number of voices

d_x = total duration or x^{th} voice duration when x is set, implying a value between 1 (the first voice) and n

ρ = ratio between 0 and 1 respectively excluded

$x = x^{th}$ voice with the duration d_x; if x is set – i.e. $x \neq nil$ – this implies the algorithmic process described in $\boldsymbol{B}$

if x is nil **then**
  /* algorithmic process $\boldsymbol{A}$ */
  $T = n.interpolation[\, d_x, d_x\rho \,]$
  $D = n.interpolation[\, 0, d_x\rho(1-\rho) \,]$

else
  /* algorithmic process $\boldsymbol{B}$ */
  $T = [\,]$
  $i = 1 - x$
  while $i = n - x$ **do**
    $T.add\,[d_x\rho^i]$
    $i = i + 1$
  $D = [\,]$
  for i from 1 to n **do**
    $D.add\,[T[0](\rho - \rho^i)]$

return $(T, D)^T$
/* The result is a list of durations with delays. */

From here, we can add a third point in a way to consider only the rhythmic dimension. Like this, we are free to consider each voice independently of a predefined melody, on their own or in a holistic way.

Of course, all these points are empirical, but they can be integrated into a system of concepts. For instance, the number of voices can be equal to the number of notes of a given chord, thus each voice plays a unique note respectively from the notes composing the aforesaid chord as a 'pedal note'.

6.3 *Fractal*

See also appendice 4 on page 147.

Fractal

A fractal is a natural phenomenon or a mathematical set that exhibits a repeating pattern that displays at every scale. It is also known as expanding symmetry or evolving symmetry. If the replication is exactly the same at every scale, it is called a self-similar pattern – as showed on figure 6.4.

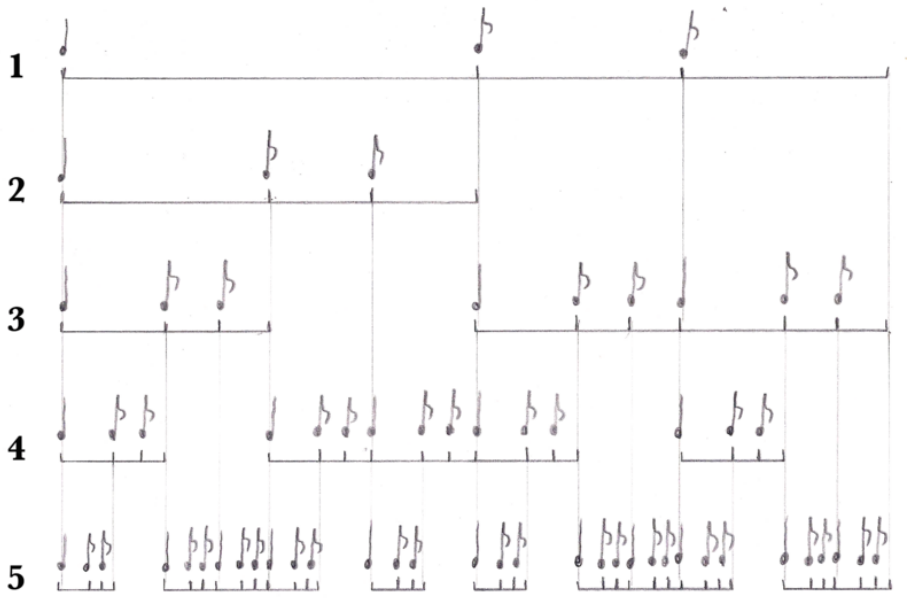


Figure 6.4: Illustration of 5 levels of fractal recursivity from the rhythm: quarter note, eighth note, eighth note. In this case, each recursivity implies the doubling of the tempo.

The fractal recursivity stops when the number of recursivities rec is reached, unless the minimal duration defined with min (meaning $min \neq$ nil) is reached before. For instance, figure 6.4 could be parametrised with $rec = 5$ or min equal to the value of the shortest eighth note recognized on level 5.

This kind of fractality describes a process from a final duration into an 'inferior' dimension according to the maximal differential duration for each recursivity.

Algorithm 4 $\sim$FRACTAL $(rtm, dur, rec, min, al \mid R, int)$

args
rtm = rhythm defined by a numerical array as a list of durations of event.
dur = total duration (mean to scale such as $\sum rtm = dur$).
rec = number of dimensions or recursivity.
min = minimal duration accepted.
al = is a list of events according to rtm. If $al = nil$ then the new events according to the level of recursion, are defined as 1, if not as 0.
args − recursive call
R = temporary result.
int = intermediate durations in relation with the number of recursivity.

$tmpR = [\,]$
if $rec = nil$ **then** $rec = 0$
if $min = nil$ **then** $min = 0$
if $R = nil$ **then** $R = [rtm.as(\sum rtm = dur)]$
if $int = nil$ **then** $int = assoc(rtm, al)$
for i in $R[0]$ **do**
  if $i = max(R[0])$ **then** $tmpR.add[rtm.as(\sum rtm = i)]$
  else $tmpR.add[[i]]$

if $rec = 0$ **or** $min \geqslant min(R[0])$ **then**
  return $[R, int]$
else
  $idr = [\,]$
  for $item$ in $R[0]$ **and** $position$ from 0 to $|R[0]| - 1$ **do**
    $idrn = [\,]$
    repeat $idrn.add[item]$
    until $|tmpR[position]|$
    $idr.add[idrn]$
  $\sim$FRACTAL$(rtm, dur, rec - 1, min, al,$
  $R.add[\bigcup tmpR], int.add[\bigcup assoc(idr, al)])$

Here are some precisions concerning the algorithm 6.4 $\sim$FRACTAL such as the arguments and the output result.

- The algorithm takes as arguments
  - → the initial rhythm `RTM`,
  - → the total duration `DUR`,
  - → and the number of recursivity `REC` or the lowest allowed duration, involving a variable number of recursivity, such as `REC = RES[i].size`, with `RES` as the resulting array,
  - → and optionally `AL` (see next point).

- The `RES[1]` is a list of events – for instance, each event can be defined by the command line *enkode* – according to the `RTM` defined by the option `AL`. If `AL = nil` then the new events according to the level of recursion, are defined as 1, if not as 0. Note the events are grouped according to the initial datasets defined by `RTM` and `AL`, which implies to apply a method to flat once `RES[1][i]` in order to get the bijection between `RES[0][i]` and `RES[1][i]`.
- For n from `0` to `REC`-1;

  $\rightarrow$ `RES[0][`n`].size = RES[1][`n`].flatten(1).size`

  $\rightarrow \sum$ `RES[0][`n`] = DUR`
- `RES[0][i].size`-1 is a multiple of `RTM.size`-1.
- `RES[0][0]` is the last recursion, as an array of all events duration of the whole sequence.

6.4 Discussion

Last but not least

Now, the art – if I may use that word – should be to establish a relationship between each element of the system as a formal object inside the morphological context predefined or emergent.
This implies a metaphysical order relationship between the formal inference system and the artistic intent.

A philosophical point

The initial material, here we are talking about a sample as a sound file or a partition, convey its own immanence according to the developmental process implementation. I do make a link between the transcendence of the process and the immanence of the initial component(s), as respectively formal and structural aspects (like the crystallization phenomenon). Thus, the result is an emergent form, which can be determinist or indeterminist in chaotic terms [Gleick, 1991] depending on the case therefore sensitive to initial conditions.

6.5 Perspectives

6.5.1 Generating data files

One possible approach to manage the algorithms described in this paper is to generate data files in order to be used as arrays according to some third-party algorithms.

Also, in this perspective, the interesting part is to work from existing music or soundscape as a sound file. Then, the sound file is analyzed with the command line *enkode* in order to generate data as a list of musical or sonic events. The analysis returns for each event the duration, $f0$ as the first significant partial, the centroid, the loudness, and the bass loudness after low pass filtering. These data can be interpreted as such, that is to say as raw data, or according to the discrimination in classes described in 1.4 on page 22.

In this sense, it is possible to apply the contrastive analysis with the dendrogram for symbolization as described previously in 4.3 on page 47 within the Common Lisp package N3 of the artificial neural network *Neuromuse3*. Note that the sound file is supposed to be in a *dataFolderPath*:

```
;; Display graph to select the optimum number of classes:
N3> (open-graph <treeName>)
;; Write the file as a structure sequence according to selected the
     number of classes:
N3> (write-file (structure-s (list (alpha-seq <somName> <treeName>
    <classesNumber>)) :result :last) :name '<fileName>.dat'
    :path '<dataFolderPath>/')
```

6.5.2 Reading data files

From this point, the data can be collected by a third-party application, for example, SuperCollider. Then, according to some analysis developed in this paper, all data can be collected into arrays in order to retrieve them by their respective indices.

Note in the case of using *enkode*, if the result is about classes, these numbers should be associated with the values of the 5 first lines of *<fileName>.info*.

```
 // if needed ------------
~arraySamples = PathName(<dataFolderPath>).loadFilesToArray
(ext: "wav");
 // ----------------------
~arrayScores = PathName(<dataFolderPath>).loadFilesToArray
(ext: "score", type: \dat, as: \integer);

~arrayStructures = PathName(<dataFolderPath>).loadFilesToArray
(ext: "dat", type: \dat, split: true);
 // ----------------------
 //  ~arraySamples[n] ---> ~arrayScores[n].size = number of events
     = ~arrayStructures[n].sum{|subAr| subAr.size}
```

6.5.3 Interpreting data files

The algorithmic interpretation of the data in SuperCollider context may require some preliminary function to select sub-sequences according to the segmentation of the contrastive analysis.

1. Select randomly a rhythm pattern for the algorithms such as *fractal* or *proportional canon* with at least `diffarg` different durations and according to the `test` function as string or symbol applied to the length of the pattern such as `"odd"` or `"even"`:

```
~rtm = RTM.new
(score: ~arrayScores[n], structure: ~arrayStructures[n],
    diffarg: 3, test: \odd, limit: 10);
```

2. Select a sub-sequence from a sound file, randomly according to minimal and maximal values (which can be used to adjust the fade in and the fade out, for a Doppler effect for instance), or depending on a given structure, randomly or according to a sub-sequence indices:

```
~rndSample = ~arraySamples[n].select(maxDur: 10, minDur: 5);

~subSample = ~arraySamples[n].selectSubStructure
(~arrayScores[n], ~arrayStructures[n])
```

Chapter 7

Sound design studies

2018 – 2021

This is some convenient *Pseudo-UGens*[1] designed in the context of this writing.

7.1 Distance

UGens>Filters

See description appendice4 on page 145.

7.2 Doppler4

UGens>Buffer

See description on page 86.

7.3 Pan4MSXY

UGens>Buffer

This *Pseudo-UGen* convert quadraphonic recording MS/XY (using the recorder Zoom H2N for instance, see page 117 for details) to a four channels equal power panner, with outputs in order LeftFront, RightFront, LeftBack, RightBack.

[1] An *UGen* – i.e. an Unit Generator – is a SuperCollider object that processes or generates sound.

Note the method `convertPan4toArray`, which distributes amplitudes in the quadraphonic space according to the panoramic positions (see description on page 90).

Pan4MSXY

Arguments:

bufMS		The index of the stereo buffer MS.
bufXY		The index of the stereo buffer XY.
dist	0	0 to +1, relative distance from the listener.
rate	1.0	speed ratio of the soundfile. 1.0 is the normal rate, 2.0 is one octave up, 0.5 is one octave down, etc.
mid	1	as level.
side	1	as level.
xy	1	as level.
xpos	0	-1 to +1, x-axis position, *i.e.* left to right.
ypos	0	-1 to +1, y-axis position, *i.e.* back to front.

7.4 Sow

`UGens>Buffer`

The idea is to apply some 'macro' formal objects to a sonic object as it is. These macro-forms are described in the previous chapters respectively as *Proportional canon as an event climax* on page 71 and *Fractal* on page 75.

Sow

Arguments:

buf	The index of the buffer.
rat	An array of ratios.
del	Position of the 'climax' within the sample duration in second. By default, this position is the absolute maximal value of the signal.

Note the array of ratios must be non-zero positive numbers.

The respective delays according to the array of ratios are computed as follows:

$$\bigcup_{r \in \mathbf{rat}} \mathbf{del}\left(\frac{1}{min(\mathbf{rat})} - \frac{1}{r}\right)$$

7.5 InH2O

`UGens>Filters`

Simple underwater filter.

InH2O

Arguments:

in		The input signal.
rdepth	0.5	The relative depth, from 0 the surface to 1 (excluded) the deepest point (inaudible).
turbulance	#[0.1, 20]	Literal array setting the frequency and the level of the modulation of noisy perturbation.
abs	-7	The curvature of the decreasing relative depth.
numChans	2	The number of output channels.

7.6 Ulam

`UGens>Generators>Deterministic`

See the description of the Collatz algorithmic process on page 101.
See also the class method `collatz` in the SuperCollider extension **cycle**.
https://github.com/yannics/cycle

For a given positive integer n, the Collatz algorithm generates a finite series which is interpreted as an amplitude envelope on the time/x-axis. The envelope is normalized on the amplitude/y-axis by dividing the envelope by n as a frequency or as a maximal value of the profile.

Each profile as an envelope is multiplied by its respective *Ugen* taking into account the value n as the frequency argument. This is done recursively through a given array of frequencies as integers.

Chapter 8

in situ

8.1 *Triptyque*

Quadraphonic installation interpreted the 19th of September, 2015 in the church of Lescouët-Gouarec in *Kreiz Breizh*.

Présentation

Triptyque se compose de trois tableaux sonores que l'on peut décrire de la façon suivante:

à propos :

a/
Lorsque l'on écoute une œuvre musicale, il y a toujours des moments saillants qui suscitent une attention ou une émotion particulière – François Nicolas, *Une écoute* à l'œuvre: *d'un moment favori dans la* Chute d'Icare [Szendy, 1999, pp. 27-45]. Ces moments ne sont pas les mêmes pour tout le monde, même si il y a consensus, soit délibéré par le compositeur, soit reconnu par des facultés cognitives communes des auditeurs.

Le concept de cette partie repose sur un choix stochastique de moments potentiels d'une œuvre donnée mis en exergue par l'effet doppler, focalisant le moment au passage au plus près de l'auditeur.

« **La vitesse tend à réduire l'espace en une ligne droite et à nous extraire de la gravité terrestre, et finalement revêt le caractère insaisissable de la divinité.** » [Marinetti, 1916]

Le décontextualisation de ces moments ainsi traités vont s'inscrire dans le temps selon une distribution stochastique de type égale pour le détermination du point émergent dans l'espace quadriphonique et les durées de silence entre deux événements – comprises entre une valeur minimal et une valeur maximal; de type décroissante linéaire concernant le zenith; et de type gaussienne concernant la distance de passage au plus près. Ceci constituera le premier mouvement de ce triptyque.

b/
La répétition, la variation et l'accumulation d'une pulsation initial va engendrer un processus de transformation du matériel ponctué par des objets percussifs distribués aléatoirement dans l'espace quadriphonique.

Ce processus de distribution est défini selon une même trajectoire d'un mouvement brownien dans l'espace quadriphonique et selon la forme musicale du canon résumé par la note de programme suivante:

« **Canon spatial dit de proportion à cent voix.** »

Progressivement, la perception que l'on aura de l'objet musical va se « désabstraire » de façon anamnésique chez l'auditeur évoquant des situations sonores familères. De plus, la précipitation événementielle induit une contraction temporelle significative. L'objet sonore prend de la vitesse; réminescence du premier tableau. Le silence qui suit acquiert de la sorte une résonance particulière dans la durée.

c/
D'un enregistrement d'une performance musicale à la synthèse pure, le concept musical peut aussi s'inscrire dans un environnement naturel immersif. Ainsi, selon un paysage sonore donné, une synthétisation – jonglant sur les concepts évoqués dans le deuxième tableau – va se créer et s'ajouter pour constituer le troisième tableau de ce triptyque.

« ***Cantus Lupus*** **versus synthétisation.** »

L'illustration sonore du film *Lupi-Les loups de Coat Fur* réalisé par Nicolas Charles en 2016 en est une réduction stéréophonique.

Doppler4

Le *Pseudo-UGen Doppler4* consiste à appliquer l'effet Doppler à une source sonore monophonique délibérée se déplaçant dans une trajectoire rectiligne de type $S(t) = mx + p$, à vitesse constante et dans le champ audible de l'auditeur. Ainsi, le point d'émergence susceptible d'être perçu ou autrement dit la distance seuil de perception est fonction de l'amplitude de la source, donc relative. Présentement, ce point est égale à 1 en coordonnée radiale polaire (voir figure 8.1).

Doppler4

Arguments:

buf		The index of the buffer to use.
xIn	1	-1 to 1, position of the emergent point in x-axis. The value zero is forbidden.
yIn	1	-1 or 1, position of the emergent point in y-axis (respectively back or front).
dist	0.1	-1 to 1, relative distance from the listener. Negative value means the source passes in back of the listener and positive value in front of the listener. The value zero is forbidden.
zenith	0	0 to 1, elevation of the source according an angle θ from the horizontal such as $z = sin\theta$.
rate	1	Ratio to scale the playback speed of the input buffer in relation with the duration in second to cross the audible space. The latest is equal to `BufDur.kr(buf) * rate.reciprocal`.

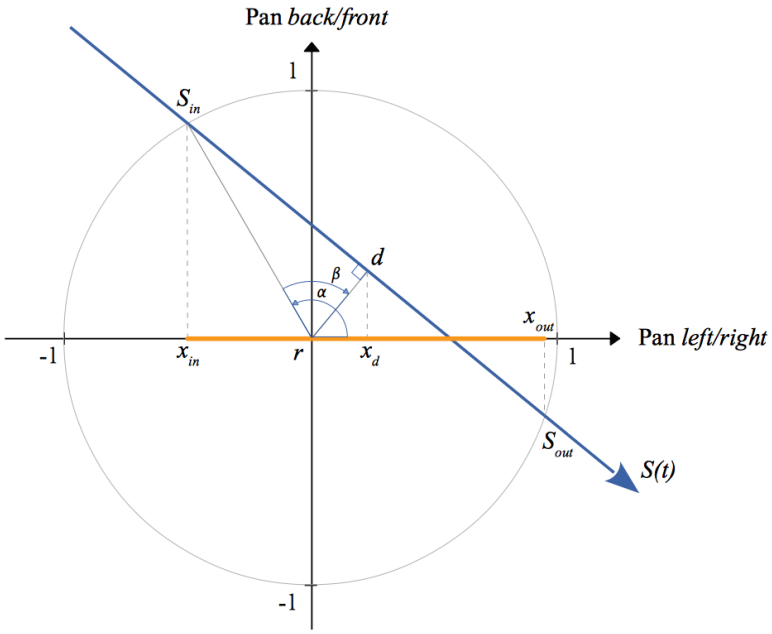


Figure 8.1: Trajectoire de l'objet sonore $S(t)$ par rapport au récepteur r.

Trajectoire de l'objet sonore

La trajectoire est calculée à partir de quatre paramètres qui vont conditionné le panoramique. Ainsi, le point émergent S_{in} est localisé sur l'axe des ordonnées par rapport à l'axe des abscisses – définit par le paramètre x_{in} – et indiqué par le signe de q ($+1$ ou -1) déterminant respectivement le panoramique *front/back* du point d'émergence. Dans le cas de la figure 8.1, la valeur de $q = +1$.

1. Coordonnées de S_{in}:
   Soit $\alpha = cos^{-1}\, x_{in}$;
   $y_{in} = q\, sin\alpha$

2. Coordonnées de d:
   Soit $\beta = cos^{-1}\, d$;

   $$x_d = \begin{cases} d\, cos(\alpha + \beta) & \text{si } x_{in} > 0 \\ d\, cos(\alpha - \beta) & \text{si } x_{in} < 0 \end{cases}$$

   Soit $\gamma = cos^{-1} \dfrac{x_d}{d} = \widehat{drx_d}$;
   $y_d = d\, q\, sin\gamma$

Pour déterminer $S(t)$, il reste à évaluer m, c'est à dire le coefficient directeur (celui-ci détermine le sens du panoramique, à savoir pour $m > 0$ de la droite vers la gauche, et $m < 0$ de la gauche vers la droite) et l'ordonnée à l'origine p (qui détermine si la source passe à notre gauche pour $p > 0$ ou à notre droite pour $p < 0$) en fonction du panoramique de départ x_{in} (point d'emergence auditive) et de la distance d de passage (au plus près) de la source au récepteur r.

3. Calcul de la pente m de $S(t)$:
   $m = \dfrac{y_{in} - y_d}{x_{in} - x_d}$

4. Calcul de l'ordonnée à l'origine p de $S(t)$:
   $p = y_{in} - m\, x_{in}$

Enfin, il reste à localiser le point de fuite définit en S_{out}.

5. Déterminer x_{out} afin de définir l'ambitus panoramique sur l'axe des abscisses:
   Soit $x^2 + y^2 = 1$ (selon le théorème de Pythagore), la substitution de y par $mx + p$ implique la résolution d'une équation du second degré de type: $ax^2 + bx + c = 0$,
   Ainsi, pour:

$$a = m^2 + 1$$
$$b = 2mp$$
$$c = p^2 - 1$$
$$x_{out} = \begin{cases} \dfrac{-b - \sqrt{\Delta}}{2a} & \text{si } x_{in} > 0 \\ \dfrac{-b + \sqrt{\Delta}}{2a} & \text{si } x_{in} < 0 \end{cases} \quad \text{avec } \Delta = b^2 - 4ac$$

Evaluation de la distance

La distance s'inscrit dans une perspective dynamique dans le champs audible en termes de filtre passe-bas (*cutoff frequency* du LPF) combiné avec l'amplitude qui sera appliquée – le cas échéant – au *drylevel* de la reverbération, en fonction de la localisation radial par rapport à r.

Niveau sonore et Filtrage

See appendice 4 on page 145.

Elevation

Dans le contexte présent, l'élévation consiste à 'rectifier' la trajectoire par rapport à l'auditeur en conservant le niveau sonore définit par la distance de passage au plus près.

Effet Doppler

La fréquence perçu par l'auditeur (récepteur immobile) est définit par la formule, $f_r = f_S \dfrac{1}{1 - \dfrac{v_s \, cos\theta}{v}}$ avec f pour la fréquence, v pour la vitesse (du son sans indice), r le récepteur et S la source, θ est l'angle que fait la direction de la source avec le récepteur.

Au même titre concernant la relativité du seuil de perception déjà évoquée pour les points S_{in} et S_{out}, le temps tps que prendra la source en terme de vitesse pour parcourir l'ambitus panoramique sera relatif.

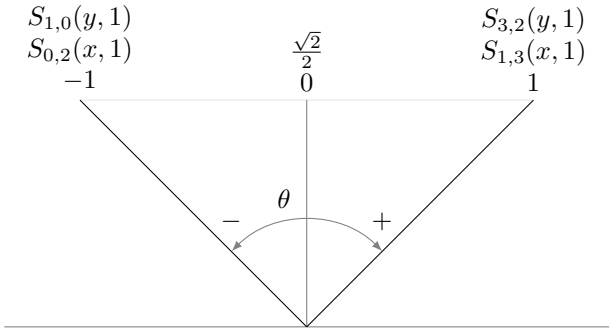
Le rapport vitesse de la source par la vitesse du son dans l'atmosphère sera par conséquent remplacé par l'inverse du temps relatif soit tps^{-1}, alors $f_r = f_S \dfrac{1}{1 - \dfrac{cos\,\theta}{tps}}$ avec θ la coordonnée angulaire de l'objet sonore à l'instant t, toujours par rapport à l'auditeur.

Configuration spatiale

À noter que dans le cadre de la performance *in situ* (voir figures 8.2 et 8.3), chaque haut-parleur est dirigé vers l'extérieur, en direction des murs et selon un angle estimé empiriquement afin d'éviter le son direct et de profiter de la première reflexion comme substitut au signal direct. Cela n'est certes pas très orthodoxe mais s'avère efficace dans la configuration *in situ* et selon le matériel à disposition – à savoir 2 monitorings Yamaha HS80M plus 2 monitorings Yamaha HS50M (ces derniers sont placés vers l'avant) et une carte son Motu ultralite mk3.

Figure 8.2: Église de Lescouët-Gouarec – Septembre 2015.

Distribution des amplitudes dans l'espace quadriphonique



Soit θ l'angle en coordonnée polaire du point $([-1,1],1)$; le retranchement par $\frac{\pi}{2}$ permet de définir un intervalle $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$.
Ainsi, la panoramisation à puissance constante est définit par [Roads, 2007, p. 134]:

$S_{1,0/0,2} = \frac{\sqrt{2}}{2} [\, cos\,\theta + sin\,\theta \,]$
$S_{3,2/1,3} = \frac{\sqrt{2}}{2} [\, cos\,\theta - sin\,\theta \,]$

Telle que:

$S_0 = S_{0,2} \times S_{1,0}$
$S_1 = S_{1,3} \times S_{1,0}$
$S_2 = S_{0,2} \times S_{3,2}$
$S_3 = S_{1,3} \times S_{3,2}$

De sorte à confirmer l'égalité suivante:

$\sum_{i=1}^{n}(S_{i-1})^2 = 1$ avec $n = 4$, soit 4 haut-parleurs.

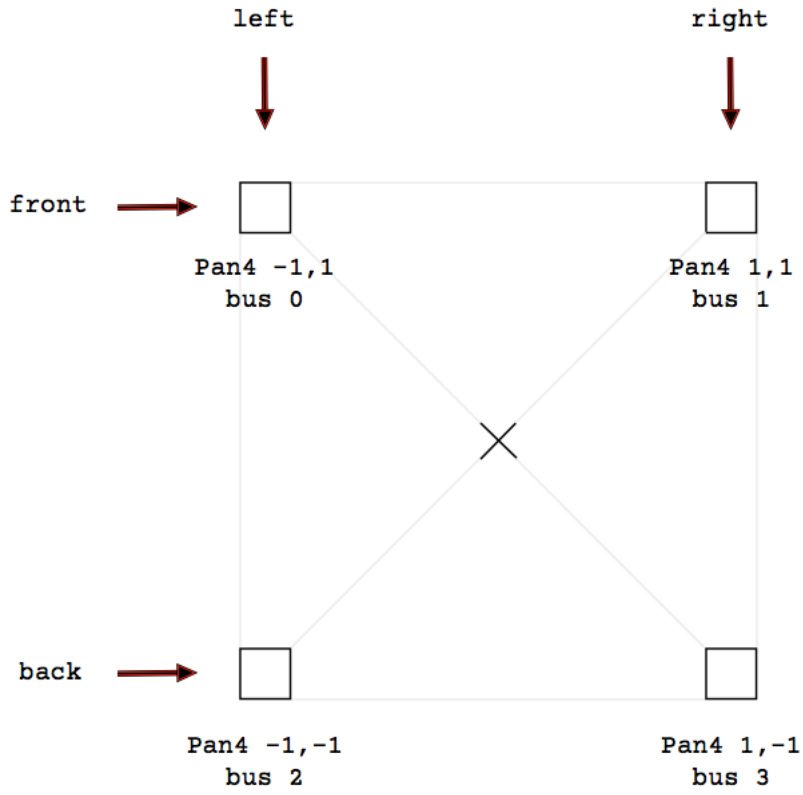
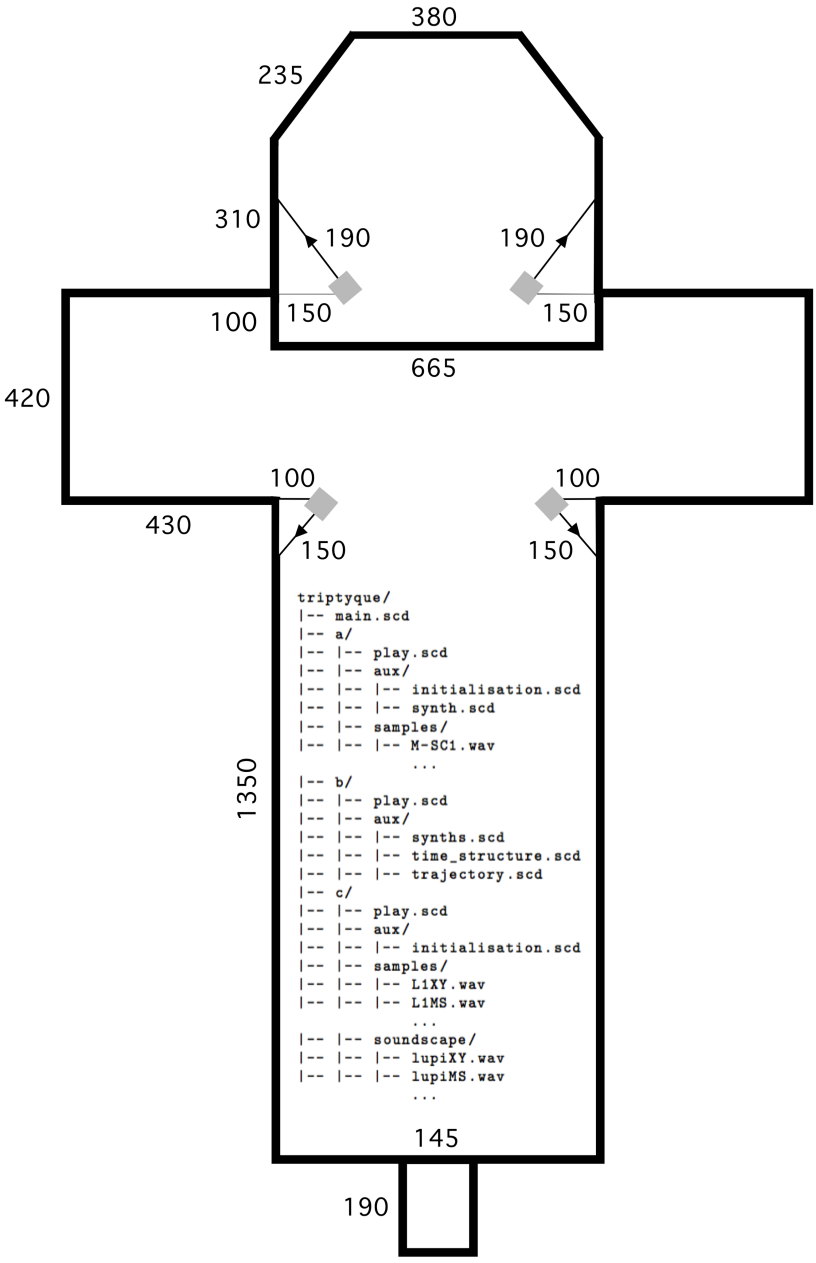


Figure 8.3: Panoramique quadriphonique de *Triptyque*, tel qu'il est interprété avec l'*UGen* `Pan4` avec les numéros de bus respectifs.

Tree-like format of *Triptyque*




```
triptyque/
|-- main.scd
|-- a/
|-- |-- play.scd
|-- |-- aux/
|-- |-- |-- initialisation.scd
|-- |-- |-- synth.scd
|-- |-- samples/
|-- |-- |-- M-SC1.wav
                ...
|-- b/
|-- |-- play.scd
|-- |-- aux/
|-- |-- |-- synths.scd
|-- |-- |-- time_structure.scd
|-- |-- |-- trajectory.scd
|-- c/
|-- |-- play.scd
|-- |-- aux/
|-- |-- |-- initialisation.scd
|-- |-- samples/
|-- |-- |-- L1XY.wav
|-- |-- |-- L1MS.wav
                ...
|-- |-- soundscape/
|-- |-- |-- lupiXY.wav
|-- |-- |-- lupiMS.wav
                ...
```

8.2 HEXO

Composition pour dispositif électronique en quadriphonie, interprétée le 25/11/16 – Salle des fêtes de Perret – *Kreiz Breizh*.

Présentation

HEXO est composée de deux mouvements distincts puisant leurs ressources dans un panel de partitions [Marcel-Dubois et al., 1939] préformatées identifiées avec l'extension *score*.

- **Mouvement I** : FRACTAL. Il s'agit de 'fractaliser' dans l'espace une partition de rythmes choisi aléatoirement – ou pas – parmi le panel précédemment évoqué. L'algorithme est appliqué indépendamment pour chaque phrase dans un rapport de durée proportionnel.
- **Mouvement II** : ARIA. Airs choisi parmi le panel de partitions traduit par un filtre résonant appliqué à une impulsion dans l'espace et une banque de fichiers sons dans un déroulement temporel en termes de grains et de mixage entre les partitions et les samples.

Description

a/ Score

```
24 12 12 24 6 18                    phrase 1
57 58 60 62 62 65
24 8 8 20 12+rrand(3,12)            phrase 2
62 63 60 60 62
36 18 6 20 8 8                      phrase 3
62 63 62 60 62 63
9 3 12 24 24                        phrase 4
62 60 58 60 0
12 24 24 6 12 8 8 8                 phrase 5
60 62 63 62 60 60 62 63
28 4 4 12 24                        phrase 6
62 60 59 58 0
2 84 24 9
```

La première ligne représente les durées en nombre entier selon une valeur minimal irréductible appliqué à toute la partition.
La deuxième ligne représente les notes en format midi.
Le `n+rrand(a,b)` correspond à un point d'orgue compris entre `n+a` et `n+b` selon la valeur minimal défini.
Le zéro correspond à un silence.

Figure 8.4: Description du fichier *score*. La dernière ligne est réservée au nombre de dimension par événement et par groupe de lignes, et respectivement au tempo + le nombre de valeurs irréductibles associés au tempo + le nombre de répétitions de la partition.

`HEXO` est conditionné par un corpus de fichiers partitions permettant de paramétrer l'ensemble de l'œuvre. Ces fichiers doivent être formater selon la syntaxe décrite à la figure 8.4 – voir aussi 3.2.1 page 40.

b/ Mixage

Dans le deuxième mouvement, le mixage entre le chant et le 'grain' est assuré par modulation de fréquence selon 3 arguments; la durée minimale a entre 2 événements, la durée maximale b entre 2 événements et le niveau c d'écrêtage.

La modulation de fréquence est effectuée sur une onde porteuse sinusoïdale et une fréquence de modulation selon une distribution stochastique linéaire appelée *gendy*[1] avec pour argument la moyenne des durées – soit $(a+b)/2$ – convertie en *hertz* – soit $m = 2/(a+b)$.

Cette modulation est ensuite transposée afin d'osciller selon une fréquence définie par les durées minimale et maximale prédéfinies. Ces durées sont converti en *hertz* – soit respectivement $1/a$ et $1/b$. La fréquence porteuse doit alors être multipliée par $x = (b+a)/ab$ auquel résultat on ajoute $y = (b-a)/ab$.

L'écrêtage c permet des paliers de durée continu pour chaque événement. Celui-ci est normalisé en multipliant le résultat par $1/c$ – voir figure 8.6.

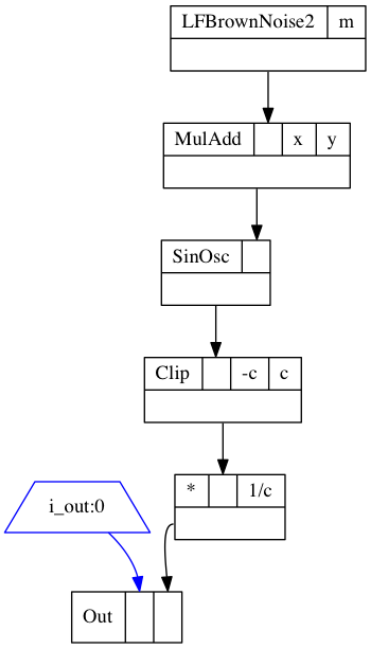


Figure 8.5: Mixage par modulation de fréquence.

Notes

L'*UGen* fait parti des éléments de base permettant la génération et le traitement du signal en termes de computation. L'*UGen* `SplayAz` permet en l'occurrence de diffuser un signal à travers un réseau de canaux audio.

Concernant l'*UGen* `SplayAz`, deux arguments optionnels méritent d'être souligner ici : *spread* et plus particulièrement *center*. Le premier diffuse plus ou moins quantitativement et respectivement de 1 à 0 le signal autour de la valeur du second. Le dernier admet une valeur allant de -1 à $+1$ traduit par une localisation circulaire impliquant $-1 \Leftrightarrow +1$.

[1] *Gendy is an implementation of the dynamic stochastic synthesis generator conceived by Iannis Xenakis and described in his book* Formalized Music [Xenakis, 1992, chapter 9 pp. 246-254 and chapters 13 and 14 pp. 289-322].

La différence de configuration des canaux, notamment entre l'*UGen* `Pan4` et l'*UGen* `SplayAz` implique une correction consistant à inverser les canaux 2 et 3 avec la méthode *swap* appliquée directement sur l'*UGen* concerné, étant lui-même une liste de canaux ordonnés.

Par exemple, si l'installation quadriphonique est correct avec l'*UGen* `Pan4`, alors la correction se fait sur l'*UGen* `SplayAz` de la façon suivante:

```
SplayAz.ar(4, in).swap(2,3)
```

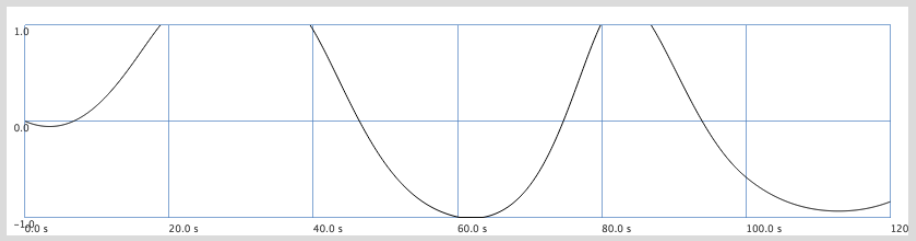


Figure 8.6: Mixage oscillant entre 2 signals avec ses plages écrêtées à 0.8 normalisées entre +1 et −1 sur une durée de 2 minutes. Le paramétrage est de 10 secondes pour la durée minimale et de 30 secondes pour la durée maximale.

8.3 K540

8.3.1 Version I

Composition pour dispositif électronique en quadriphonie, interprétée le 29/01/18 – *Kulturhuset Hausmania* – Oslo.

Présentation

`K540` se compose de trois mouvements autour de la partition Kjølhiea (voir appendice 4 on page 205).

- **Mouvement I** : Harmony. Kjølhiea est interprétée selon une série – harmonique ascendante ou harmonique descendante – liée à une fréquence fondamentale donnée, en terme de résonance selon le mode de transition *peak-morphing* (voir algorithme 2).

  Réf. Ce type de synthèse s'apparente dans une certaine mesure à une complexe combinaison de glissandi dit de Shepard-Risset. De plus, j'aime l'idée que les glissandi se réfèrent aux sirénes chères à Edgar Varèse.

- **Mouvement II** : HARMONIC. Kjølhiea est interprétée selon une série – harmonique ascendante ou harmonique descendante – liée à une fréquence fondamentale donnée, par signaux sinusoïdaux selon un mode de transition *cross-fading*.

  Réf. *You will be hearing a sound. Just allow your body to relax deeply into the sound and learn what it has to teach you.* Tuning forks [Beaulieu, 1987, pp. 89–99].

  Dans ce modèle, l'ambitus harmonique et leurs superpositions en 'accords' favorisent l'émergence de fréquences différentielles en tant que sons résultants mais aussi en termes de pulsation et d'infra-sons, permettant ainsi d'interagir en termes de bienfaits thérapeutiques – relaxation et concentration – sur l'ensemble du corps incluant les ondes cérébrales [Morlot, 2012] par effet de résonance.

- **Mouvement III** : *Echo*. Kjølhiea est interprétée selon l'*histogram* de la partition déterminant le profil des accords. La mélodie est découpé de façon aléatoire afin de créer plusieurs échos sur l'impact du premier événement de chaque segment.

  Réf. L'écho est un rebond dans l'espace 'caverneux' – se référant volontier aux origines de la musique remontant au moins à l'age de l'art pariétal (soit environ 35 000 BC) et de la philosophie ontologique – en terme de répétitions et de variations.

Description

a/ Conversion du fichier midi

Dans le cas présent, Kjølhiea est un fichier midi dont les 4 voix ont été combinées sur une portée et réinterprétée en tand que partition MDS avec un *histogram* calculé en arrondissant la somme des durées différentielles d'une note donnée divisée par le nombre d'occurrences de cette même note.

b/ Transposition des accords en profil harmonique

Le profil harmonique est calculé pour chaque note[2] formant l'accord de façon à ne retenir que la plus grande valeur pour chaque harmonique d'une fréquence fondamentale donnée.

Détails de la procédure de transposition:

1. Créer une série harmonique de référence.
   Les arguments sont le nombre d'harmoniques, la fréquence fondamentale et s'il s'agit d'une série ascendante ou descendante.

[2] *Each note of the score is defined by the harmonic series of a given root frequency. This is done by selecting recursively the harmonic range of the nearest harmonic according the modulo 12 as midi note, or in other words the degree of the note according the note C equal zero as root.*

```
if(desc,
     {serRoot=Array.fill(nharms,{|i| freqroot/(i+1)})},
     {serRoot=Array.series(nharms,freqroot,freqroot)});
```

2. Créer une série harmonique pour une note donnée. Les arguments sont le nombre d'harmoniques retenues défini par le ~`spread`, la note midi et s'il s'agit d'une série ascendante ou descendante.

```
if(desc,
    {serNote=Array.fill(~spread,{|i| note.midicps/(i+1)})},
    {serNote=Array.series(~spread, note.midicps,
        note.midicps)});
```

3. Puis de manière récursive à travers le résultat précédent.

```
serNote.do{ |hr|
   // 3.1
   serDiff= serRoot.collect({|it,i|
       (it.cpsmidi.mod(12)-hr.cpsmidi.mod(12)).abs});
   // 3.2
   arIn= serDiff.indicesOfEqual(serDiff.minItem);
   // 3.3.
   tmp= ~getNearestHarm.value(hr,arIn,serRoot,profil);
   if(tmp.isNil.not, {profil=profil.add(tmp)});
  };
```

   3.1. Lister les différences en midi et en modulo 12 entre chaque harmonique d'une note donnée et les harmoniques de la fréquence fondamentale.

   3.2. Lister les indices de la valeur minimal du résultat précédent.

   3.3. La fonction ~`getNearestHarm` sélectionne l'indice – si il existe – le plus près de la fréquence réelle et n'appartenant pas encore au profil.

```
~getNearestHarm = {
  |note, arIn, arRoot, arRef|
  var res, indexOrder, out;
  // 3.3.1.
  indexOrder=arIn.collect({|it|
      (note-arRoot[it]).abs}).order;
  res=Array.fill(arIn.size,0);
  indexOrder.do({|it,i| res=res.put(it,arIn[i])});
  // 3.3.2.
  res.do({|item|
    if(arRef.includes(item),
        {out},
        {out=out.add(item)})});
  out.first;
 };
```

3.3.1. Ordonner les indices selon la proximité de leur fréquence réelle avec la fréquence fondamentale.

3.3.2. Ajouter l'indice dont la fréquence est la plus proche dans le profil harmonique – si l'indice n'est pas déjà dans le profil.

4. Le résultat est une liste ordonnée d'indices se référent à la série harmonique de la fréquence fondamentale.

Le résultat précédemment décrit s'inscrit dans le profil harmonique d'un accord selon un profil d'intensité de référence lié à une note (avec un cardinal de la valeur du ∼`spread`). Cela induit, le cas échéant, de ne retenir que la plus grande valeur d'intensité pour un harmonique donné.

De la sorte, un accord est défini par une liste d'indices harmoniques associés à leur respective intensité.

c/ Profil d'amplitudes des accords

Le profil d'amplitudes consiste à normaliser la somme des poids de chaque note de l'accord définie par l'*histogram* (voir ***a/***) avec la fonction ∼`recArWeight`. Ainsi, le profil des accords pour chaque cycle est réalisé aléatoirement entre cette normalisation, l'inverse de cette normalisation et un mélange des deux.

d/ Profil échoïques segmentaire

Pour un thème mélodico-harmonique donnée (en l'occurence Kjølhiea, le profil échoïques est réalisé – selon une segmentarisation du thème dont le nombre d'éléments est compris entre deux valeurs données (voir paramétrage) – avec la fonction ∼`rtmEcho` comme suit:

```
Array.geom(rtm.size, 1, 1.618).reverse.normalize(∼farest,0)
```

1.618 est le nombre d'or Φ et constitue la raison de la série géométrique de cardinal le nombre d'éléments constituant le segment considéré (ce qui se rapproche de la série de Fibonacci). La normalisation se fait en inversant la série du point le plus éloigné au point le plus près.

Paramétrage

`[1]` `[2]` `[3]`

∼***partDur*** – durée des parties `[1]`, `[2]` (affecte les durées des accords) et `[3]` (affecte le nombre de cycle mélodico-harmonique avec le paramétre ∼***tempoDivisionnel***).

`[1]` `[2]`

∼***spread*** – nombre d'harmoniques retenu pour chaque note (ceux-ci s'inscrivent dans un profil d'amplitude défini par ∼***ampSer***). Ceci implique que pour un accord donné, le nombre d'harmonique est compris entre ∼*spread* et ∼*spread* × *chord.length*.

~freqRoot – fréquence fondamentale définissant la série de ***~nharms*** harmoniques ascendant ou descendant (défini par ***~desc***, respectivement *false* ou *true*) – voir fonction *~midinote2harm*.

`[3]`

~farest – valeur de l'écho le plus éloigné (valeur comprise entre 0 et 1, respectivement près et loin). Le nombre d'écho est compris aléatoirement entre ***~minVal*** et ***~maxVal***.

Spatialisation

Mvt `[1]`

Chaque *glissandi* et *sustained note* est distribué individuellement et aléatoirement à une seule enceinte de l'espace quadriphonique.

Mvt `[2]`

Distribution égale de chaque harmonique dans l'espace quadriphonique (coordonnée x,y).

Mvt `[3]`

Distribution gaussienne de l'accord en fonction de la valeur de l'écho le plus éloigné moins sa distance effective à l'instant t.

```
Pgauss(~farest-Pkey(\dist),0.25,inf)
```

8.3.2 Version II

Composition for quadraphonic installation, interpreted the 7th and the 8th of June at the *UddeboFestidalen 2019* in Sweden.

Concept

`K540 v.II` echoes the first version of this work and the previous work `HEX0`. Indeed, it is about mixing by frequency modulation mentioned in `HEX0` between the movements I and II – respectively *Harmony* and *Harmonic* – of `K540 v.I`, reinterpreted for the circumstance.

That is to say, the sequence is generated at random as a brownian walk in term of presence as amplitude level correlated to the distance.

Also, each sound as a sine wave or as a 'peak morphing' assumes the quadraphonic space as a moving object according to the Bell distribution regarding the space to avoid, in this case the middle of the quadraphonic space.

`K540 v.II` resumes the synthesis of *Triptyque* parts ***b/*** and ***c/*** as expanded material in rhythmic terms.

8.3.3 Version III

Composition for quadraphonic installation, interpreted the 27th and 28th of July, at the *Trans' Festival* 2019 in Norway.

Concept

`K540 v.III` is a mix between the mix of `K540 v.II` and the mix of *Electronic background sketch* (6.1 on page 68) with its *octava bassa* according to Kjølhiea.

8.4 `data-01`

Composition for quadraphonic installation, interpreted on the 17th of June, 2019 at NOTAM in Oslo.

Presentation

The composition `data-01` is a direct application of the work described in the chapter *Analytical modeling* applied to a given sample defined as `data-01` and it is composed of 3 parts as layers.

`Can you imagine ...`

- **Part I** : Drop water as a Markov chain walk.

  `... before the language ...`

  Interpreted as a continuum background, part I is a Markov chain applied to the *Symbolisation* as *Hierarchical clustering* and the *Systemic analysis* as *Developmental process* of the referent sample.

- **Part II** : Collatz synthesis as a proportional canon.

  `... and beyond the language ...`

  Triggered process, part II is a proportional canon of a chosen rhythm (see figure 8.8) defined by the *Contrastive analysis* as *Segmentation by marker* of the sample. The Collatz algorithm as a normalized profile controls the bandwidth ratio of the *UGen* `Resonz` applied to a pink noise.

- **Part III** : Fractal on Radio.

  `... and into the language.`

  Triggered process, part III is a 'fractalization' of the previously defined rhythm as an alternately spatialized streaming radio.

Synthesis

Collatz resonance

This synthesis requires a Collatz sequence defined as follows:

Collatz sequence

The Collatz sequence is a sequence of numbers relevant to the Collatz conjecture, which theorizes that any number using this algorithm will eventually be reduced to 1 or stop on the trivial cycle `(4 2 1)`.
Note that in this work, according to the range of numbers we might use, we are not concerned about this conjecture.
So, to get a Collatz sequence from any integer n superior to 1, each step depends of the previous such as if n_i is even, divide n_i by two to get n_{i+1}, and if n_i is odd, multiply n_i by three and add one to get n_{i+1}, until $n_{i+1} = 1$.

The Collatz sequence is interpreted as a normalized profile – see figure 8.7 – envelope which is defined by its `duration` as the length of the sequence according to a value of `dx` in second. Note that the value zero is added at the end of the sequence.

```
Env.new(n.collatz.add(0).normalize, Array.fill(n.collatz.size, dx))
// Env.collatz(n, duration, 0, \max) // i.e. class method of cycle
```

This envelope is currently interpreted musically according to the value of the bandwidth ratio rq (bandwidth/centerFreq) of a resonant filter. Then, this normalized envelope is set dynamically rq with the initial value of the Collatz sequence as the resonant frequency applied to a pink noise.

Algorithm 5 $\sim$COLLATZ $(n \mid r)$

arg
    n = integer
arg – **recursive call**
    r = step result

if n is even **then**
    $x = n/2$
else
    $x = 3n + 1$

if $x \in r$ **then**
    return r
else
    $\sim$COLLATZ$(x, r.add[x])$

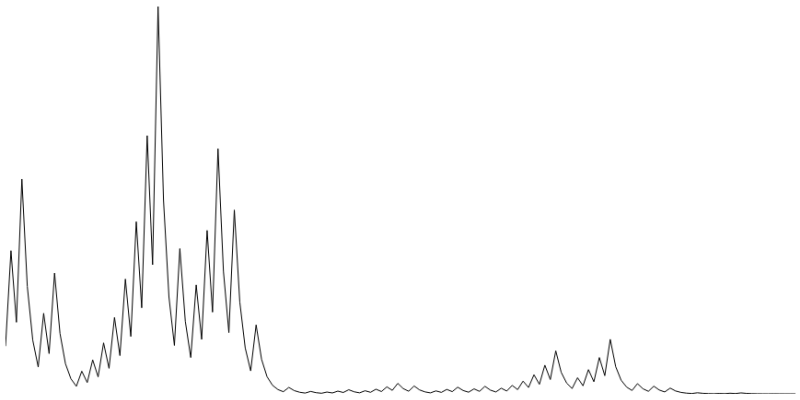


Figure 8.7: Profile of the Collatz sequence started with the number 8083.

Description

Markov chain walk instantiation

According to the Markov chain walk described in the section *4.7* p. 52, the values of the list to send – done with the discriminative classification analysis of *enkode* – according to durations as triggers *in time*, are respectively the loudness as distance, the centroid and $f0$ as the frequencies drop variation interpreted as millimeter unit (see table p. 103).

```
;; Common Lisp Markov chain initialisation

N3> (defvar *DATA* (remove-duplicates (read-file "aaa.score")
   :test #'equalp))
N3> (create-mlt 'aaa (length (car *DATA*)) (length *DATA*)
   :carte #'rnd-map)
N3> (loop for neuron in (neurons-list aaa)
   for dat in *DATA* do (setf (output neuron) dat))
N3> (dendrogram aaa 3 :and-data t)
-6031.028+
N3> (open-graph *tree*)
N3> (defparameter *alpha-seq-9* (alpha-seq aaa *tree* 9))
*ALPHA-SEQ-9*
N3> (write-file (structure-s (list *alpha-seq-9*) :result :last)
    :path "aaa.dat")
N3> (mk-alraw *alpha-seq-9* (read-file "aaa.raw"))
    min           max
0:  0.35861862    4.7338142
1:  1.8155038     4.197111
2:  260.4863      1782.7802
3:  1.6           1.6
4:  213.9862      2075.2625
```

```
;; the values to send are defined as follows:
(let ((nrand (random (length (car (getraw nep))))))
   (send-udp (read-from-string
      (format nil "(\"/~A\" ~{\"~S\"~})"
         'N3
         (list
            (nth nrand (nth 1 (getraw nep))) ;f0
            (nth nrand (nth 2 (getraw nep))) ;centroid
            (nth nrand (nth 3 (getraw nep))) ;loudness
          )))
      (string "127.0.0.1") 7771)
   (sleep (nth nrand (nth 0 (getraw nep)))))
```

Then, the incoming OSC message is interpreted in SuperCollider as follows:

Analysis	*Parameter in SC*	*Computation*
duration	OSC trigger	none
loudness	`dist`ance	`.linlin(`*min*`, `*max*`, 0.6, 0.1)`
centroid, $f0$	`rad`ius 1	`rrand(`*centroid*`, `$f0$`)` `.linexp(`*min*`, `*max*`, 4, 1)`

For convenience, the data needed for the Markov chain are stored in a *lisp* file. This concerns the `alpha-seq` according to the retained number of classes as `*alpha-seq-9*` and the hash table `*alraw*` which associates the alpha symbol to all corresponding raw values.

Proportional canon instantiation

1. Initiate some preliminary variables.

   ~rtm – the selection of the rhythm **~rtm** is done inside the panel of rhythm as indices according to the structure discrimination. The choice is made among rhythms as proportional graphic notation in a *pdf* file (see figure 8.8).

   The latest is generated as follows:

```
N3> (load (concatenate 'string *NEUROMUSE3-DIRECTORY*
      "opt/lilypond.lisp"))
N3> (defparameter *seq-as-dur*
  (loop for i in
    (group-list (read-file "aaa.score")
      (loop for i in (read-file "aaa.dat") collect
        (car (mapcar #'(lambda (x)
          (length (coerce (string x) 'list))) i))))
    collect
      (loop for dur in (car (mat-trans i)) collect
        (round (* 1000 dur)))))
*SEQ-AS-DUR*
N3> (write-rtm-seq *seq-as-dur*
      :path "aaa.ly"
      :title "data-01")
```

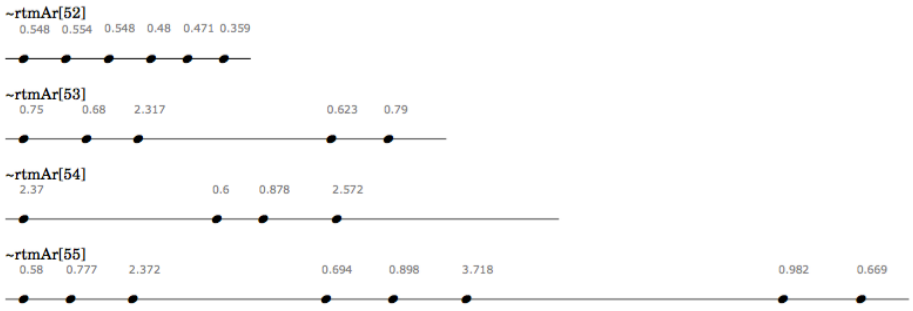


Figure 8.8: Extract of the list of rhythms generated with Lilypond as a *pdf* file.

Then compile the Lilypond *aaa.ly* file.

~repeat – number o repetition of **~rtm**.

~duration – duration of the proportional canon.

~ratio – currently the golden number.

~numberOfVoices – currently 4 voices.

2. Compute durations and delays.

```
~rrr = ~proportionalCanon.value(~numberOfVoices,~duration,
        ~ratio,1);
```

3. Display for each event of each voice the timing, the associated score event, and the voice number.

```
~rrr = ~rrr.collect({|it,i|
        var a = (~rtm.flop[0].wrapExtend(~rtm.size*~repeat)
                .normalizeSum*it.first).addFirst(0);
        var b = a.copyToEnd(1);
        a.pop;
        [
                a.integrate+it.last,
                b,
                ~rtm.wrapExtend(~rtm.size*~repeat),
                Array.fill(~rtm.size*~repeat, i+1)
        ].flop
});
```

4. Sort according to the timing for all voices.

```
~rrr = ~rrr.flatten(1).sort({ arg a, b; a[0] < b[0]});
```

5. Add duration between two successive events.

```
~rrr = [~rrr.flop[0].differentiate, ~rrr].flop;
```

To complete the last point, each event `i` as element of ∼`rrr` is defined by the structure `[a, [b, c, [d, e, f, g, h], i]]` corresponding to:

`a` → duration from the previous event until this event, all voices combined (done in point *5.*) ⇒ ∼`rrr[i][0]`;

`b` → timeline of onset according to the total duration;

`c` → duration of the event until the next event for this voice;

`d` → initial duration of ∼`rtm`;

`e` → $f0$ ⇒ ∼`rrr[i][1][2][1]`;

`f` → centroid ⇒ ∼`rrr[i][1][2][2]`;

`g` → loudness;

`h` → loudbass;

`i` → voice number.

Note that the array `[d, e, f, g, h]` defines the event's parameters, which is computed in this case with the command line *enkode*.

8.5 *untitled* #1

Composition for quadraphonic installation, interpreted the 17th of December, 2019 at NOTAM in Oslo.

Concept

This composition is an adaptation of the spatial and proportional canon of the part ***b/*** of *Triptyque* with the Risset's bells synthesis – used in the movement I of `HEX0` – as the bass drums, followed by an *Electronic background sketch* (6.1 on page 68) as a *Postlude*.

Description

The data used in this performance – that is frequencies and *Melody to Tone* analysis – was generated from the quotation find in the book *Understanding Media*, [McLuhan, 1994] p. 351, according to the following procedure.

First, in the terminal:

```
# each sentences is written in a separate file
# then the number of sentences is interpreted as a number of
     voices in order to make interesting combination done
    with ~OCWR and symmetric permutation
echo "Any process that approaches instant interrelation of a
     total field tends to raise itself to the level of
    conscious awareness, so that computers seem to think." >
    1.txt
echo "In fact, they are highly specialized at present, and
    quite lacking in the full process of interrelation that
    makes for consciousness." > 2.txt
echo "Obviously, they can be made to simulate the process of
     consciousness, just as our electric global networks now
    begin to simulate the condition of our central nervous
    system." > 3.txt
echo "But a conscious computer would still be one that was
    an extension of our consciousness." > 4.txt

# voice synthesis as aiff file
for text in *.txt; do say -o "${f%.txt}.aiff"; done
# convert to wav
for f in *.aiff; do ffmpeg -i "$f" "${f%.aiff}.wav"; done
# save spectrum analysis in the current directory
for f in *.wav; do enkode --spectrum -o './' "$f"
   2>/dev/null; done
```

```
# generate formantic coll file
mkdir wavFilesFolder
mv *.wav ./wavFilesFolder
./formantAnalysis.sh ./wavFilesFolder/
```

Then, with the M2T library:

```
(dotimes (i 4) (M2T::m2tab (M2T::sort-melody :spectrum
   (read-file (format nil "a~S.spectrum" (1+ i))) :partial 5)
   (format nil "a~S" (1+ i))) (format t "~%-----------------"))
```

To finish, in the SuperCollider context:

```
// read and write M2T files in an array
 ~profiles = PathName("".resolveRelative) +/+ "src")
 .loadFilesToArray("m2t", \dat, \collectFloat);
// get the n first items until 5000 hz
~profiles = ~profiles.collect{|mt| mt.copyRange(0,
               mt.asList.detectIndex{|item| item[0]==5000}-1)};
```

8.6 105A1408

Di-quadraphonic – post quarantine – installation, interpreted the 26th and the 27th of June, 2020 in Kråkstad.

Presentation/Concept

The performance requires four suspended 'prepared' guitars[3] with vibrating speakers inside the guitar boxes fixed as sound post of violins, inside a quadraphonic space using then four 'normal' speakers – see figures 8.9 (b) and 8.10.

This composition plays on three layers, which are mixed and interpreted as an improvisation through the SuperCollider GUI – see figure 8.11. Hence, the duration is *ad libitum* and should be managed as an installation, playing currently the 'birdscape' in guitars as a 'standby' state.

The mix concerns the layers as such and the balance between the guitars and the speakers.

(a) (b)

Figure 8.9: (a) *Sverm-resonans*, installation at the Ultima Contemporary Music Festival in 2017 by Alexander Refsum Jensenius. (b) `105A1208` installation in the shed – Kråstad, June 2020.

[3]Inspired by the work of Alexander Refsum Jensenius – see figure 8.9 (a).
https://www.arj.no/2017/09/11/sverm-resonans/

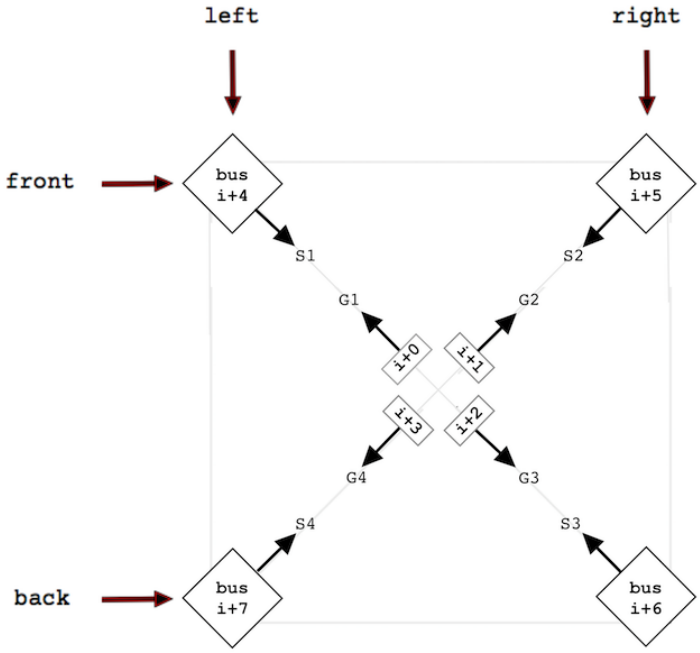


Figure 8.10: 'Di-quadraphonic' disposal in the shed – `S` → Speaker, `G` → Guitar.

- **Layer I** : 'Birdscape'
- **Layer II** : Fractal → on the rhythm analysis of the 'birdscape' – involved the command line *enkode* (→ see Chapter 1) for the analysis as initial data, the Common Lisp libraries N3 (function `differential-vector` → see Chapter 5) and *cl-mst* (functions `rhizome-degree` and `boruvka` → see Section 4.9) – as MDS → see Chapter 3.
- **Layer III** : Formantic Wind

Description

a/ MDS

Generating rhythmic patterns as Musical Data Score from the 'birdscape' analysis.

1. *enkode* analysis

```
enkode -I 3 birdscape.wav > data.txt
```

2. generating MDS – require the packages N3 and *cl-mst*

```
(defparameter *DATA* (mapcar #'N3::read-file "data.txt"))

;; For each event only its duration and the centroid is
    retained.
(defparameter *SEQ* (loop for i in *DATA* collect
   (list (car i) (caddr i))))

(defvar *N* nil) ;; as number of events

;; search pattern of n (*N*) events in seq (*seq*)
(defun loop-rtm (seq n)
  (loop for i from 0 to (- (length seq) n) by n
     collect (subseq seq i (+ i n))))

(dotimes (n 5)       ;; 5 times ...
  (setf *N* (+ 5 n)) ;; ... from 5 (then to 9)
  ;; write mds files
  (N3::>data-file (format nil "mds/105A1408-~S.mds" *N*)
    (let ((dat (loop for i from 0 to (1- *N*)
           collect
        (let* ((mds (N3::mat-trans (nthcdr i *SEQ*)))
               (tmp (mapcar #'list
                            (loop-rtm (car mds) *N*)
                            (loop-rtm (cadr mds) *N*)))
               (matrix (N3::differential-vector tmp nil
                                :result :diff-x))
               (mst (cl-mst:boruvka (mapcar #'(lambda (x)
                              (list (car x) (cadr x)
                              ;; avoid zero distance
                                (+ 0.01 (caddr x))))
                                          matrix)))
              (len (1- (length tmp)))
              (res (loop for iii from 0 to len
                   collect (cl-mst:rhizome-degree iii mst)))
              (maxi (car (N3::ordinate res '>))))
    (list maxi (nth (+ i (position maxi res)) tmp))))))
        (append (loop for i in dat append (cadr i))
                ;; last line such as 2 for grouping mds
                ;; plus the related degrees of each grouping
                ;; as weigths
                (list (cons 2 (mapcar #'car dat)))))))
```

In this case, we only compare incrementally a pattern of `*N*` events sequentially – that is to say every `*N*` events from the initial pattern – within the remainder of the sequence with the function `loop-rtm`.

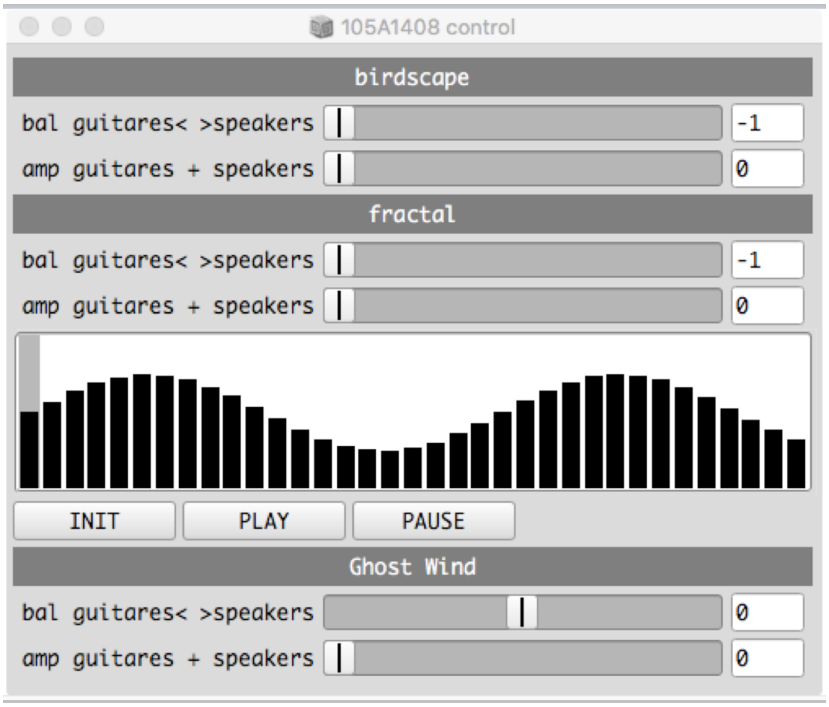


Figure 8.11: Playing and mixing *in situ* with the SuperCollider GUI.

b/ Formantic wind

	source (Hz)	range (Hz)	scale
formant 1 (f1)	312 - 750	300 - 750	lin
bandwidth f1	90	45 - 180	exp
formant 2 (f2)	781 - 2031	750 - 2050	lin
bandwidth f2	110	55 - 220	exp
formant 3 (f3)	2469 - 3187	2450 - 3200	lin
bandwidth f3	170	85 - 340	exp

Table 8.1: Formant frequency ranges with their respective bandwidths, interpreted as an exponential range between half and the double of the initial value. The column 'source' refers to the data in [Cheveigné, 1999].

Part III

MUSIC PROGRAMMING ... ABOUT MUSIC VESTIGES

Liminaire

With the advent of the internet, sharing and access to knowledge have opened a new world of communication toward I hope a new real world. Here is my contribution as a composer of what I call 'conceptual music'.

Conceptual Music consists to highlight and formulate a nodal point of acquired concepts that can be understood in ontological terms. In essence, a concept is the formalization of structuring object(s) whose emerging phenomenology is either deterministic or indeterministic, and identified as such. My purpose is to point out some nodal points notably through the programming language paradigm as for instance a sketch, a description or a process. This is naturally a work in progress, and the philosophical interpretation of the concept as such remains to be done.

Or, to put it another way, the definition of the concept in a musical context, consists of pinning down the emergent phenomenon as a nodal point of structuring elements. The emergent phenomenon is the sonic object itself that I produce to question or integrate my environment as a significative nodal point.

The application – as performances or as scores for instance – of the said concepts aim to stimulate the imagination and possibly even the creativity of the listener, notably by anamnesis in cognitive terms.

Note: Some codes are stamped `[private]` because they are too messy to be shared. They will be re-writed and rationalised to be so.

Works

Ghost Wind

Concept

Wind synthesis correlated to a spoken word recording using pitch analysis and formantic analysis with a degree of shape smoothing.

Context

Study in the framework of the acousmatic tale *The Robot And The Baby* conducted by Frédéric Voisin in 2012.
→ http://www.fredvoisin.com/spip.php?article215

Required

BASH
→ https://www.gnu.org/software/bash

PRAAT
→ http://www.fon.hum.uva.nl/praat

SBCL
→ http://www.sbcl.org

Source

→ https://github.com/yannics/GSA/tree/master/SC/ghost-wind

Alternative

The *UGen* `LPCAnalyzer` with noise as the source – sounding windy in that way – uses the linear predictive coding analysis [Makhoul, 1975] on the input signal, but in this case, the formantic part is eluded because it works only as frequency bandwidths of which the size is determined by the parameter `noise`.

```
SynthDef(\LPC, {
  | outBus=0, inBus=0, amp=0, noise=256,
    xpos=0, ypos=0 |
  Out.ar(outBus,
    Pan4.ar(
      LPCAnalyzer.ar(
        In.ar(inBus, 1),
        PinkNoise.ar(0.25),
        1024,
        noise),
      xpos, ypos, amp))
}).add;
```

RM236

Concept

Rhythmic counterpoint – between 5 complementary rhythms – for 4 voices and 3 layers, with one as tuning radio sound effects, one as far low bass drums, and some kind of high frequencies 'sparkles'.

Context

Participation of the *Lake Radio* open call – `Works for Radio #4` – in 2020.
→ http://thelakeradio.com/call

Source

→ https://github.com/yannics/GSA/tree/master/SC/RM236

Selenes Havbrev

Concept

Interactive quadraphonic soundscape using Open Sound Control over WIFI with the modular control surface TouchOSC.

Context

See booklet *Blåstjernehav Familie- & Dukketeater* on page 181.
See also article in *HAKAPIC, et nettmagasin for kunstkritikk*.
→ https://www.hakapik.no/home/2020/10/8/levende-kunst-i-en-bygning-i-forfall

Required

FFmpeg
→ https://ffmpeg.org/

SOX
→ http://sox.sourceforge.net

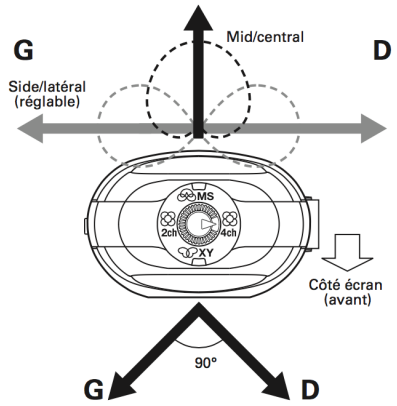
TouchOSC
→ https://hexler.net/products/touchosc

Source

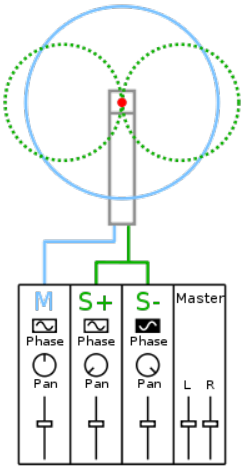
→ https://github.com/yannics/GSA/tree/master/SC/SelenesHavbrev
– [private]

Notes

1. Quadraphonic XY/MS
   The recording of the soundscapes is done with the recorder Zoom H2N, which allows recording in 4 channels mode, generating two stereo sound files involving respectively the microphones MS and XY.



2. Decode MS



D'origine allemande, MS signifie Mitte Seite en Allemand et Middle-Side en Anglais. La radio stéréophonique transmet le signal sous la forme d'un signal monophonique, et d'un signal de différence entre canaux, qui s'ajoute à gauche et se retranche à droite. La prise de son MS utilise le même principe dès la captation. Une capsule cardioïde ou omnidirectionnelle est pointée vers le centre de la scène sonore, une deuxième capsule, à directivité en 8, est placé perpendiculairement aussi près de la première que possible, Le passage des canaux gauche et droite aux canaux M et S, et inversement, s'effectue par un procédé de somme et différences dit matriçage:

$$M = L + R$$
$$S = L - R$$
$$M + S = (L + R) + (L - R) = 2L$$
$$M - S = (L + R) - (L - R) = 2R$$

La profondeur de l'effet stéréo se règle facilement en ajustant l'intensité relative des deux composantes.[1]

3. Extract audio from MP4

```
$ ffmpeg -i in.mp4 out.wav
```

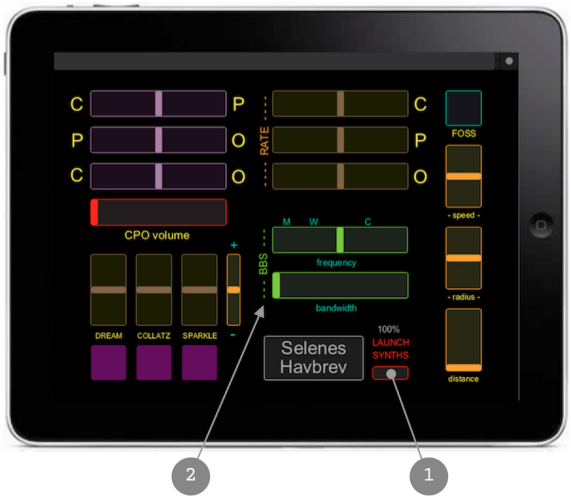
4. Trim audio files

```
$ sox initial.wav snippet.wav trim [SECOND TO START] [SECONDS DURATION]
```

5. Remove part in audio files

```
$ sox in.wav out.wav trim 0 =[SECOND TO START] =[SECOND TO STOP]
```

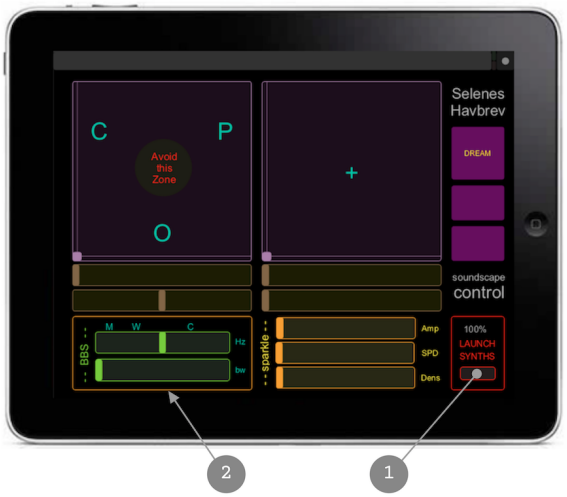
6. TouchOSC setup
   Set iPad TouchOSC Layout editor hosts and connections OSC with the IP of the remote laptop (see Network System Preferences).




```
CPO balance
CPO rate
CPO volume
DREAM push
DREAM amp
COLLATZ push
COLLATZ amp
SPARKLE push
SPARKLE amp
SPARKLE density
FOSS on/off
FOSS speed
FOSS radius
FOSS distance
```

Layout used during the play on the 23rd of August 2020 at Tromsø Kunstforening.

[1] Captation stéréophonique. *Wikipédia, l'encyclopédie libre.* [Page consultée le 31/07/20] http://fr.wikipedia.org/w/index.php?title=Captation_st%C3%A9r%C3%A9ophonique



CPO balance 2D
CPO amp
CPO rate
DREAM push
SPARKLE amp
SPARKLE speed
SPARKLE density
FOSS position 2D
FOSS distance
FOSS collatz density

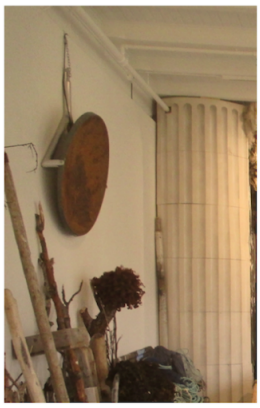
Layout used during the play on the 10th of October 2020 at Tromsø Kunstforening.

On ❶ stop and start the system – boot SC plus start synths –, and on ❷ a band reject filter applied to the soundscape according to the actor voices during the play.

Appendices

→ Study with vibrating speakers on two recycled corrugated sheets as installation. This installation performed the score of the second guitar of the composition *Selenes Havbrev* as a MDS – *Music Data Score* – with the digital wave guide physical model of a bowed instrument (SuperCollider *Ugen* `DWGBowedSimple` part of the SC3plugins) and `RM236`.

→ Surrealistic sound installation experimenting the SuperCollider *Ugens* `DWGBowedSimple` – playing with the bow parameters such as velocity, force, and position – and `NTube` (see Acoustics of Tube Models on page 175) – playing alternatively with the input as a pink noise on two tubes and with the impulse oscillator on ten tubes – (both are part of the SC3plugins) using respectively vibrating speakers on cello and on a parabolic antenna, as part of a 'post-apocalyptic' ambient quadraphonic soundscape.



SuperCollider | Browse | Search | Indexes ▼ Table Of Contents ▼

Guides (extension) | Open Composition

Solutions

Extension

Collaborative work

This is one proposition from my own work with the Themis' recording using SuperCollider.

Initially, we composed without consulting each other with the idea to use our respective composition as materials to re-compose.

This work requires the `GSA` and `cycle` extensions on https://github.com/yannics/

```
Document.open(Platform.gsa[0] ++ "oc" +/+ "xy.scd");
```

Initial works

Themis

https://thetasounds.org/

```
Platform.gsa[0] ++ "audio" +/+ "themis.wav";
```

Yann Ics

https://yannics.github.io/

Set some initial data :

```
(
/*
~frequencies = [ ... ]; // timbre profile
~melody = [ ... ];
~rhythm = [ ... ];
For a 'live' performance, this can be controlled via midi keyboard
*/
~rtmfactor = 8;
~delIn = 1;
~sustainRatio = 1.618;
~detune = 5;
~rndAmp = 0.1;
~ampRatio = 0.1;
// set the generator type, 0 -> reson, 1 -> sin
~sw = 0;
// adjust volume with relative distance
~ddist = 0;
// output buses
~xyBus1 = Bus.audio(s, 1);
~xyBus2 = Bus.audio(s, 1);
)
```

Prepare a routine on two channels with a distance parameter according to the pitch :

```
(
~xy = Routine {
    ~delIn.wait;
    ~melody.do({ arg midinote, i;
        var ser = ((~frequencies/~frequencies.minItem)*(midinote.midicps)).asInteger;
        var dist = midinote.linlin(~melody.minItem, ~melody.maxItem, 0, 0.5) + ~ddist;
        {
            [
                Out.ar(~xyBus1, Distance.ar(Ulam.ar(ser, ~sw, ~rtmfactor*~sustainRatio*~rhythm[i], detune:~detune,
rndAmp:~rndAmp), dist) * ~ampRatio),
                Out.ar(~xyBus2, Distance.ar(Ulam.ar(ser, ~sw, ~rtmfactor*~sustainRatio*~rhythm[i], detune:~detune,
rndAmp:~rndAmp), dist) * ~ampRatio)
            ]
        }.play;
        (~rtmfactor*~rhythm[i]).wait;
    });
    "Routine completed.".postln;
};
)
```

RESON :

```
// instance
Platform.gsa[0] ++ "audio" +/+ "reson.wav";
```

SIN :

```
~sw = 1;
~ddist = 0.3;

// instance
Platform.gsa[0] ++ "audio" +/+ "sin.wav";
```

Add convolution reverb by booting SC as done on section **Examples** in PartConv with the Dan Stowell impulse response.

Set some initial data :

```
// output buses
~busXY1 = Bus.audio(s, 1);
~busXY2 = Bus.audio(s, 1);
// Themis' work busses, path and buffers
~themisBus1 = Bus.audio(s, 1);
~themisBus2 = Bus.audio(s, 1);
~themis = "/full/path/of/themis.aiff";
~bufT1 = Buffer.readChannel(s, PathName(~themis), channels: [0]);
~bufT2 = Buffer.readChannel(s, PathName(~themis), channels: [1]);
```

Re-evaluate the rhythm factor to fit a given duration including some delay at the beginning and at the end of the routine :

```
//  according to the Themis' work duration:
// |---------- ~bufT1.duration ----------|
// |-------|-------...-------|-----|-------|
//  ~delIn     ~rhythm.sum    ~rel  ~delOut
~delIn = 10;
~delOut = 30;
~rel = ~rhythm.last * ~sustainRatio - ~rhythm.last;
~rtmfactor = (~bufT1.duration - ~delIn - ~delOut) / (~rhythm.sum + ~rel.amclip(~rel.clip2))
```

Load the phase vocoder :

```
SynthDef(\voc, {
    arg busCar, busMod;
    var car, mod, chainCar, chainMod, size=1024;
    car = InFeedback.ar(busCar);
    mod = InFeedback.ar(busMod);
    chainCar = FFT(LocalBuf(size), car);
    chainMod = FFT(LocalBuf(size), mod);
    chainCar = chainCar.pvcalc2(chainMod, size, {
        arg mag1, phs1, mag2, phs2;
        [mag1, phs2];
    });
    mod = IFFT(chainCar) * \amp.kr(0.2);
    Out.ar(\out.kr(0), mod);
}).add;
```

Play it as VOC 1 or VOC 2 :

```
(
{ PlayBuf.ar(1, ~bufT1)}.play(outbus: ~themisBus1);
{ PlayBuf.ar(1, ~bufT2)}.play(outbus: ~themisBus2);
~xy.play;
{
    // convolution on ~xy
    var input1 = In.ar(~xyBus1);
    var input2 = In.ar(~xyBus2);
    Out.ar(~busXY1, PartConv.ar(input1, ~fftsize, ~irspectrum.bufnum));
    Out.ar(~busXY2, PartConv.ar(input1, ~fftsize, ~irspectrum.bufnum));
}.play;

// VOC 1
a = Synth(\voc, [\out, 0, \amp, 1, \busCar, ~themisBus1, \busMod, ~busXY1]);
b = Synth(\voc, [\out, 1, \amp, 1, \busCar, ~themisBus2, \busMod, ~busXY2]);

// VOC 2
//a = Synth(\voc, [\out, 0, \amp, 1, \busCar, ~busXY1, \busMod, ~themisBus1]);
//b = Synth(\voc, [\out, 1, \amp, 1, \busCar, ~busXY2, \busMod, ~themisBus2]);
)
```

VOC 1 :

```
// instance
Platform.gsa[0] ++ "audio" +/+ "voc1.wav";
```

VOC 2 :

```
// instance
Platform.gsa[0] ++ "audio" +/+ "voc2.wav";
```

Discussion

Obviously, the Ics' recordings remain some propositions which can be modulated, adapted or developed according to the context of diffusion.

This basic material is destinated to be expended into a quadraphonic space. In that case, Themis' work would start solo for a deliberate duration, and 'adapted' to fit the quadraphonic space, according to a given directivity, either algorithmically or through sensor(s). The real time 'melody' wil be played on midi keyborad ...

WORK IN PROGRESS...

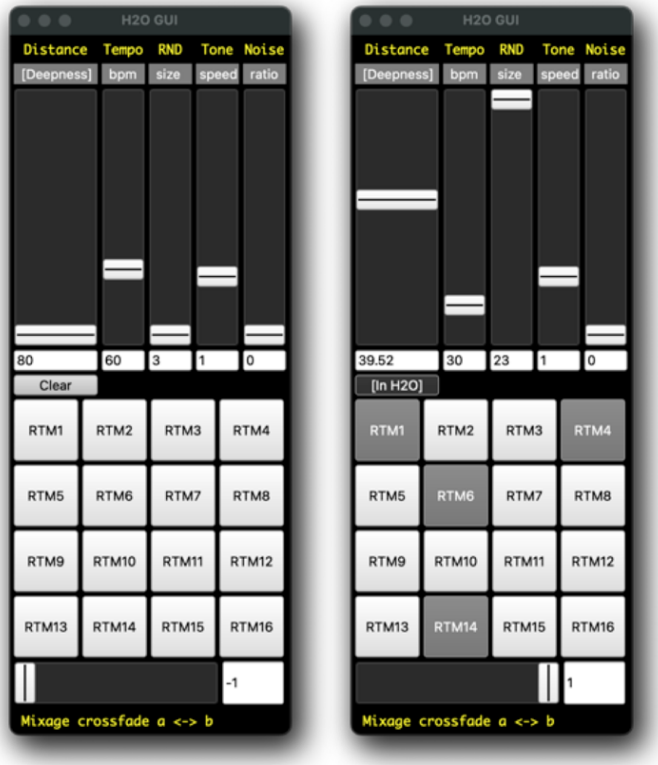
helpfile source: /Users/yannics/Documents/GitHub/GSA/HelpSource/Guides/Solutions.schelp
link::Guides/Solutions::

→ https://github.com/yannics/GSA/tree/master/SC/Solutions

H2O

Concept

The main idea is according to the topic of the ludes, to improvise through the SuperCollider GUI and some kind of live coding through global variables and routines.



– **Prelude** : DROP
Water drops ambiance playing within the quadraphonic space in terms of density and rhythm with the parameters `Distance`, `Tempo`, `RTM` matrix, and processes of synchronization and of desynchronisation.

– **Interlude** : STREAM
Water noise from different sources as a background ambiance – using the *Pseudo-UGen* `InH2O` (see Section 7.5). Some occurrences like the Doppler effect on a random part of the sample's bank, plus some traces of the Prelude in terms of variation playing with `Tone` and `Noise`.

Context

Sonic illustration during the storytelling/discussion evening about water, proposed and organized by the *café associatif de Lanester*. This composition for quadraphonic installation was interpreted on the 23rd of June, 2023 at the *Tiers-Lieu du Centre Social Albert Jacquard*, Lanester – *Kreisteiz Breizh*.

Required

`FluCoMa`
→ `https://www.flucoma.org`

Source

→ `https://github.com/yannics/GSA/tree/master/SC/H2O`

Notes

a) **Isolate and analyse water drops**
Within a bank of samples as the recordings of water drops from home use done by the persons involved in this project, the discrimination is done thanks to the tools of FluCoMa as a SuperCollider extension such as `FluidBufOnsetSlice`, and the analyze with the combination of `FluidBufMFCC` and `FluidBufStats`. Then the `FluidKDTree` is built from these data in the perspective to apply the method `kNearest` as described on the next point about the process of the Prelude.
The results of these analyses are stored in JSON files respectively as datasets named ds_analysis, ds_indices, and kdtree.

b) **Processing `kNearest`**
The function ~`processNearest` takes three arguments: the index of the rhythm involves (`RTM1, RTM2, ..., RTM15`), the nearest or the farthest point as boolean, and the length of the result which will be randomly picked (which will assure some kind of disparity and avoid monotony).
Then the process goes inside a SC routine triggered from the `RTM` matrix. Note that the rhythms of the matrix (~`rtmBase`) are completely arbitrary.
Each position of the drops (and this for each rhythm) – defined in a panoramic of four channels (see argument `xpos` and `ypos` of the *UGen* `Pan4`) – is previously randomly computed and stored in an array (~`xposAr`, ~`yposAr`) and can be re-evaluated on the fly, as well as the tempo ratio (~`ratAr`).

FluCoMa

The Fluid Corpus Manipulation project (FluCoMa) instigates new musical ways of exploiting ever-growing banks of sound and gestures within the digital composition process, by bringing breakthroughs of signal decomposition DSP and machine learning to the toolset of techno-fluent computer composers, creative coders and digital artists.

FluidBufOnsetSlice

a spectrum-based onset slicers according to a deliberate metric.
https://learn.flucoma.org/reference/onsetslice

FluidBufMFCC

a classic timbral spectral descriptor, the Mel-Frequency Cepstral Coefficients (MFCCs).
https://learn.flucoma.org/reference/mfcc

FluidBufStats

a statistical analysis on buffer channels such as the mean, the standard deviation, the skewness, the kurtosis, and low, middle, and high as the percentile values of the dataset .
https://learn.flucoma.org/reference/bufstats

FluidKDTree

a k-dimensional tree for efficient neighbourhood searches of multi-dimensional data.
https://learn.flucoma.org/reference/kdtree

Open Composition

« *I musicisti futuristi deveno allargare ed arrichirre sempre più il campo dei suoni.* »[2]

Prolégomènes

« *Everything we do is music.* »[3]

Du concept ...

1. Historique
   - 1.1. 1913 « *L'Arte dei rumori* » [Luigi Russolo]
   - 1.2. 1916 « *Ready-made* » [Marcel Duchamp]
   - 1.3. 1939 Musiques « expérimentales » [John Cage]
   - 1.4. 1967 « Le temps de le prendre » [Jean-Yves Bosseur]
2. Formalisation d'idées directrices
   - 2.1. musicales
   - 2.2. formelles
   - 2.3. structurelles
   - 2.4. interrelationnelles
   - 2.5. interactionnelles
3. Intelligibilité du discours musicales en termes
   - 3.1. intellectuelles
   - 3.2. expérientielles

... à la *praxis*

a. *In vivo* et *in situ*

b. Accessibilité à tout un chacun

c. Partage des savoirs et des connaissances.

[2] Luigi Russolo, *L'Arte dei rumori*, 1913, Conclusioni 1.
[3] John Cage in: Richard Kostelanetz, *Conversing with Cage*, 2003, p. 74.

« Comment comprendre la musique conceptuelle? Le processus de sélection d'un *ready-made* nous donne des éléments de réponse. Dans L'Œuvre de l'art, Gérard Genette explique, à propos de cet objet industriel élevé au rang d'œuvre d'art par un artiste dès 1913[4], que la représentation intellectuelle véhiculée par le ready-made est considérable: ce qui importe n'est 'ni l'objet proposé en lui-même, ni l'acte de proposition en lui-même, mais l'idée de cet acte'. » [Stévance, 2009, §11]

From tonal music with its rigorous counterpoint to free jazz, the composition offers, if not an infinity, at least a plethora of modalities. We are talking about compositional modalities, which often are agreed upon stereotypes, identified as archetypes, or emerging as prototypal.

« Noter, ce n'est plus nécessairement indiquer une hauteur de son, un rythme ... noter, c'est aussi inventer une écriture. » [Bosseur, 2005, p. 133]

Diwar Les Coet

Concept

- Un espace à occuper
- Un ensemble de musiciens, en petits groupes, formations, ou bien solistes
- Des partitions reposant sur un choral, une progression harmonique, et un profil rythmique
- Un tempo partagé
- Une performance

Context

Initialement écrit pour le Refuge des loups de Coat Fur dans le cadre d'un 'mini festival' sur une ou deux journées, le projet restât en l'état.
→ http://refugedesloups.org

Notes

- L'*espace à occuper* doit être suffisamment grand pour discerner les différents intervenants comme autant de '*stages*', mais pas trop de façon à percevoir la plupart d'entre eux dans les espaces 'neutres'.

[4] *i.e.* Marcel Duchamp.

- Les *partitions* doivent se construire à partir ou s'inspirer des schémas harmoniques et rythmiques proposés en annexe page 189.
- Le *tempo commun* consiste à une synchronisation formelle pour l'ensemble des participants. Cela peut se faire avec l'application INScore par exemple, voir en annexe page 154.
- La *perfomance* requiert une concertation apriori entre les participants afin de réaliser un parcours formel (enchainement, superposition, tempo, ...).

1. Les possibles dissonances comme autant de couleurs harmoniques – notamment intergroupe mais pas que – ne doit pas rebuter les participants. Il convient de rester concentrer sur sa partition au tempo donné.

   Le concept lui-même repose sur les superpositions intergroupes.
2. Il est à préciser que ce type d'interprétation ne mobilise pas tous le monde tous le temps (sauf pour les *tutti*). Aussi, une 'mise en scène' ou une 'théâtralisation' de la performance – pouvant reposer sur un texte et par conséquent scénarisable – est à prendre en compte, afin que chacun se repère dans le temps sans être cloué à son instrument ou à sa place durant toute la performance.
3. Pour finir, la performance doit être interprétée sans aucune sonorisation – sauf effet acoustique irréalisable autrement ou pour les instruments définitivement électriques et ce, dans le respect de l'équilibre sonore de l'ensemble.
4. L'effectif minimal envisageable est de 9 personnes – 3 voix (basse, baryton et ténor), un djembé, un dumdum et 4 guitares.

Nord

Concept

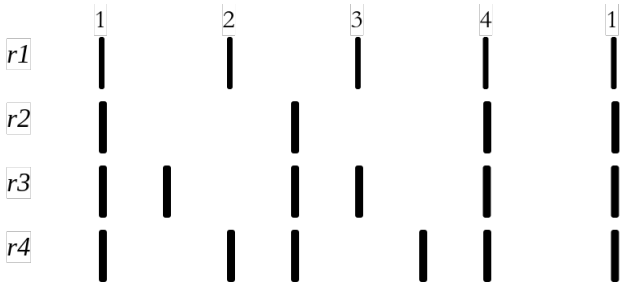
- One quest
- Two chords
- Three movements plus an interlude for at least deux guitars
- A workshop based on a rhythmic pattern
- The performance
- An *egregore*

Context

The quest for the pole.

Workshop

- One movement will be a collective composition with a graphical score involving the creativity of every participant, according to the constituent elements or the spirit of this Open Composition.
- Another movement, only based on rhythm, consists of following a diagram as the graphical conductor score.



This diagram is repeated as many times as needed in order to start very slow, then increase the pulsation gradually until very fast. If some participant drop out during the *accelerando*, the leader(s) will take over dynamically (*i.e. crescendo forte fortissimo* then *subito* silent or *decrescendo perdendosi*).

- Another one movement, will be based on the chord diagrams on annex page 195. Each musician or group leader is invited to create their own variation.
- Interludes for two guitars according to the scores in annex page 196, plus ideally one wind instrument.

Note

To introduce the musical event itself, there are some general recommendations to be followed.

1. The performance should be done as far as possible outside and without any amplification, except for some special effects. Also, the performers should be all around and even among the audience in order to occupy all space available.
2. There is no limit for the duration of each part of the performance, and no limitation for the number of performers involved.

3. Each soloist is free to act at any moment during the performance.
4. The order of the movements has to be defined by the participants, including the Interlude.

Kjølhiea

Concept

– One prayer
– Patterns as a guitar quartet
– Performances with electronic

Context

... And the wind blows at the mountain's summit.
At night, in gusts ...

Electronic versions in quadraphony:

- `K540` version 1 (see description page 95) – interpreted the 29th of January, 2018 – *Kulturhuset Hausmania* – Oslo.
- `K540` version 2 (see description page 99) – interpreted the 7th and the 8th of June, 2019 – *UddeboFestidalen* – Sweden.
  → `http://uddebo.uddetopia.com/uppleva-gora/uddebofestidalen`
- `K540` version 3 (see description page 100) – interpreted the 27th and the 28th of July, 2019 – *Trans' Festival* – Norway.

Source

Inspiré par un chant dont l'origine et la source se perd dans ma mémoire, dont les mots eux-même m'échappent, comme une prière hermétique, l'harmonie reste.

Note

The initial idea of *Kjølhiea* was to confront a guitar quartet (see appendix page 205, section *for 4 guitars*) – while allowing them a space of freedom, as much in terms of improvisation as in terms of a free interpretation, which could be based on chord diagrams for instance (see appendix page 206, section *chord diagrams*) – with an algorithmic composition for quadraphonic installation. The latter has been performed on several occasions as a solo piece under the title of K540 (see *context* point above).

Studies

This chapter describes some side works, most of them incomplete, as a proposition or outline, but still in relation to or in the continuity of this writing.

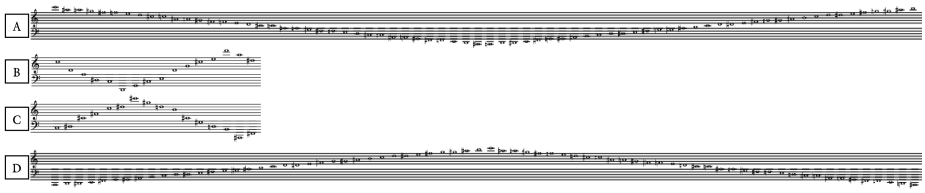
Study 1 – Entrelacs

This study consists of interlacing motivic patterns[5] within a given *ambitus*, according to other parameters such as the scale, the starting note, and the starting interval (toward upward scale if positive, or downward scale if negative), either as a musical score or as a process into the SuperCollider environment.

Generate Lilypond partition with Common Lisp

Initially, this work was made in the PWGL[6] context as a package named *Clavicorde*, just because the initial idea was to generate a score for clavicorde.

The idea then is to interlace two scales, one defined as the Messiaen mode `3 (2 1 1 2 1 1 2 1 1)` and an original one `(5 6 3 10)`. Then, I choose to start each scale on the C note by octava (i.e. 4 octava as 4 voices recovered the clavicorde *ambitus*), the two first ascending and the two last descending (by inverse intervals with `g-opposite`).



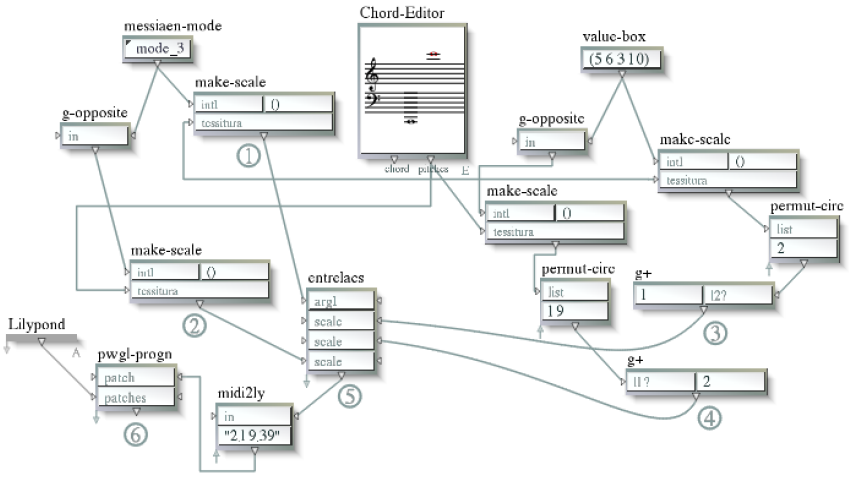
[5] See `interlace-cycle` in the Common Lisp library *cl-cycle* and the method `interlace` in the SuperCollider extension **cycle**.

[6] PWGL is a programming environment for an algorithmic composition written in the LISP programming language. It belongs to the Patch Work family initiated by Michael Laurson in 1985 including PatchWork and OpenMusic from the IRCAM and PWGL from the Sibelius Academy in Finland. Unfortunately, this software is not maintained anymore (the official website disappeared in 2020), but it is still possible to download it and to use it via emulation.

The Messiaen mode is wrapping the second scale. The latest need to be adjusted by a circular permutation (boxes `permut-circ`) to fit the cycle with the starting note (at least the closed one), then there are transposed (boxes `g+`) in order to start on the note C at the right octava.

Note that the *ambitus* (output of the `Chord-Editor`) is firstly set between C below F key (midi note 36) and C above G key (midi note 84). The real ambitus goes one tone beyond high C, that is to say, D (midi note 86), which is reached with the transposition of the two median scales.

PWGL Clavicorde Patch



The previous scales are defined as [A], [B], [C], and [D], and the outputs of the boxes of the following patch, matches as follows:

① → [D]

② → [A]

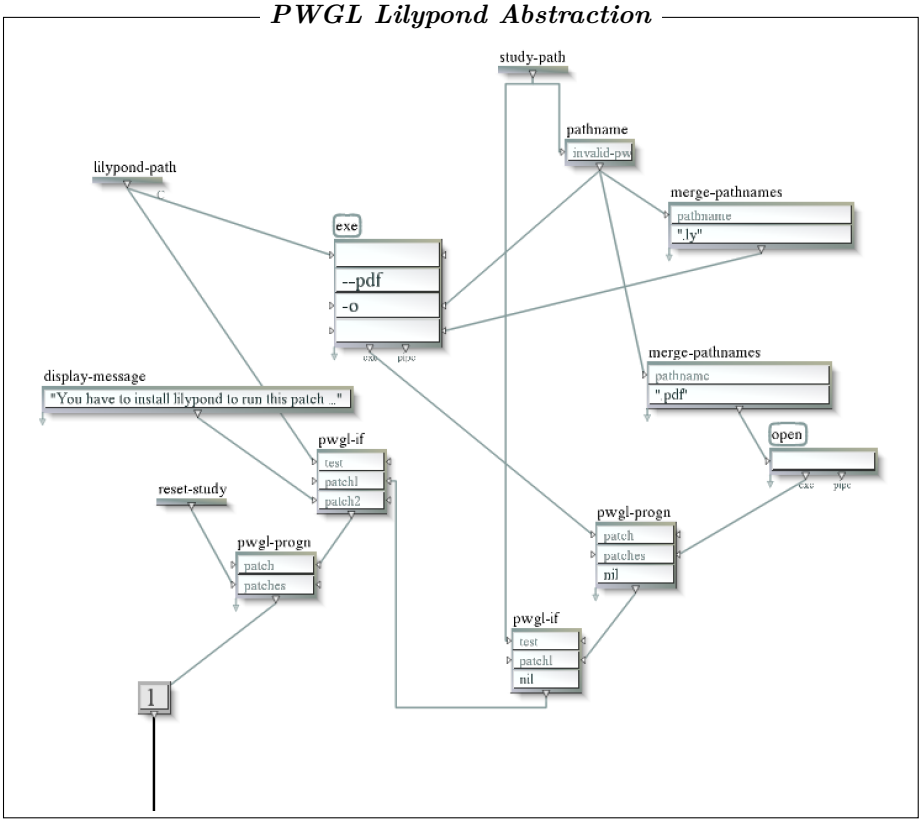
③ → [B]

④ → [C]

The output ⑤ applies the interlacing scales in such a way as to form a complete cycle by repeating 2 times [A] and [D], and 9 times [B] and [C].

Then, ⑥ write the Lilypond file (generated with the function/box `midi2ly`) and export it as PDF (see PWGL Lilypond Abstraction box next page and the score in appendix page 207).

PWGL Lilypond Abstraction



This was rather basic code and needed some *ad hoc* adjustments. PWGL code has been therefore rewritten and improved in 'pure Common Lisp', completing some gap previously outlined, into the package *cl-gsa* as the lisp file studies.lisp. The package *cl-cycle*, as well as the generic package Lilypond, are required to generate the score for Clavicorde previously done in the PWGL context.

The function `make-scale` has been improved by adding two optional key parameters the midi root note (the minimal value of tessitura by default), and the beginning degree of the scale (0 by default).

As an example, and in order to explore the possibilities of this work, we reproduce the scales used in PWGL tutorial such as:

```
(defvar *interlace*
  (cl-cycle:INTERLACE-CYCLE
    (make-scale (messiaen-mode 3) :opp '(36 84))
    (make-scale '(5 6 3 10) :opp '(36 86) :root 37 :deg 2)
    (make-scale (opp '(5 6 3 10)) :opp '(36 86) :deg 3)
    (make-scale (opp (messiaen-mode 3)) :opp '(36 84))))
```

```
(entrelacs *interlace*
  :path "~/Desktop/entrelacs.ly"
  :title "Études"
  :subtitle "N°1 Entrelacs"
  :instrument "clavicorde"
  :composer "Yann Ics"
  :character "Vivacetto, leggiero e comodo"
  :endnote
  "Cette étude représente un cycle formel. Les éventuelles
   reprises sont laissé à la discrétion de l'interprète et
   se font à partir de la barre de mesure en pointillé.
   L'absence de la double barre de fin indique simplement
   le caractère cyclique de cette étude et suppose une
   continuité hors de l'espace audible.")
```

The generated score is in the appendix page 207.

Obviously, this remains a work in progress, and some more keys need to be added to the function `entrelacs` to improve the score, like the key type according to the instrument, the open or the closed end bar (with or without coda), to name a few.

Generate SuperCollider process

```
~scale1 = [ 2, 1, 1, 2, 1, 1, 2, 1, 1 ];
~scale2 = [ 5, 6, 3, 10 ];

~entrelacs =
[
 [ 36, 84 ].cycleMidiScale(~scale1, ~scale1.reverse, 36),
 [ 36, 86 ].cycleMidiScale(~scale2, ~scale2.reverse, 37, 2),
 [ 86, 36 ].cycleMidiScale(~scale2.reverse, ~scale2, 38, 3),
 [ 84, 36 ].cycleMidiScale(~scale1.reverse, ~scale1, 36)
].interlace;

Pbind(\freq, Pseq(~entrelacs.flatten.midicps, inf), \dur,
    0.2).play;
```

Study 2 – Experimental Score Graph (ESG)

Convert an audio file to a graphical score with the analysis of *enkode*. The score reveals as such one aspect of the audio in terms of structure and/or form and can offer some interesting perspective as a concept for a musical interpretation.

```
$ enkode -I '(5 3.5 3.5 4 4)' --loudness-diff-thres=1
  --loudness-min-thres=2.5 --loudness-max-thres=15
  --min-duration=0.1 --max-duration=5 --cutoff-frequency=100
  --smooth-frequency=500 --time-step=0.01
  .../audio.wav > data.enk
```

The arguments of `-I` refer respectively to a discrimination for:

- **duration** of 31 $(2^5 - 1)$ classes (because of the limited note values) as proportional notation,
- **f0** of 8 $(2^{\lfloor 3.5 \rfloor})$ classes (in order to fit inside the stave) as the number of different notes on the top stave,
- **centroid** of 8 $(2^{\lfloor 3.5 \rfloor})$ classes (in order to fit inside the stave) as the number of different notes on the bottom stave,
- **loudness** of 15 $(2^4 - 1)$ classes as the size of the note of the bottom stave,
- **bass loudness** of 15 $(2^4 - 1)$ classes as the gray shade of the note of the bottom stave.

So, the values of `-I` can't be modulated in order to fit the graphical constraints.

```
;; generate lilypond file ...
CL-GSA> (score-graph ".../data.enk")
;; for details see comments in studies.lisp file
```

The generated score is in the appendix page 211.

Appendices

Codes

1 Figures

1.1 1.2

```
# GNUPLOT
set terminal png size 800,300
set output 'profile.png'
set title ''
unset border
unset xtics
unset ytics
plot  "profile" every ::1::500 using 1 title '' with lines
# 500 as 5 seconds according to the time step of 0.01 second
```

1.2 2.1

```
# GNUPLOT
set terminal png size 400,300
set output 'sample.png'
set title ''
set logscale y 10
set xtics 1
set tic scale 0
unset key; unset ytics; unset border
plot  "sample.txt" using ($0+1):1 title '' with impulse
# sample.txt is the harmonic profile of the sample
```

1.3 4.2

```
# Shell script
# require SOX + GNUPLOT
#!/bin/bash
# $1 = soundfile
# $2 = textgrid
# $3 = duration

name=`basename "$1" | cut -d. -f1`
dur=$3
sox $1 1.wav trim 0 $dur
dsr=`soxi -s 1.wav`

# convert sound file to data text
nc=`soxi -c 1.wav`
if [ "$nc" -eq 1 ]
then sf=`sox 1.wav 1.dat`
elif [ "$nc" -eq 2 ]
```

```
then
sox 1.wav 2.wav remix 1,2
sf=`sox 2.wav 1.dat`
else
echo "Accept only mono (1 channel) or stereo (2 channels)
    sound file."
fi

> 2.dat
# the number of bin sample divided by n allows to reduce
# the number of data in order to reduce the computation
# time according to the image resolution required
n=10
tail -n +3 1.dat > 3.dat
value=0
while read line
do
    if [ $(( $value % $n )) -eq 0 ] ; then
        echo -e "$line" | xargs >> 2.dat
    fi
        let value=value+1
done < 3.dat

# write timing segmentation
l=`cat $2 |wc -l`
ll=`expr $l - 11`
tail -n $ll $2 > 4.dat
awk 'NR == 1 || NR % 3 == 0' 4.dat > 5.dat
rm 4.dat
while read p; do
if [ 1 -eq "$(echo "${p} < ${dur}" | bc)" ]
then
echo  `awk "BEGIN {printf \"%.3f\n\", $p}"` >> 4.dat
fi
done < 5.dat

# write gnuplot file
echo "set terminal png size 1200,300" > 1.pl
echo "set output '$name.png'" >> 1.pl
echo "unset border;unset xtics;unset ytics" >> 1.pl
echo "plot \"4.dat\" every ::0::$dsr using 1:(\$1 <=$dur ? 2
     : 0) title '' with impulses lc rgb '#A9A9A9', \"2.dat\"
    every ::0::$dsr using 1:(\$2+1) with lines lc rgb
    '#696969' title \"\"" >> 1.pl
gnuplot 1.pl
rm 1.pl 1.dat 2.dat 3.dat 4.dat 5.dat 1.wav 2.wav
```

2 Transpose

```
# source: https://stackoverflow.com/a/1729980
transpose(){
awk '
{
    for (i = 1; i <= NF; i++) {
        if(NR == 1) {
            s[i] = $i;
        } else {
            s[i] = s[i] " " $i;
        }
    }
}
END {
    for (i = 1; s[i] != ""; i++) {
        print s[i];
    }
}' $1
}
```

3 formantAnalysis.sh

```
#!/bin/bash

dirfile=$1
dirpraat='/Applications/Praat.app/Contents/MacOS/Praat'
if [ ! -e $dirpraat ]
then dirpraat=`readlink -f $(which praat)`;
fi

#-----------------------------------------------------
# convert praat.formant in coll file for SuperCollider
# coded by Fred Voisin - January 2012
# modified 21022012/18102015
convert(){
praatfile=$1
l=`cat $praatfile |wc -l`
ll=`expr $l - 9`
tail -n $ll $praatfile > "$praatfile.tmp"
coll="$praatfile.coll"

i=1
index=1
while read LINE
do
        ii=$((i + 1))
        amp=$LINE
        read n
        line="$amp "
        k=1
```

```
        n=`expr $n \* 2`
        while [ $k -le $n ]
                do
                read p
                line="$line $p"
                k=$((k + 1))
                done
        echo "$line "|cut -d\  -f 1,2,3,4,5,6,7,8  >> $coll
        i=$((ii + n + 1))
        index=`expr $index + 1`
done < $praatfile.tmp

rm $praatfile.tmp
}
#--------------------------------------------------
echo "form Formant analysis
        sentence File ...
endform
        Read from file... 'file$'
        To Formant (burg)... 0.005 5 5000 0.025 50
        Write to short text file... .tmp
select all
Remove" > formant.praat
#--------------------------------------------------
> formant.coll

for file in `find $dirfile/* | grep '/*.mp3\|/*.MP3\|/*.aiff\|/*.
    AIFF\|/*.WAV\|/*.wav'`
do
    $dirpraat formant.praat $file

    convert .tmp

    cat .tmp.coll >> formant.coll
    rm .tmp .tmp.coll
done
rm formant.praat
```

4 nw_display

```
# Source: newick-utils-1.5.0
# http://cegg.unige.ch/newick_utils
# ---> radial
nw_display -sr -S -w 1000 -W 20 -b 'opacity:0' in.nw
> out-r.svg
# ---> orthogonal
nw_display -s -w 1000 -u '' -t -W 20 -b 'opacity:0' in.nw
> out-o.svg
```

Distance

The perception of a sonic object with its distance implied a substantial amount of parameters more or less subtle such as the nature of the source itself (mainly power level, the directivity of the source, frequencies composing the sound, and the mode of propagation), the environment in terms of reverberation, and more specifically in terms of absorption, diffusion, reflection and diffraction, which are dependant of the environment as the density (e.g. air humidity), the temperature, the pressure and perhaps the flux of the milieu (e.g. the wind in the air or the current in water).

In this extremely simplified model, I consider a given relative distance from the source, the inaudible point as a relative distance from the source, and the absorption rate defined by its curvature.

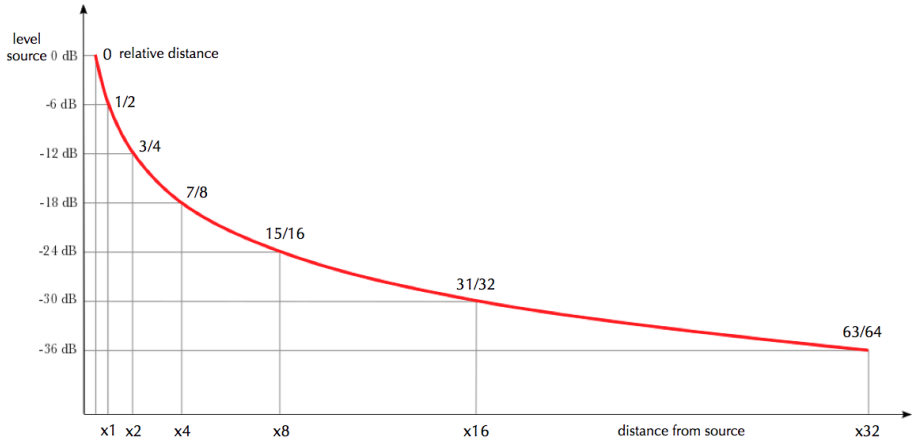


Figure 1: The sound level decreases of 6 dB for each doubling of the distance such as $-20\,log_{10}\,distance$.

We talk about relative distance because too many parameters interact with each other. Then, it is quasi impossible to estimate it with an usual metric system, especially if we compose with abstract object(s). So, the idea is to say the value 0 for the closest distance, and the value 1 for the farthest which has to be inaudible as a point somewhere on the curve of the figure 1.

```
// relative distance decibel conversion
// ---> .dist2db
(-20*(1-dist).reciprocal.log10)
// ---> .db2dist
1-(10**(db/20))
```

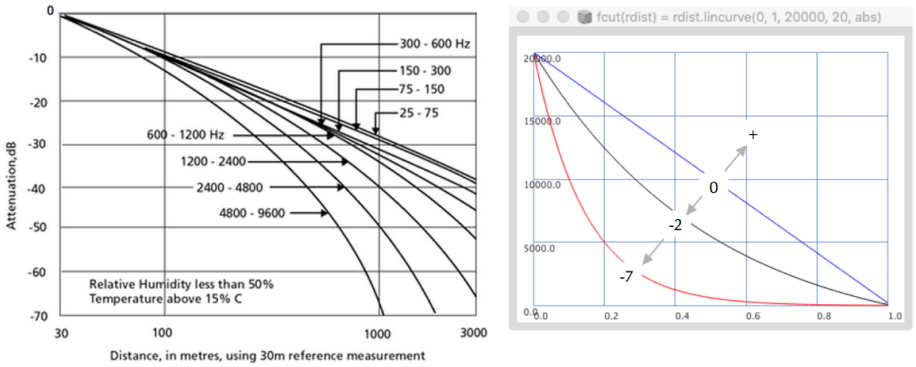
The distance involves a dynamic perspective inside the audible field in terms of frequency attenuation due to the absorption rate between the source and the listener (understood the attenuation increases with the frequency).

Basically, this is defined as follow:

$$A_x = A_0 \cdot e^{-x/\zeta}$$

A_0 is the amplitude of the plane source. A_x is the amplitude after it has propagated a distance x through the medium. ζ is the amplitude decay constant in meter unity. It depends on frequency – proportionally to its square –, temperature, humidity and pressure.

The further is the source, less we perceive the high frequencies which are absorbed by the air – see figure 2 (a). This process is resumed as a low pass filtering scaled according to the human audible field, i.e. between 20 Hz and 20000 Hz with an exponential progression – see figure 2 (b).



(a) (b)

Figure 2: (a) [Brüel & Kjær, 2021] Approximate correction for air attenuation including the inversesquare law. (b) The SuperCollider method `lincurve` determines the cutoff frequency of the low pass filter such as for a given relative distance `rdist`, the cutoff frequency is equal to `rdist.lincurve(0, 1, 20000, 20, abs)`. Thus, the parameter `abs` setting the curvature of the curve allows adjusting the rate of the absorption – -7 by default.

Speculative Fractal Composition

Algorithm versus concept

Article by Yann Ics

It is worth recalling the relevance of fractals in music, which remains a relatively frequent compositional modality, in an intuitive manner more or less consciously, or deliberately. Because as we know, repetition and variation are the essence of music, algorithms such as fractal allow us to systematize such a multidimensional approach with any compositional parameter. By coding, thanks to the SuperCollider concept, any pattern and recursivity can be managed independently to offer flexibility in terms of variation, opening a wide field of musical exploration, experimental, research-based, including sound synthesis.

tags: concept, fractal, composition, algorithm

Concept

Being speculative in art relates to a hypothetical, futuristic, or imaginary outcome through (a) specific medium(s) – involving most of the time interdisciplinarity such as science, sociology, cognition, and philosophy, to name the most common. This is expressed or covered in terms of concept. The concept is a mind construction that allows one to name a state of knowledge that results in a set of epistemological relationships in a chain of signifieds in a given context conditioned by our culture and the acquired knowledge. Far from being frozen in time, the concept evolves and changes by using it, serving more to name an idea in its diversity than to describe an object in its uniqueness.

Music is by definition a concept. Music is also a speculative object because it does not exist as such in all cultures[1], and each culture, even each person has its own view of what is or what should be music (inescapable tautology). The speculative perspective I propose, besides describing or defining the object itself, aims to challenge conventional boundaries and more widely allows the exploration of possible futures through ineluctable or expected societal changes, with this intent in music to go further the experience itself, through or by composition, invention, and transmission of musical ideas.

Thus, the concept(s) becoming music can be depicted as the frame in which the musical phenomenon is possible, often expressed in terms of structure or form. In this sense, the concept aims a result in or for a given situation. Indeed,

a piece of music can be the expression of several concepts as compositional or interpretive modalities, before becoming a concept itself in certain cases. These cases concern prototypes as original compositions towards archetypes and identified as compositional modalities.

Like music, aesthetics is an elusive concept and by definition speculative. Basically, we can say that aesthetics tends towards a transcendent state or at least towards a certain satisfaction. But despite the strong subjectivity that aesthetics implies, both individually and collectively, and beyond cultural legacy, even a certain atavism, aesthetics in music is highly related to the 'zeitgeist'. Concerning composition and performance, it is a trend, shared more or less with our contemporaries, intimately related to the level of our own understanding of music in terms of cognition[2], algorithmic, and synthesis.

In this context, the aesthetic is not one of my purposes as such, but rather a way for questioning the listener about music as media understood as an extension of 'Man'[3], through my quest of thinking and composing music otherwise.

Fractal

Music, like all things, is a process involving repetitions and variations. Of a usage without place or date, repetitions and variations are combined in various modalities, such as transpositions, symmetric transformations, and occasional or punctual rhythmic or pitch alteration, among others, mainly conditioned by the modality of conception (improvisation, partition, or process). Because the perception itself changes in time according to the context and the 'focus window' of the listener, strict repetition in the time domain is only a view of the mind. Therefore, variations can take as many directions as contrasts or differences perceived as such. Then, from micro variations to extreme contrasts, the field of possible paths is as large as the understanding and the possible discretizations of the sonic phenomenon.

One of these approaches is to condition the entire structure by self-similarity by repeating a pattern or profile, at various time lags, scales, dimensions, or levels. It is from the sixteenth century we name the imitations of melody as a contrapuntal compositional technique the canon. In this paper, I focus on a particular case, known as the prolation canon, where imitations are carried out at different time scales[4]. Although fractal is a relatively recent term defined by Benoit Mandelbrot in 1975[5] for a geometrical object that fulfills these criteria of self-similarity (strict replication at every scale), the term fractal as a system in music and in the time domain is in its sonification. Nevertheless, in nature, although fractals are omnipresent, the self-similarity remains approximate because of the environmental constraints and, can be formalized otherwise like in biology such as a formal system called the Lindenmayer system based on grammar rules.

Anyway, this work focuses on fractals as a self-similar pattern only, and it is the interpretation of each repeated pattern according to its position in time and dimension that determines its degree of variation if any.

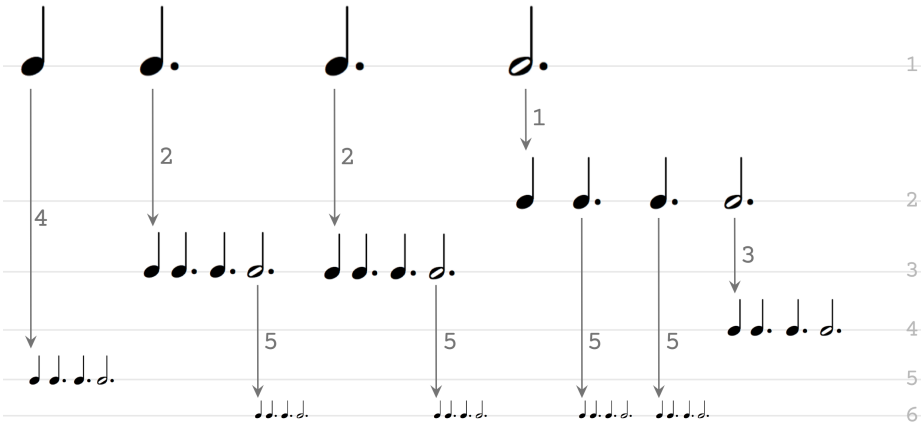


Figure 1: `a = Fractal.newFrom([ 2, 3, 3, 6 ], rec:5)`

SuperCollider, as a programming language for real-time audio synthesis and algorithmic composition, is the perfect tool to manage fractals by coding the algorithm itself [6, Section 6.3] as a class well-named `Fractal` [6, `gsa.quark`] and managing its interpretation thanks to the flexibility of the coding.

```
procedure FRACTAL(rtm, dur, rec, min, al | R, int)
    if R is nil then R ← rtm.normalizeSum*dur
    if int is nil then int ← ASSOC(rtm, al)
    tmp ← [ ]                          ▷ for one dimension as sub-array of R

    for i in R[0] do
        if i = max(R[0]) then
            tmp ← rtm.normalizeSum*i
        else  tmp ← i

    if rec = 0 or min ⩾ min(R[0]) then
        return [R, int]
    else
        if rec ∈ ℕ+ then rec = rec−1
        FRACTAL(rtm, dur, rec, min, al,
                    tmp.flat → R,
                    ASSOC(tmp, al) → int)
```

▷ rtm *is a rhythm defined by a numerical array as a list of event's durations* ◁
▷ dur *is the total duration* ◁
▷ rec [optional] *is the number of dimensions or of recursions* ◁
▷ min *is the minimal duration accepted which is required as* min ∈ $\mathbb{R}^{*+}$ ◁
▷ al [optional] *is an associative list respecting the order of* rtm ◁

The algorithm itself is trivial, but this is the purpose of fractals. The complexity is closely related to the initial rhythm and the level of recursion, which (that level) is correlated with the duration of the piece and the value of the minimal duration. Complexity also depends on the interpretation in terms of variations and synthesis parameters.

Note the function ASSOC which assigns for each duration in time their effectiveness as an array of 1 (or an event as an array) if so and as 0 (or a set of 0 according to the length of the event) if not. The events are grouped according to the initial dataset.

The figure 1 illustrates the algorithm process using the code implemented in SuperCollider. In this instance, the number of dimensions defined by `a.depth` returns six as five recursions plus the 'seed' (line 1).

SuperCollider code :
https://github.com/yannics/GSA/blob/master/mp/fractal/code/fractal2ch.scd

The speculative fractal composition is a technical tool to structure music, a way to explore complex structures from simple ideas in order to manipulate sound at different levels related to its initial 'axiom'. This can be seen precisely as a compromise between simplicity and complexity[7], an edge over which the funambulist composer evolves. With this approach, different aspects of the fractal can be considered to structure the work as the level of recursivity, the value of discretization, with the related object of each event when it is provided, to name those I have implemented. Some controlled or random micro-variations are also sought within an algorithmic music context to create more 'musical' results, which can be related to the expressivity or something more 'organic' in general terms. For written music, this can be equally a tool, a guideline, which must be adapted, interpreted to fit the composer's musical intention.

References

[1] *Musiques. Une Encyclopédie pour le XXIe siècle. 2007. « 5. L'Unité de la musique »*, sous la direction de Jean-Jacques Nattiez. Arles-Paris : Actes Sud/Cité de la musique

[2] Bob Snyder, Robert Snyder, *Music and Memory: An Introduction*, A Bradford Book Mit Press, 2001.

[3] Marshall McLuhan, *Understanding Media – The Extensions of Man*, First edition 1964, MIT Press edition, 1994.

[4] John McDonough, Andrzej Herczyński, *Fractal patterns in music*, Chaos, Solitons & Fractals, Volume 170, 2023.
Online https://doi.org/10.1016/j.chaos.2023.113315

[5] Benoit Mandelbrot, *Les objets fractals : forme, hasard et dimension*, (first edition 1975), Flammarion, 2010.

[6] Yann Ics. *Journal of Generative Sonic Art*. Articles/Reports, 2014–2024. Online https://github.com/yannics/GSA/

[7] Henri Pousseur, *Musique, Sémantique, Société*, Ed. Casterman, Paris, 1972.

*****

Furthermore

To illustrate some possible interpretations applied to the SuperCollider class `Fractal`, here is a list of my previous works using it in a quadraphonic installation context.

`HEXO` **mvt 1** [6, Section 8.2]

According to a set of musical phrases, the fractality is applied to the well-known Risset Bell controlling its frequency and sustain, and the spatialization in terms of distance correlated with the level of fractality. The durations depend on a given ratio.

`DATA-01` **part 3** [6, Section 8.4]

From the contrastive analysis [6, Section 4.4] applied to a given sound file, a deliberate substructure determines the fractal onsets, playing on the spatialization of streaming radio in terms of distance and panoramic, like the movie editing, from one shot to another according to a rhythm conducted by the said fractal onsets.

`105A1408` **layer 2** [6, Section 8.6]

The fractal is applied to the identified recurrent patterns inside a given soundscape (bird songs concerning this performance) – according to the analysis *enkode* [6, Chapter 1] involving only the duration and the centroid for each event – and chosen randomly for each initialization. A SuperCollider GUI [6, Figure 8.11] controls also the presence in terms of distance for each dimension. Each onset triggers a percussive sample among a significative set. The centroids as data condition the rate of the samples.

*****

At the marge of and related to this work, I explored some ideas between the prolation canon and fractal that I called proportional canon. The main idea is to scale a given object according to an 'attractor' (which can be seen as a climax) and according to a given duration and ratio(s). This object can be an array of durations or a sample as a sound file. Also coded in SuperCollider context, the first one called `Canon` [6, Section 6.2] is a class allowing two modalities, either by linear interpolation between the total duration and this duration multiplied by a given ratio or by the same ratio applied recursively on each voice. The position of the 'attractor' depends on this ratio which must be a number between 0 and 1 respectively excluded. The second one called `Sow` [6, Section 7.4] is a *pseudo-UGen* applied to a sound file and played at different ratios. In this case, the position of the 'attractor' is provided in second within the sample at ratio one.

Even though I experimented with the 'proportional canon' in some of my works, there is no need to go further in detail because it is still very experimental and under development. Be that as it may, this remains relevant to mention here.

*****

To complete my writing, I would like to mention an analytical tool I have elaborated on, which gives the possibility to retrieve the fractalities of a given pattern inside a sequence using an algorithm called `differential-vector` [6, Chapter 5]. This algorithm developed in Common Lisp compares two patterns defined by their respective durations and pitches (the latter can refer to any numerical coupled with the durations) and returns either a vector or the normalized norm of this vector. This estimates numerically the difference or the similarity between these two patterns according to the concordances on the onsets (time domain on the x-axis) and the profile (of the first derivative of the pitches or other profiles on the y-axis). Then, if we apply this method to a whole sequence, by windowing according to a given length, we can detect the number and the position of each pattern matching the referent one, giving us an idea of redundancy of occurrence at different levels and different positions, as deliberate intent of the composer or as emergent phenomenon or surface structure, regardless all musical 'decorations'. There exist biases, but this can be lessened with some settings of the algorithm, and according to the required relevance or accuracy of the analysis.

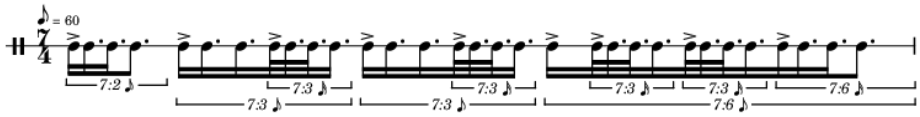


Figure 2: Transcription of the figure 1 considering only the onsets.

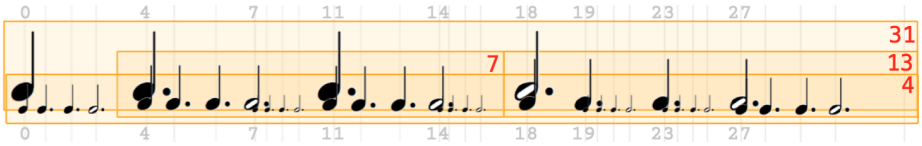


Figure 3: Analysis in terms of redundancy with the Common Lisp algorithm `differential-vector` of the fractal sequence generated on figure 1 in SuperCollider context.

Common Lisp code :

https://github.com/yannics/GSA/blob/master/mp/fractal/code/differential-vector.lisp

For instance, if we consider the fractal described in figure 1, based only on the differential durations, windowing from the length of the referent pattern to the size of the whole sequence, the algorithm returns six occurrences with a length of 4 at the indices 0, 7, 14, 19, 23, and 27, two occurrences with a length of 7 at the indices 4 and 11, one occurrence on 18 with 13, and one occurrence for the whole sequence; which match as expected the accents of the transcription on the figure 2. The settings of the algorithm `differential-vector` are the output result as the vector of difference, the concordance on the x-axis according to the length of the referent (which means the minimal cardinal understood we check from this value, that is to say 4 in this instance), and a threshold of 0.001 because of rounded float numbers involved. The figure 3 resumes the analysis.

*****

INScore ... in action

An environment for the design of interactive, augmented, dynamic musical scores.

https://inscore.grame.fr

October 21, 2024

https://scsynth.org/u/Yann_Ics

INScore version 1.31
SuperCollider version 3.13.0

This short article aims to illustrate one way to use the open-source software INScore, like a tutorial introduction within the SuperCollider context. Furthermore, this could be the first step toward developing an interface as an extension of SuperCollider.

* * *

Here is the context

I am studying on a musical project named *Untitled* `#2` developed in SuperCollider as an algorithmic and synthesis composition. In short, a third-party application generates a melody based on the dynamic Markov chain and an *ostinato* as an emergent phenomenon due to the structure of a resulting network. This said application is about an AI developed in Common Lisp called *Neuromuse3* based on a connectionism network and trained with the encoded voices of the first book of J. S. Bach's *Das Wohltemperirte Klavier* (BWV 846 – 869). All these data are sent via the OSC protocol. Then SuperCollider interprets them musically via different syntheses. A GUI mixer allows one to formalize the work in a real-time performance.

From this point, I imagined that one musician performer could interact by improvising according to some guidelines displayed through the INScore viewer.What should shown is the harmonic context (*in-time* with a before and an after to anticipate the oncoming) as note names plus optional transitional notes, and the rhythm related to this event. So, there is no direct interaction with INScore, only indications to serve the musician playing.

* * *

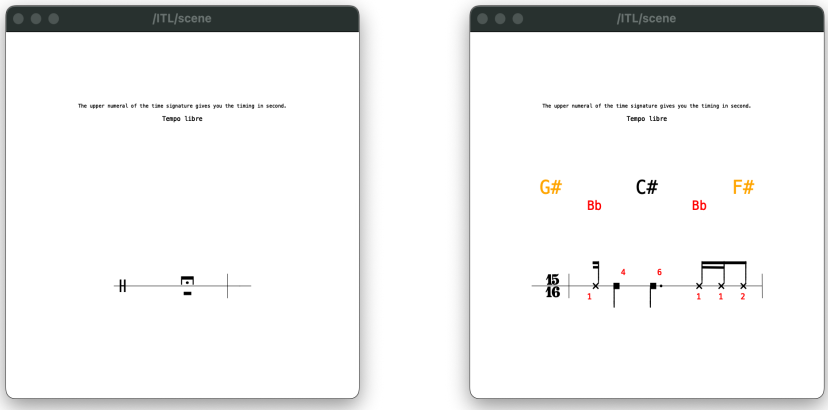
The coding part

Beforehand, to initialize the INScore display, some preliminary setup needs to be written on a file to be loaded remotely. These setups mainly consist of positioning each object related to each other. Although this can be done dynamically from a third-party application via OSC messages.

```
# clear the scene
/ITL/scene/* del;
/ITL/scene width 1;
# ---
/ITL/scene/timing set txt "The upper numeral of the time signature gives you
 the timing in second.";
/ITL/scene/timing y -0.6;
/ITL/scene/timing scale 0.8;
/ITL/scene/tempo set txt "Tempo libre";
/ITL/scene/tempo y -0.53;
# harmony
# --- previous
/ITL/scene/a1 set txt "";
/ITL/scene/a1 scale 3;
/ITL/scene/a1 color 'orange';
/ITL/scene/a1 y -0.15;
/ITL/scene/a1 x -0.55;
# --- transitional previous
/ITL/scene/a2 set txt "";
/ITL/scene/a2 scale 2;
/ITL/scene/a2 color 'red';
/ITL/scene/a2 y -0.05;
/ITL/scene/a2 x -0.3;
# --- current
/ITL/scene/a3 set txt "";
/ITL/scene/a3 scale 3;
/ITL/scene/a3 y -0.15;
/ITL/scene/a3 x 0;
# --- transitional next
/ITL/scene/a4 set txt "";
/ITL/scene/a4 scale 2;
/ITL/scene/a4 color 'red';
/ITL/scene/a4 y -0.05;
/ITL/scene/a4 x 0.3;
# --- next
/ITL/scene/a5 set txt "";
/ITL/scene/a5 scale 3;
/ITL/scene/a5 color 'orange';
/ITL/scene/a5 y -0.15;
/ITL/scene/a5 x 0.55;
# rhythm (pause)
/ITL/scene/rtm set gmn '[ \meter<""> \staffFormat<style="1-line">
 \clef<"perc"> \fermata<"long">(_/1) | empty*1/32 \staffOff ]';
/ITL/scene/rtm y 0.4;
/ITL/scene/rtm scale 0.8;
# set scene font
/ITL/scene/* fontFamily Courier;
```

Each line is defined by the discriminant or tag `/ITL/`, the name of the scene, i.e. the window viewer, plus an identifier, followed by a keyword and its parameter(s) if required, which are well documented with examples in the INScore application folder as PDF and INScore files. Here, only the lines concerning the text of harmony and the score `gmn` (acronym of guido musical notation) will be dynamically set according to the context. The code is saved (in a file named `ics.inscore` currently) and must be loaded in a remote server (from the INScore API) in order to be accessible from any device.

The score consists of displaying in the INScore viewer the previous harmony, the current harmony, and the next harmony plus the passing notes as a set of note names, and the rhythm partition.



(a) initialization and waiting state ... (b) ... in action ...

Figure 1: This is what should be displayed on the musician's device.

On the SuperCollider side, I receive data from a third-party application – namely *Neuromuse3*.

```
// opening incoming port
thisProcess.openUDPPort(7772); // ostinato
```

Note: In this case, the data are collected through OSC but can be obviously set or generated within SuperCollider. Also, I had to use SC as a router because as long as there is no standard in Common Lisp (it exists an interface within INScore but requires Lispworks which is not free and even less open source) I have taken side to code in SC. The Common Lisp interface remains hence to be done or at least extended to SBCL and CCL among others.

Then, SC needs to initiate the network address to communicate with INScore, and incidentally, the INScore file previously discussed.

```
// set sending address
~inscore = NetAddr.new("10.25.172.200", 7000);
// load init setup
~inscore.sendMsg("/ITL/scene", "load",
                 "http://remote.server/ics.inscore");
```

Note: The initialization file needs to be accessed from a remote server for convenient purposes, mainly because like so it is always possible to load the file directly from any device. Anyhow the IOS INScore version does not allow to loading of files in the API.

From this point, each specific data has to be sent in real-time with the component of the `/ITL/` concerned scene according to the following syntax:

```
<net-address>.sendMsg("/ITL/scene/identifier", <remaining-message>)
```

The remaining message is a list of parameters as strings or numbers separated by commas.

Note: I coded some methods to format notes, chords, and musical notations, according to the Guido syntax (see lines 286 to 391 of `gsa.sc`) intended to the 'remaining message':

- `midiguido` – Integer/Array *arg* [dur]
- `degguido` – Intger *args* [dur, range, to, asChord]
- `tag` – String *args* [tag-name, rest-arg(s)]

For obvious convenience, I made functions to well-format the messages to be sent according to raw data.

```
~harmonyINScore = {
    | prevOst, degOst, intOst, nextOst, netaddr |
    var nn=["C","C#","D","Eb","E","F","F#","G","G#","A","Bb","B"];
    var cn = degOst.midiguido(noteNames: nn);
    var res = Array.with(
        [if(prevOst.asArray[0].isNil)
            {""} {prevOst[0].midiguido(noteNames: nn)}],
        [if(prevOst.asArray[1].isNil)
            {""}
            {(prevOst[0]+prevOst[1]-11)
                .mod(12)
                .midiguido(noteNames: nn)}],
        [cn],
        [(degOst+intOst-11).mod(12).midiguido(noteNames: nn)],
        [nextOst.midiguido(noteNames: nn)]);
    res.do{
        |it,i| netaddr.sendMsg("/ITL/scene/a%".format(i+1), "set",
            "txt", it.join(" "))
    }
}
```

* * *

```
~gmnINScore = {
    | rtm, denom=16, netaddr|
    var midinote=67, dy=8 /*stem size*/, tmp="", res="", score;
    rtm.do{ |dur|
        if (dur/(denom/4) >= 1)
        {
            if (tmp.isEmpty.not)
               { res = res+tmp.tag(\bm, [\dy, dy]); tmp="" };
            res = res+format("\\stemsDown \\noteFormat<'square'>
                \\text<'%', font='Courier', fsize=8pt, color='red',
                 dy=6, dx=4> ", dur)
                    ++ midinote.midiguido(dur/denom)
        }
        {
            tmp = tmp + format("\\stemsUp \\noteFormat<'x'> \\text
               <'%', font='Courier', fsize=8pt, color='red', dy=-2,
                    dx=2> ", dur) ++ midinote.midiguido(dur/denom)
        }
    };
```

```
    // the gmn score
    score = "[ \\meter<'" ++ format("%/%", rtm.sum, denom)
            ++ "'> \\staffFormat<style='1-line'> \\clef<'none'>
                \\bar \\dotFormat<dy=0,dx=0,size=1.2> "
            ++ if (tmp.isEmpty) {res} {res+tmp.tag(\bm, [\dy, dy])}
            ++ " \\bar ]";
    // send to INScore
    netaddr.sendMsg("/ITL/scene/rtm", "set", "gmn", score)
}
```

* * *

Structure of the received OSC messages in SC:

```
#   tag  |  seq   | reldur |  deg   |  int   |  nextcli  |     opt_rtm       | realdur
[ msg.[0], msg.[1], msg.[2], msg.[3], msg.[4],  msg[5..7], msg[8..msg.size-2], msg.last ]
```

* * *

Receiving OSC messages from N3 and sending OSC messages to INScore through SC's functions previously defined (and incidentally where data are interpreted for synthesis and analysis computation):

```
OSCdef(\ostinato,
    {
        arg msg, time, addr, recvport;
        //msg.postln;
        if (msg[1].asInteger != 0)
        {
            ~gmnINScore.(
                msg[8..msg.size-2].asInteger,
                netaddr: ~inscore
            );
            ~harmonyINScore.(
                ~prevOst,
                msg[3].asInteger,
                msg[4].asInteger,
                msg[5..7].asInteger[1],
                ~inscore);
            ~prevOst = [ msg[3].asInteger, msg[4].asInteger ]
        }
        {
            ~inscore.sendMsg("/ITL/scene/rtm", "set", "gmn",
                "[ \\meter<\"\"> \\staffFormat<style=\"1-line\">
                    \\clef<\"perc\"> \\fermata<\"long\">(_/1) |
                    empty*1/32 \\staffOff ]");
            5.do{ |i| ~inscore.sendMsg("/ITL/scene/a%".format(i+1),
                 "set", "txt", "")};
            ~prevOst = nil
        }
    }, '/N3', recvPort:7772
)
```

* * *

Discussion

In fact, the principle of INScore is rather simple as long as you are ready to learn the syntax, which is relatively intuitive. Moreover, JavaScript language for advanced or specific use offers the possibility to solve any requirement. On the Guido engine side, things seem to be limited especially for advanced notation, which can be worked around with MusicXML files or images generated by any third-party application. Anyway, we have to take into account that *it inevitably ends up being a sight reading challenge*. It is certainly not a panacea when it comes to animated notation, but I was seduced by its simplicity and also by the fact that the application met my needs.

Beyond what I have described, INScore allows wide interactivity not only as I did in the sense of third-party application toward INScore, but also to share any kind of data using the same OSC protocol toward any third-party application. All of these are listed in chapters 16 *Sensors* page 63 and 17 *Events and Interaction* page 68 of the document named *INScore OSC Messages Reference v.1.31* (referenced in Resources).

Now, what can be done in the SuperCollider context is a class to well-format the message to send, which can be seen as an extension (or an interface according to the INScore terminology), plus some convenient methods as I did for my work. Despite my modest contribution to what INScore can do, for sure I will use it more and more in that direction. Anyway, I plan to extend the proposition to the technical aspects of the instrument involved like the ocarina and the clarinet, toward *Untitled* #2 ...

* * *

Resources

- INScore message syntax : https://inscoredoc.grame.fr/rsrc/INScoreMessages.pdf
- Guido syntax online : https://guidoeditor.grame.fr/
- Others :
  - `// install SuperCollider package gsa.quark`
    `Quarks.install("https://github.com/yannics/GSA/gsa")`
  - *Neuromuse3* & *Untitled* #2 as study:
    https://github.com/yannics/Neuromuse3/
  - BWV corpus : http://www.titanmusic.com/data.php

Outline of the Standard MIDI File Structure

Go to: [header chunk | track chunk | track event | meta event | system exclusive event | variable length values]

A standard MIDI file is composed of "chunks". It starts with a header chunk and is followed by one or more track chunks. The header chunk contains data that pertains to the overall file. Each track chunk defines a logical track.

```
SMF = <header_chunk> + <track_chunk> [+ <track_chunk> ...]
```

A chunk always has three components, similar to Microsoft RIFF files (the only difference is that SMF files are big-endian, while RIFF files are usually little-endian). The three parts to each chunk are:

1. The track ID string which is four charcters long. For example, header chunk IDs are "`MThd`", and Track chunk IDs are "`MTrk`".
2. next is a four-byte unsigned value that specifies the number of bytes in the data section of the track (part 3).
3. finally comes the data section of the chunk. The size of the data is specified in the length field which follows the chunk ID (part 2).

Header Chunk

The header chunk consists of a literal string denoting the header, a length indicator, the format of the MIDI file, the number of tracks in the file, and a timing value specifying delta time units. Numbers larger than one byte are placed most significant byte first.

```
header_chunk = "MThd" + <header_length> + <format> + <n> + <division>
```

`"MThd"` 4 bytes the literal string MThd, or in hexadecimal notation: 0x4d546864. These four characters at the start of the MIDI file indicate that this is a MIDI file.

<`header_length`> 4 bytes length of the header chunk (always 6 bytes long--the size of the next three fields which are considered the header chunk).

<`format`> 2 bytes **0** = single track file format
1 = multiple track file format
2 = multiple song file format (i.e., a series of type 0 files)

<`n`> 2 bytes number of track chunks that follow the header chunk

<`division`> 2 bytes unit of time for delta timing. If the value is positive, then it represents the units per beat. For example, +96 would mean 96 ticks per beat. If the value is negative, delta times are in SMPTE compatible units.

Track Chunk

A track chunk consists of a literal identifier string, a length indicator specifying the size of the track, and actual event data making up the track.

```
track_chunk = "MTrk" + <length> + <track_event> [+ <track_event> ...]
```

"MTrk" 4 bytes the literal string MTrk. This marks the beginning of a track.
<**length**> **4 bytes** the number of bytes in the track chunk following this number.
<**track_event**> a sequenced track event.

Track Event

A track event consists of a delta time since the last event, and one of three types of events.

```
track_event = <v_time> + <midi_event> | <meta_event> | <sysex_event>
```

<**v_time**> a variable length value specifying the elapsed time (delta time) from the previous event to this event.
<**midi_event**> any MIDI channel message such as note-on or note-off. Running status is used in the same manner as it is used between MIDI devices.
<**meta_event**> an SMF meta event.
<**sysex_event**> an SMF system exclusive event.

Meta Event

Meta events are non-MIDI data of various sorts consisting of a fixed prefix, type indicator, a length field, and actual event data..

```
meta_event = 0xFF + <meta_type> + <v_length> + <event_data_bytes>
```

<**meta_type**> **1 byte** meta event types:

Type	Event	Type	Event
0x00	Sequence number	0x20	MIDI channel prefix assignment
0x01	Text event	0x2F	End of track
0x02	Copyright notice	0x51	Tempo setting
0x03	Sequence or track name	0x54	SMPTE offset
0x04	Instrument name	0x58	Time signature
0x05	Lyric text	0x59	Key signature
0x06	Marker text	0x7F	Sequencer specific event
0x07	Cue point		

<**v_length**> length of meta event data expressed as a variable length value.
<**event_data_bytes**> the actual event data.

System Exclusive Event

A system exclusive event can take one of two forms:

```
sysex_event = 0xF0 + <data_bytes> 0xF7
```

or

```
sysex_event = 0xF7 + <data_bytes> 0xF7
```

In the first case, the resultant MIDI data stream would include the 0xF0. In the second case the 0xF0 is omitted.

Variable Length Values

Several different values in SMF events are expressed as variable length quantities (e.g. delta time values). A variable length value uses a minimum number of bytes to hold the value, and in most circumstances this leads to some degree of data compresssion.

A variable length value uses the low order 7 bits of a byte to represent the value or part of the value. The high order bit is an "escape" or "continuation" bit. All but the last byte of a variable length value have the high order bit set. The last byte has the high order bit cleared. The bytes always appear most significant byte first.

Here are some examples:

```
Variable length             Real value
0x7F                        127 (0x7F)
0x81 0x7F                   255 (0xFF)
0x82 0x80 0x00              32768 (0x8000)
```

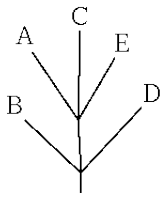
---

craig@ccrma.stanford.edu

The Newick tree format

Introduction

The Newick Standard for representing trees in computer-readable form makes use of the correspondence between trees and nested parentheses, noticed in 1857 by the famous English mathematician Arthur Cayley. If we have this rooted tree:



then in the tree file it is represented by the following sequence of printable characters:

```
(B,(A,C,E),D);
```

The tree ends with a semicolon. The bottommost node in this tree is an interior node, not a tip. Interior nodes are represented by a pair of matched parentheses. Between them are representations of the nodes that are immediately descended from that node, separated by commas. In the above tree, the immediate descendants are B, another interior node, and D. The other interior node is represented by a pair of parentheses, enclosing representations of its immediate descendants, A, C, and E. In our example these happen to be tips, but in general they could also be interior nodes and the result would be further nestings of parentheses, to any level.

Tips are represented by their names. A name can be any string of printable characters except blanks, colons, semicolons, parentheses, and square brackets.

Because you may want to include a blank in a name, it is assumed that an underscore character ("_") stands for a blank; any of these in a name will be converted to a blank when it is read in. Any name may also be empty: a tree like

```
(,(,,),);
```

is allowed. Trees can be multifurcating at any level.

Branch lengths can be incorporated into a tree by putting a real number, with or without decimal point, after a node and preceded by a colon. This represents the length of the branch immediately below that node. Thus the above tree might have lengths represented as:

```
(B:6.0,(A:5.0,C:3.0,E:4.0):5.0,D:11.0);
```

The tree starts on the first line of the file, and can continue to subsequent lines. It is best to proceed to a new line, if at all, immediately after a comma. Blanks can be inserted at any point except in the middle of a species name or a branch length.

The above description is actually of a subset of the Newick Standard. For example, interior nodes can have names in that standard. These names follow the right parenthesis for that interior node, as in this example:

```
(B:6.0,(A:5.0,C:3.0,E:4.0)Ancestor1:5.0,D:11.0);
```

Examples

To help you understand this tree representation, here are some trees in the above form:

```
((raccoon:19.19959,bear:6.80041):0.84600,((sea_lion:11.99700,
seal:12.00300):7.52973,((monkey:100.85930,cat:47.14069):20.59201,
weasel:18.87953):2.09460):3.87382,dog:25.46154);
```

```
(Bovine:0.69395,(Gibbon:0.36079,(Orang:0.33636,(Gorilla:0.17147,
(Chimp:0.19268, Human:0.11927):0.08386):0.06124):0.15057):0.54939,
Mouse:1.21460):0.10;
```

```
(Bovine:0.69395,(Hylobates:0.36079,(Pongo:0.33636,(G._Gorilla:0.17147,
(P._paniscus:0.19268,H._sapiens:0.11927):0.08386):0.06124):0.15057):0.54939,
Rodent:1.21460);
```

```
A;
```

```
((A,B),(C,D));
```

```
(Alpha,Beta,Gamma,Delta,,Epsilon,,,);
```

(Non-)Uniqueness

The Newick Standard does not make a unique representation of a tree, for two reasons. First, the left-right order of descendants of a node affects the representation, even though it is biologically uninteresting. Thus, to a biologist

```
(A,(B,C),D);
```

is the same tree as

```
(A,(C,B),D);
```

which is in turn the same tree as

```
(D,(C,B),A);
```

and that is the same tree as

```
(D,A,(C,B));
```

and

```
((C,B),A,D);
```

Rooted and unrooted trees

In addition, the standard is representing a rooted tree. For many biological purposes we may not be able to infer the position of the root. We would like to have a representation of an unrooted tree when decribing inferences in such cases. Here the convention is simply to arbitrarily root the tree and report the resulting rooted tree. Thus

```
(B,(A,D),C);
```

would be the same unrooted tree as

```
(A,(B,C),D);
```

and as

```
((A,D),(C,B));
```

Widespread use

In spite of this limitation of nonuniqueness the readability of the resulting representation (for trees of modest size) and the ease of writing programs that read it have kept this standard in widespread use.

Its competitors include the NEXUS standard for trees (part of the more general NEXUS standard for phylogeny data sets). However the NEXUS representation of trees is based on the Newick standard -- inside the NEXUS TREES Block you will find ... Newick trees.

A less Newick-based standard is the PhyloXML standard, which is an XML representation using nesting the <CLADE> ... </CLADE> tag pairs instead of parentheses.

Origin

The Newick Standard was adopted 26 June 1986 by an informal committee meeting convened by me during the Society for the Study of Evolution meetings in Durham, New Hampshire and consisting of James Archie, William H.E. Day, Wayne Maddison, Christopher Meacham, F. James Rohlf, David Swofford, and myself. (The committee was not an activity of the SSE nor endorsed by it). The reason for the name is that the second and final session of the committee met at Newick's restaurant in Dover, New Hampshire, and we enjoyed the meal of lobsters. The tree representation was a generalization of one developed by Christopher Meacham in 1984 for the tree plotting programs that he wrote for the PHYLIP package while visiting Seattle. His visit was a sabbatical leave from the University of Georgia, which thus indirectly partly funded that work.

Source: Joseph Felsenstein and Gary Olsen at
http://evolution.genetics.washington.edu/phylip/newicktree.html

Psychoacoustics – Roughness

Roughness – fluctuation strength

Roughness is a complex effect which quantifies the subjective perception of rapid (15-300 Hz) amplitude modulation of a sound. It is hard to describe in words, but below are some sound files that hoepfully will give a sense of what is meant by rapid modulation. The unit of measure is the asper. One asper is defined as the roughness produced by a 1000Hz tone of 60dB which is 100% amplitude modulated at 70Hz [1]. For a tone with a frequency of 1000Hz or above, the maximal roughness of a tone is found to be at a modulating frequency of 70Hz. Maximal roughness is found to be at increasingly lower modulation frequencies when the carrier frequency is below 1000Hz. A just noticeable difference level in roughness is estimated to be 17% [2]. Roughness has been used to partially quantify sound quality in a number of applications including car engine noise, and in some domestic appliances such as electric razors. It has also been used in the calculation of an unbiased annoyance metric.

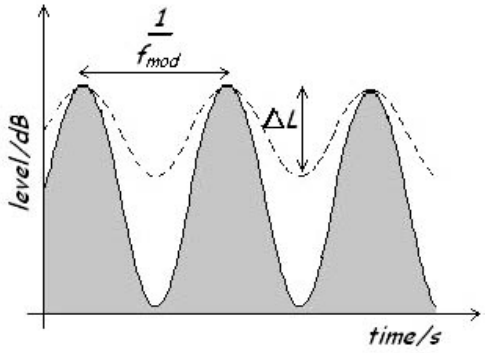


Figure 1: The effect of subjective duration on rapid amplitude modulated noise: (i) the modulation depth (unbroken line) and (ii) the perceived masking depth (dashed line).

In order to begin to construct a model for roughness we describe an amplitude modulated tone as a sound with a rapidly changing loudness level. To gain understanding of the effect on the ear of this rapidly changing level we must first understand the concept of subjective duration.

Subjective Duration

Usually the duration of a sound refers to the objective duration, and for sounds greater than 300 ms in length, this is adequate as the objective measurement and subjective perception are the same. However, as the duration of a sound gets shorter and goes below 300 ms a different subjective effect comes into play. Sounds of shorter durations are perceived to be longer than the objective measurement. For example, a sound of duration 10ms may be subjectively perceived

to be 20 ms long and this has important consequences for the subjective perception of temporarily varying sounds, such as rough sounds.

How does this relate to roughness?

Returning to our amplitude modulated tone we can picture the changing level of the tone as the unbroken line in figure 1. But because the duration of a rapidly changing level appears subjectively to be longer, the level perceived by the ear does not drop as rapidly as the objectively measured level. So the perceived level only drops as low as delta L, indicated by the dashed line in figure 1.

To summarise, this means that the perceived masking depth is smaller than the objectively measured modulation depth. So the roughness of a sound can be evaluated from the following equation:

$$R = cal \cdot \int_0^{24Bark} f_{mod} \cdot \Delta L \cdot dz$$

Where cal is a calibration factor, f_{mod} is the frequency of modulation and ΔL is the perceived masking depth [1].

Because of the difficulty in accurately quantifying ΔL, however, the roughness metric has not yet been standardised and there are several proposed methods of calculation. One method proposed by Aures [3] in 1985 requires the calculation of generalised modulation depths m_i^*. First the signal is filtered into 24 individual 1 Bark wide bands. Next, the envelope of each filtered signal is multiplied by an appropriate weighting function in the frequency domain that gives maximal values at 70 Hz (in accordance with the behaviour of roughness). Then after conversion back to the time domain the r.m.s value of the each resulting time function is divided by the D.C. value of each original filtered signal to give 24 generalised modulation depths. These generalised modulation depths m_i^* are then each multiplied by a value $g(z_i)$ where z_i is the Bark band of the signal. Each resulting value $g(z_i) \cdot m_i^*$ is equivalent to $f_{mod} \cdot \Delta L$ for a particular Bark band so the equation above becomes:

$$R = cal \cdot \sum_0^{24Bark} g(z_i) \cdot m_i^*$$

Another difficult problem to overcome when developing a roughness algorithm is to get it to return low values of roughness for random sound such as white or pink noise. Aures achieved this by using the fact that the sound was divided up into Bark channels. The filtered spectrum within each Bark channel can be calculated using slopes devised by Terhardt [6] as shown in figure 2.

Calculation of the correlation coefficients between the envelopes of adjacent Bark bands gives small values for random sounds but large values if the amplitude modulation in adjacent channels is in phase. These values can be used to

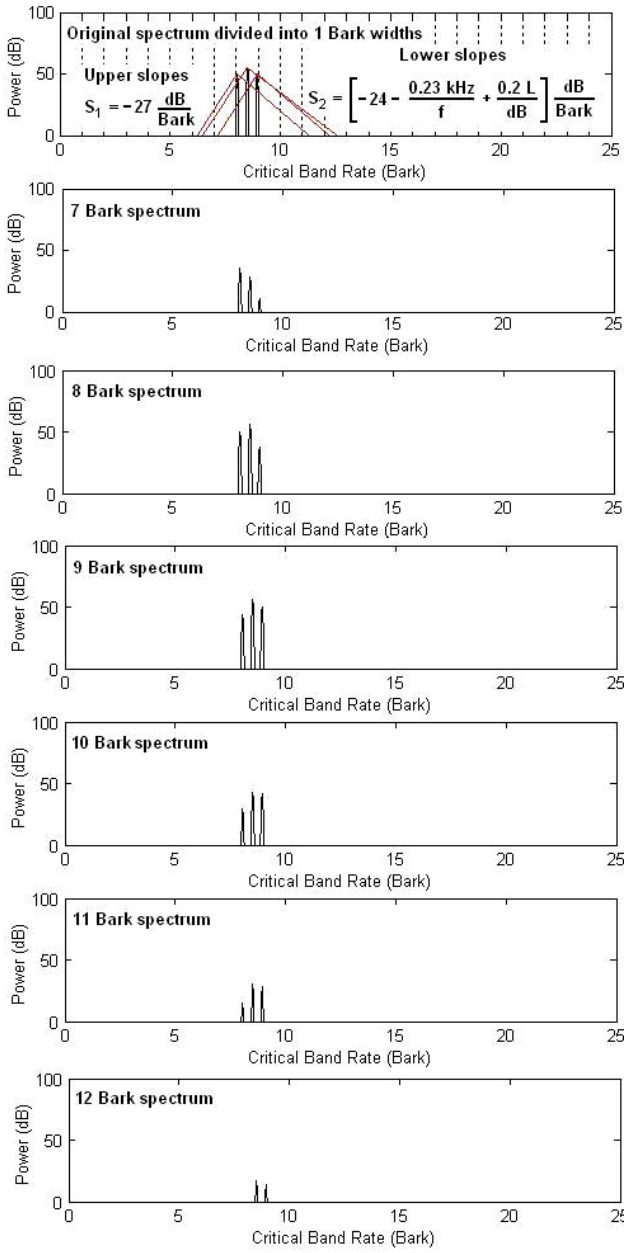


Figure 2: Diagram showing how Bark spectra 7 to 12 are obtained from the original spectrum.

reduce the values of roughness for sounds such as white noise. Daniel and Weber [2] develop these ideas further in their algorithm for calculating roughness.

Widmann & Fastl [4] propose another method for calculation, using a measure of specific loudness made every 2 ms to calculate a time variable course of the masking pattern and from this a value for ΔL can be calculated. Jeong's method [5] is another proposed method of calculation.

Fluctuation Strength

Fluctuation strength is similar in principle to roughness except it quantifies subjective perception of slower (up to 20Hz) amplitude modulation of a sound. The sensation of fluctuation strength persists up to 20Hz then at this point the sensation of roughness takes over. There is a fuzzy border at the change over of the two sensations when it is difficult to precisely quantify one or the other.

Fluctuating sound Not fluctuating sound Broad band noise Amplitude modulated white noise White noise Tonal noise Amplitude modulated 1000 Hz tone 1000 Hz tone

The unit of measure for fluctuation strength is the vacil. One vacil is defined as the fluctuation strength produced by a 1000Hz tone of 60dB which is 100% amplitude modulated at 4Hz. Maximal values are found to occur at a modulation frequency of 4 Hz. The following relation given by Fastl [1] shows the variation of fluctuation strength F with masking depth ΔL, and modulation frequency f_{mod} :

$$F = \frac{0.008 \cdot \int_0^{24Bark} \Delta L \cdot dz}{\left(\frac{f_{mod}}{4Hz}\right) + \left(\frac{4Hz}{f_{mod}}\right)}$$

It is important to note that ΔL represents the masking depth. This is not the same as the modulation depth, in this case, because of short term memory effects rather than post masking effects. Fluctuation strength has been used in applications such as to calculate an unbiased annoyance metric.

References

[1] Zwicker E., Fastl H. *Psychoacoustics: Facts and Models* (1990).
[2] P. Daniel and R. Weber, *Psychoacoustic Roughness: Implementation of an Optimized Model*, Acustica 83, 113∼123 (1997).
[3] W. Aures, *Ein Berechnungsverfahren der Rauhigkeit* ('A Procedure for Calculating Auditory Roughness') Acustica 58, 268∼281 (1985).
[4] U. Widmann and H. Fastl, *Calculating roughness using time-varying specific loudness spectra*, Proc. Sound Quality Symposium 98, 55∼60 (1998).
[5] H. Jeong, *Sound quality analysis of nonstationary acoustic signals*, Ph. D. Thesis, Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), 1999.
[6] E Terhardt *On the perceptions of periodic sound fluctuation (Roughness)* Acustica 30, 201, (1974).

Source: Sophie Maluski – University of Salford Manchester
https://hub.salford.ac.uk/sirc-acoustics/psychoacoustics/sound-quality-making-products-sound-better/an-introduction-to-sound-quality-testing/roughness-fluctuation-strength/

Acoustics of Tube Models (2): Kelly and Lochbaum method

1. *n*-section models of the vocal tract

The actual profile of the vocal tract bears little resemblance to the two-tube models considered in the previous lecture. For example, figure 1 shows the actual volume profile of a male speaker's vocal tract articulating the vowel [æ]. (The figure is computed from MRI scans, and was obtained from Brad Story's website.)

Figure 1: Vocal tract profile of [æ].

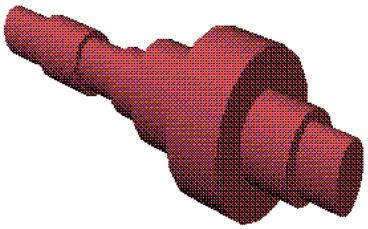


Figure 2: Eight section tube model of vocal tract.

In an attempt to model such profiles more accurately, we can examine the resonances of many concatenated tubes. Figure 2 shows an arrangement of eight tubular sections, and figure 3 the area function of that model:

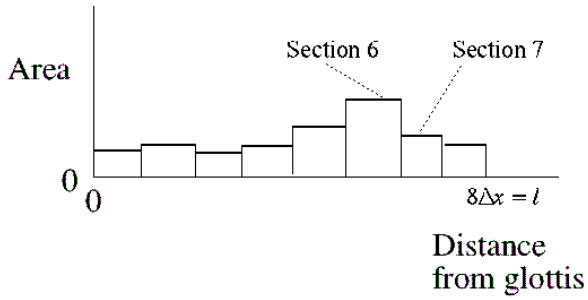


Figure 3: Pressure waves in a 2-section model.

2. Pressure waves in a 2-section model

We consider the transmission of a wave travelling from the glottis to the lips. Also, because of the steps in the area function, some of the energy is reflected back down the vocal tract. There is also a net 'echo', a wave travelling from the lips back to the glottis. Consequently, the pressure P in any section is considered to be made up of two components, a forward wave P^+ and a backward wave P^- (figure 4). Thus, $P = P^+ + P^-$.

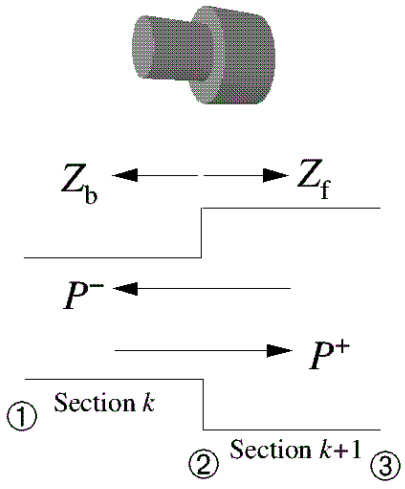


Figure 4: The interface of two tube sections.

Consider what happens at the interface between one tube section and the next (e.g. sections 6 and 7 in figure 3). Each tube section has a characteristic impedance (akin to a resistance to airflow). In figure 4 the impedance forward of the interface (point 2) is labelled Z_{f} and the impedance of the section behind the interface is labelled Z_{b}. The forward wave is partly propagated and partly reflected. The reflected fraction rP^{+} now becomes part of the backward wave, and the propagated part continues forward. The factor r is called the reflection coefficient, and is defined as:

(1) $\quad r = (Z_{\mathrm{f}} - Z_{\mathrm{b}})/(Z_{\mathrm{f}} + Z_{\mathrm{b}})$

Note that r is between -1 and 1. At the interface, P^{+} is continuous, hence the propagated fraction must be $(1+r)P^{+}$, so that:

(2) $\quad P^{+} = (1+r)P^{+} - rP^{+}.$

Exactly similar considerations apply to the backward wave P^{-}, except that since it travels in the opposite direction Z_{f} and Z_{b} are reversed. Thus the reflection coefficient to the backward wave is:

(3) $\quad (Z_{\mathrm{b}} - Z_{\mathrm{f}})/(Z_{\mathrm{b}} + Z_{\mathrm{f}}) = -r$

The propagated part of P^{-} is therefore $(1-r)P^{-}$, and the reflected part is $-rP^{-}$. In this case the reflected fraction adds to the forward going wave. This situation is illustrated in figure 5:

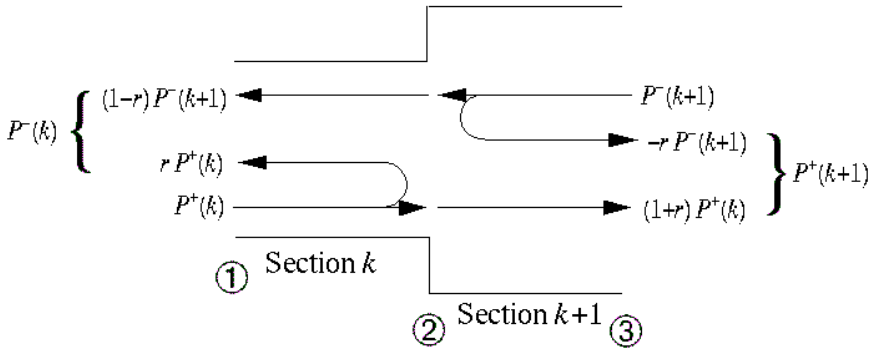


Figure 5: Reflection relationships at an abrupt discontinuity in an acoustic tube.

If there are no losses in the tube the acoustic impedances Z_{f} and Z_{b} are simple functions of the cross-sectional area $A(x)$: $Z_{\mathrm{f}} = DV/A_{\mathrm{f}}$ and $Z_{\mathrm{b}} = DV/A_{\mathrm{b}}$, from which we can work out that:

(4) $r = (A_{\mathrm{b}} - A_{\mathrm{f}})/(A_{\mathrm{b}} + A_{\mathrm{f}})$.

(In practise, we can also obtain the reflection coefficients by using linear prediction software, such as the xwaves program *refcof*.)

3. Pressure waves in an *n*-section model

In a model with n sections, where there are multiple discontinuities, there will be multiple reflections. It is therefore necessary to consider the pressures at discrete points in time, in other words, to consider the signal digitally. For the tube in figure 5, the P^+ wave travelling from left to right will propagate undisturbed from point 1 until it reaches the discontinuity at point 2. Similarly, the backward wave P^- will propagate from point 3 without reflecting until it gets to point 2. If the tube sections are of the same length, l, the time taken for P^+ to go from point 1 to point 2 is equal to the time it takes for P^- to go from point 3 to point 2, i.e. $T = l/\mathrm{c}$, where c is the velocity of sound.

Thus the pressure variations at point 2 (and every other point) only need to be calculated at discrete time intervals, multiples of T. The sample interval T must be less than or equal to half of the highest frequency that it is desired to generate. For speech, the highest frequency of interest is about 6 kHz. Sampling rates of 11025 Hz and 16000 Hz are widely used for medium-quality digital recordings of speech. The sample intervals in these two cases are about 0.000091 s (91 μs) and 0.000063 s (63 μs) respectively.

The maximum section length l is also dependent on the sampling rate, since $T = l/\mathrm{c}$. For a sampling rate of 11025 Hz, the maximum section length is 0.03084 m; for a vocal tract of length 0.175 m, a minimum of six tube sections will be required. WIth a sampling rate of 16000 Hz, the maximum section length is 0.02125 m, necessitating nine tube sections in the model.

In order to calculate the pressure at any of the junctions in the tube model (and in particular at the 'mouth' end), it is necessary to know the pressure wave at the glottis (the closed end of the tube), expressed as a digital signal at time intervals T from 0 until some time later time nT, the end of the utterance to be generated. A counter is used to increment T to 1, and the forward and backward pressures calculated at each interface, according to the reflection coefficients and the equations given in section 2. This process is repeated over and over again, until nT is reached. The forward and backward pressures in each section, will change from sample to sample. The forward wave at the mouth end, e.g. $P^+(9)$ for a nine-section model, will model the sound wave coming out of the mouth.

References

Kelly, J. L. Jr. and C. C. Lochbaum (1962) Speech Synthesis. In *Proceedings of the Stockholm Speech Communication Seminar.* Reprinted in J. L. Flanagan and L. R. Rabiner, eds. (1973) *Speech Synthesis.* Stroudsberg, PA: Dowden, Hutchinson and Ross. 127-130.

Linggard, R. (1985) *Electronic Synthesis of Speech.* Cambridge University Press. 61-4.

---

http://www.phon.ox.ac.uk/jcoleman/kelly-lochbaum.htm

See also:
Smith, Julius O. 'Singing Kelly-Lochbaum Vocal Tract', in *Physical Audio Signal Processing*, online book, 2010 edition.
https://ccrma.stanford.edu/~jos/pasp/Singing_Kelly_Lochbaum_Vocal_Tract.html

BLÅSTJERNEHAV FAMILIE- & DUKKETEATER

Presentation

The project *Blåstjernehav Familie- & Dukketeater* proposes two performative modes:

1. The play itself – once a week for instance – where the kids play in the boat-carrousel with interactive quadrophonic soundscape for their families and audience;
2. and the stage of the play as a fjord-landscape '3D-tableau' installation with experimental sonic sculpture.

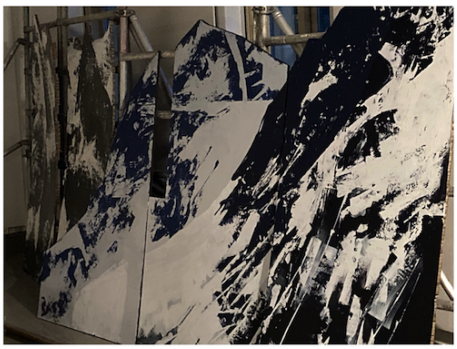
The following documentation shows a possible play, and it is best to visualize it as a sketch, showing how the theatre was played and exhibited in Tromsø Kunstforening from the 14th of August to the 11th of October 2020.

Selenes Havbrev

Ly til fantasi og fellesskap 14.08.2020 – 11.10.2020

→ http://www.tromsokunstforening.no/Ly_til_fantasi_og_fellesskap

[...] *Forestillingen 'Selenes havbrev' og 'Selenes sang' er skrevet av Ane Elene Johansen, og maleriene er laget av Asbjørn Hillingseter Løyning. Yann Ics har komponert musikk og lydbilde, og scenografien er laget av Håvard Arnhoff som ogsåer initiativtakeren bak teateret.*

1. **The play** → pages 185 to 188.





2. **On the side**

   → Study of a sonic sculpture on two recycled corrugated sheets as installation from the 22nd of August to the 29th of August 2020 in Tromsø Kunstforening.

This installation performed the score of the second guitar of the composition *Selenes Havbrev* [→ page 184] with the digital wave guide physical model of a bowed instrument plus improvised composition.

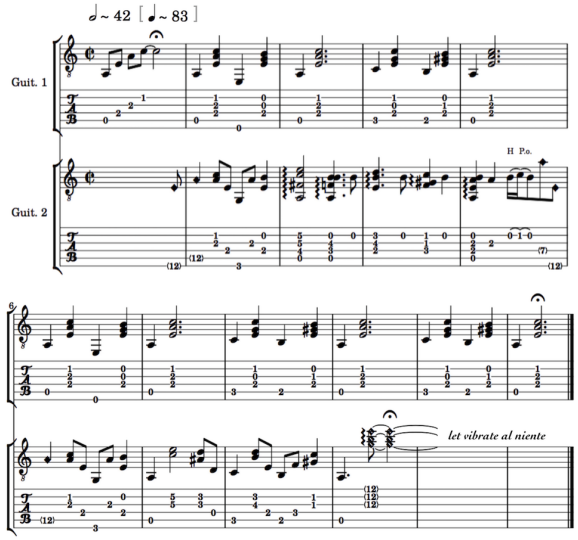
→ Surrealistic sound installation performed in Tromsø Kunstforening between the 25th of September and the 11th of October 2020.

Experimenting different physical models – as a bowed instrument and the acoustics of tube models – using respectively a cello and a parabolic antenna, as part of a 'post-apocalyptic' ambient quadraphonic soundscape which the latest can be depicted as a 'walking on the *Ersfjordbotn* beach'.

Selenes Havbrev

Gitarduett

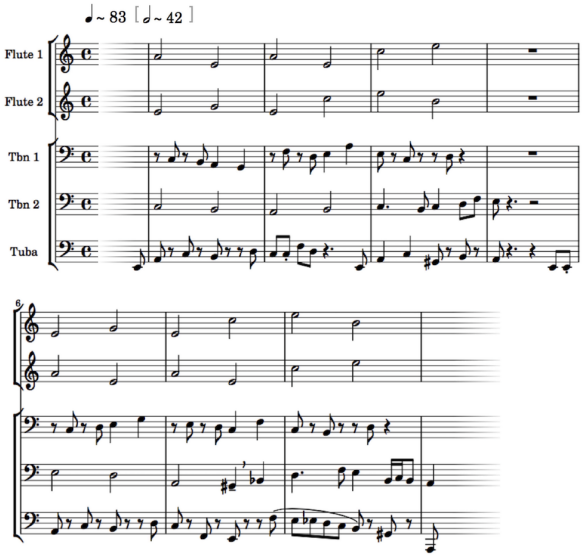
fra temaet Selenes Sang av Ane Elen Johansen



Dynamikken overlates til utøvernes skjønn og kan være fra ***piano*** til ***forte***, spesielt for fanfarene.
En gjentakelse kan gjøres fra opptakt i fra takt en til takt ni.
Tempo er rent veiledende og bør ikke være en begrensning.
Fanfaren kan utvides fritt med improvisasjon eller enhver form for tilpasning etter til utøvernes skjønn.

Fanfaren

variasjon



Selenes Havbrev

av Ane E. Johansen.

En fjord. Fjære. En båt.
Ei jente kjem fram av båten. Kravlende ut, som ei forsiktig krabbe. Stiller seg fram påscenen og proklamerer:

Marikkel : I natt hadde æen drøm. Tre lysende stjerner såvi påvårsolas himmel i natt. Det betyr at selene kjem med Havbrevet. Æog mi søster skal ståklar og ta imot!

Sardina : Han pappa sover inne i huset enda. Han treng søvnen sin no, for ho mamma har kommet til himmelen og han pappa er fryktelig lei seg.

Samtidig som Sardina sier dette, går Marikkel og setter seg i båten og venter. Hun tar påseg en sydvest og begynner åordne til noe fiskesaker.
Sola glinser i horisonten. Lyset er vidunderlig. Det skinner påfjellsidene.

Sardina : Marikkel, har du sjekka nisa om den e klar?

Marikkel : Jada, vi har alt påstell. Trur du vi ska fare med én gang?

Sardina : Best åikkje vente for lenge. Ædrømte at vi måtte være klar påstiganes flo!

Marikkel : Tenk at vi har drømt det samme! Men med floa kjem vinden..

Sardina : Det e no eller aldri!

Sardina skyver båten ut mot vannet og hiver seg oppi båten. De tar fram årene og begynner åro. De ror og ror seg lengre og lengre utover fjorden. Med ett blåser det opp til storm.

Marikkel : Det blåser opp til storm! Vi måsnu!

De får vansker med åholde årene, og de mister ei åre i bølgene.

Sardina : Ånei, jeg mista åra! Hjelp meg, får du tak i den?

Marikkel : Nei, den forsvinner!

Sardina bøyer seg over ripa og prøver åfåtak i den, og plutselig faller hun ned i vannet.

Sardina : *(roper)* Marikkel!

Uværet fortsetter, og Sardina holder påådrukne. Marikkel forsøker ånåhenne med den ene åra si. Sardina forsvinner lengre og lengre bort fra båten. Marikkel ror etter og prøver ånåhenne igjen. Men det er sterke bølger og hardt åro.

Marikkel : Ta åra! Ta åra!

Sardina griper i panikk rundt åra og Marikkel greier åfåhenne om bord i båten igjen. Sardina klemmer i ett hardt grep rundt Marikkel og da hoster Marikkel og våkner til liv.

Sardina : *(hoster)* Hvor kom det uværet fra?

Marikkel : Flovinden, Sardina! Æsa det til dæ! Vi måbare fortsette åro. Vi er kommet sålangt no, vi vil ikkje klare åro mot bølgan inn til land igjen.

De ror med årene som er igjen gjennom uværet.

Marikkel : Her kan vi gåi land!

De ror seg i land ved ei bukt og drar båten i land og fortøyer den. Sakte gir uværet seg og det blir blank stilla hav.

Sardina : Se, Marikkel! *(Peker påhimmelen)* Tre stjerner påvårsolas himmel. Det var hit vi skulle, uværet ledet oss hit!

Marikkel : Ja, tenk at det blei sånn!

Marikkel : Da er tida inne. *(En liten pause til ettertanke)*

Sola lyser fram igjen og det glitrende landskapet trer fram i lyset.
Plutselig hører de noe i havet som fanger oppmerksomheten deres. Tre seler kjem svømmende mot dem.

Sardina : Der kjem selene *(sier det andektig)*.

Selene holder et tau i munnene sine. Den ene til den andre. Tauet er spunnet av tang. I tauet henger det ei lita kiste. Selene kjem i land i fjærsteiene og danser påsitt fornurlige vis mot jentene. Selene synger:

Selene : *(Selenes Sang)*

Am Em Am
I natt har vi spunnet ei vise
C E Am
Tre stjerner i vårsolelyset
Am Em Am
Vårt brev kjem fra havets kiste
C E Am
Aldri måvi glemme eller miste.

Selene gir kista til jentene. Jentene takker og bukker.

Marikkel : Takk skal dere ha.

Sardina : Hvordan kan en drøm bli sann?

Selene : *(sier i kor)* Drømte dere hva dere skulle gjøre med brevet vi har kommet med?

Marikkel : *(Forundret)* Nei, her stopper drømmen.

Jentene ser påhverandre.

Selene : *(sier i kor)* Aldri måvi glemme eller miste, drømmen om blåstjernehav. Ta dette brevet og gi det til den dere holder mest av.

Såforsvinner selene ut i havets dyp.
Jentene står igjen med den lille kista.

Marikkel : Hva mente selene? Aldri måvi glemme eller miste, drømmen om blåstjernehav. Ta dette brevet og gi det til den dere holder mest av *(repeterer etter selene)*.

Sardina : Nei, si det. Kanskje vi mågi det til den vi stoler mest på i hele verden?

De åpner kista. Der ligger det et blått hjerte av glass. Marikkel tar hjertet varsomt ut av kista og holder det opp mot lyset.

Marikkel : Se, kor vakkert!

Sardina : Ja, se kor det glinse og banke! Et blått hjerte!

De studerer hjertet. Ser det fra alle vinkler, beundrer det.

Marikkel : Vi tar det med hjem til han pappa, såblir han glad.

Sardina : Ja, det gjør vi!

De hiver seg i båten, men oppdager at den begynner åta inn vatn.

Marikkel : Se, søster, båten begynner åta inn vann!

Sardina : Vi kan ikkje komme oss hjem over fjellene!

De ser seg omkring påfjellene rundt. Og stirrer ut i havet.

Marikkel : *(Sier rolig og bestemt)* Da er vi nødt til åro hjem.

De setter båten i havet og begynner åro. Et alvor legger seg over stemninga. De skjønner at dette kommer til åbli farlig.

Sardina : Vi får ro såstødig og fort vi bare kan.

De kommer inn i en god rytme. Nåog da tar Marikkel øsekaret og hiver ut vann fra båten. De ror og de ror. Jo lengre de ror, jo mer måde bruke øsekaret. De ser bakover for åse hvor langt de har igjen.

Marikkel : Å, no mådet ikkje være langt igjen! Båten tar inn mer og mere vann!

Sardina : Vi måbare holde ut! Æhåpe vi rekk det!

Marikkel : *(ser seg bakover og studerer huset og naustet der de bor)* E det ikkje han pappa som står i fjærsteinan der borte?

Sardina : Jo, det e det! Pappa!

Mannen i fjæra vinker med begge armene, og roper tilbake.

Pappa : Sardina og Marikkel!

Mannen hiver seg i en båt og ror i møte med dem. Når båtene møtes, klyver Sardina og Marikkel i pappas båt, og han fortøyer den fast i sin egen.

Sardina og Marikkel : Pappa!

De gir hverandre en stor klem.

Pappa : Jentene mine. E dokker ute påhavet no, med den gamle båten! Den læk, jo! Og såtidlig påmorran mens æsøv. Nei og nei, æhar vært såredd førr kor dokker va!

Marikkel : Pappa, vi, unnskyld! Vi hadde en drøm om at vi måtte påhavet, og såfikk vi ikkje sove. Pappa, vi –

Sardina : *(avbryter)* Pappa, vi har opplevd såmye! Vi har møtt sela og, vi mesta ei åra, det va såmye bølga, og selan ga oss nokka vi ville gje tel dæ-

Pappa : Har selan gjedd dokker nokka, nei, ka dokker førtell-

Marikkel og Sardina : *(i kor, andektig)* Aldri måvi glemme eller miste, drømmen om blåstjernehav. Ta dette brevet og gi det til den dere holder mest av.

Sardina og Marikkel åpner kista av skjell og gir det blåhjertet til pappa.

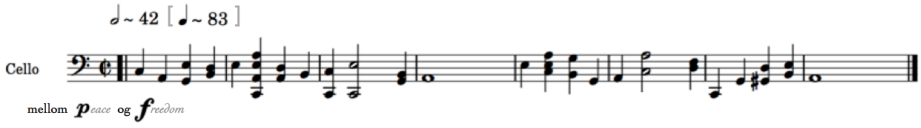
Pappa : *(utbryter)* Åh...! Mammas gamle smykkestein! Kordan...?

Pappa holder det varsomt og stryker det blåhjertet forsiktig. Tårene renner nedover kinnene hans.

Marikkel : E det mamma sin smykkestein- !

Marikkel og Sardina ser påhverandre. De alle ser påhverandre spørrende.
Med ett hører de en fanfare i horisonten. Det er melodien som selene sang. De klemmer hverandre og smiler. Lyset gnistrer og blender.

Post Scriptum



Diwar Les Coet

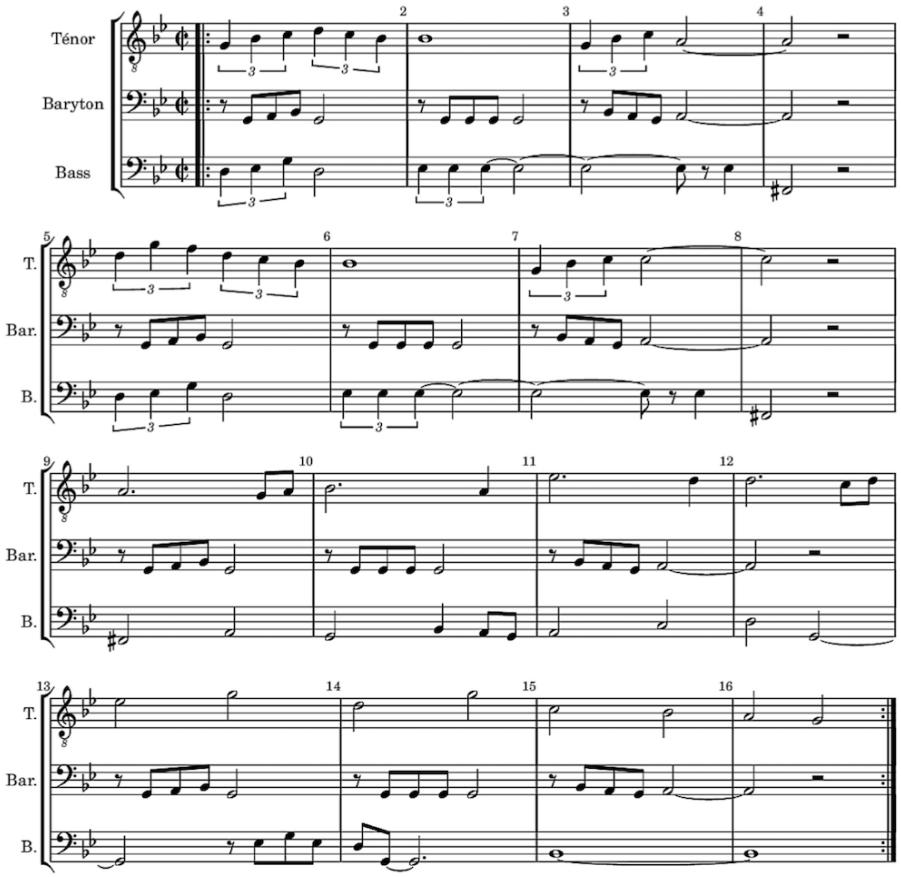
Composition Ouverte Opus 1

𝅗𝅥 = 54 ~ 66 (♩ = 108 ~ 132)

Le tempo est purement indicatif et peut varier significativement et de manière dynamique, pourvu qu'il soit partager par tous les acteurs de la performance, qu'il soit synchronisé ou non.

Choral

Initialement écrit pour 3 voix d'hommes, le choral peut être interprété librement par toutes les voix et adapté par octaviation. Les notes longues peuvent être monnayées ou vocalisées à loisirs selon le (con)texte.

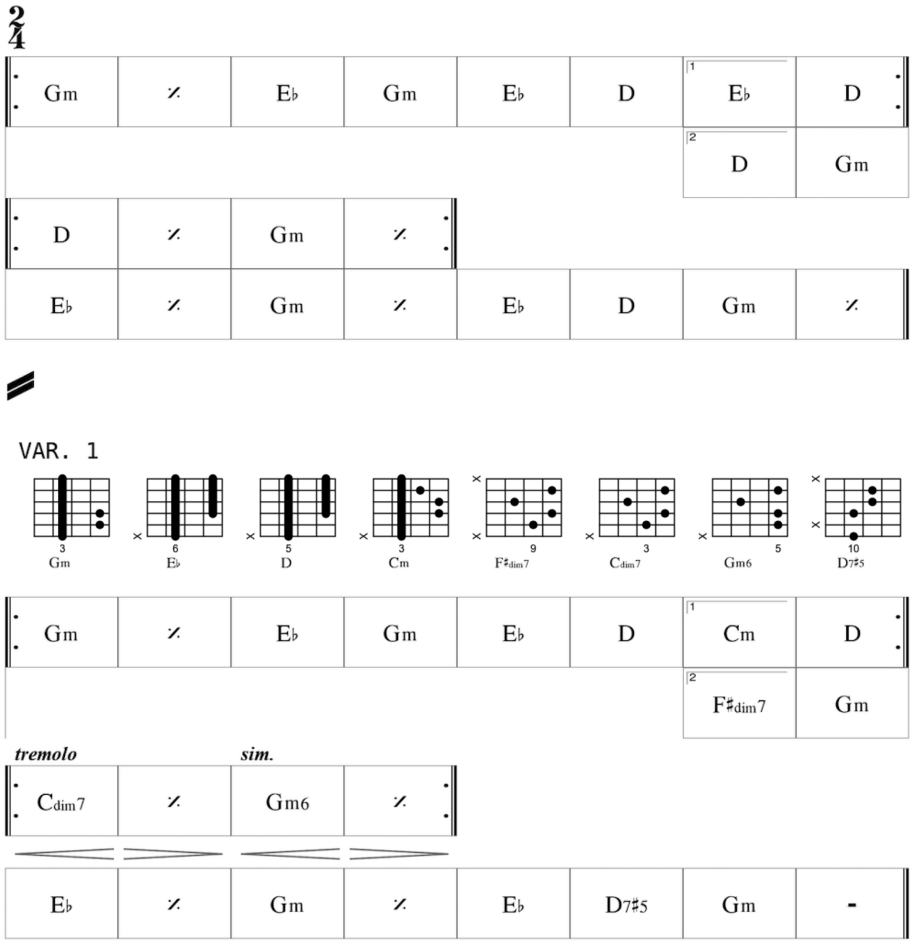


Pour certain passage, une version murmurée ou bouche fermée entre bien entendu dans le domaine du possible.

Voici les combinaisons possibles (1 = ténor, 2 = baryton et 3 = bass):
1– 2 – 3 – 12 – 13 – 23 – 123

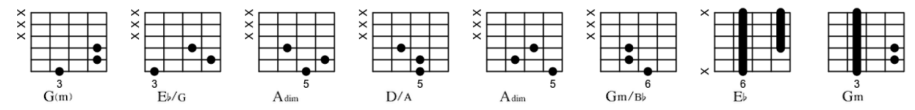
Variations pour Guitares

Les grilles d'accords des guitares sont indicatives et peuvent par conséquent être adaptées librement au jeu du guitariste ou modifiées selon l'interprétation souhaitée. La première grille étant la référence harmonique simplifiée.



Le cas échéant, les grilles harmoniques peuvent être réalisées par n'importe quel instrument ou formation.

VAR. 2

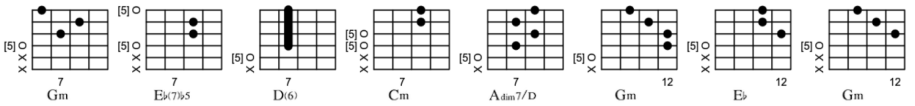


G(m)	𝄎	E♭/G	𝄎	𝄎	Adim	1 𝄎	D/A
						2 Adim	-
Adim	𝄎	Gm/B♭	𝄎				
E♭	-	Gm	-	Gm/B♭	𝄎	𝄎	𝄎

p *fff*

VAR. 3

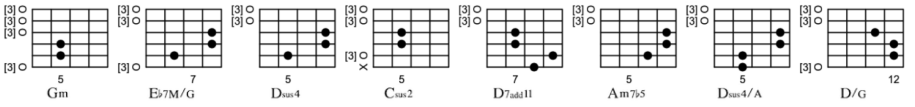
Capodastre 5e case



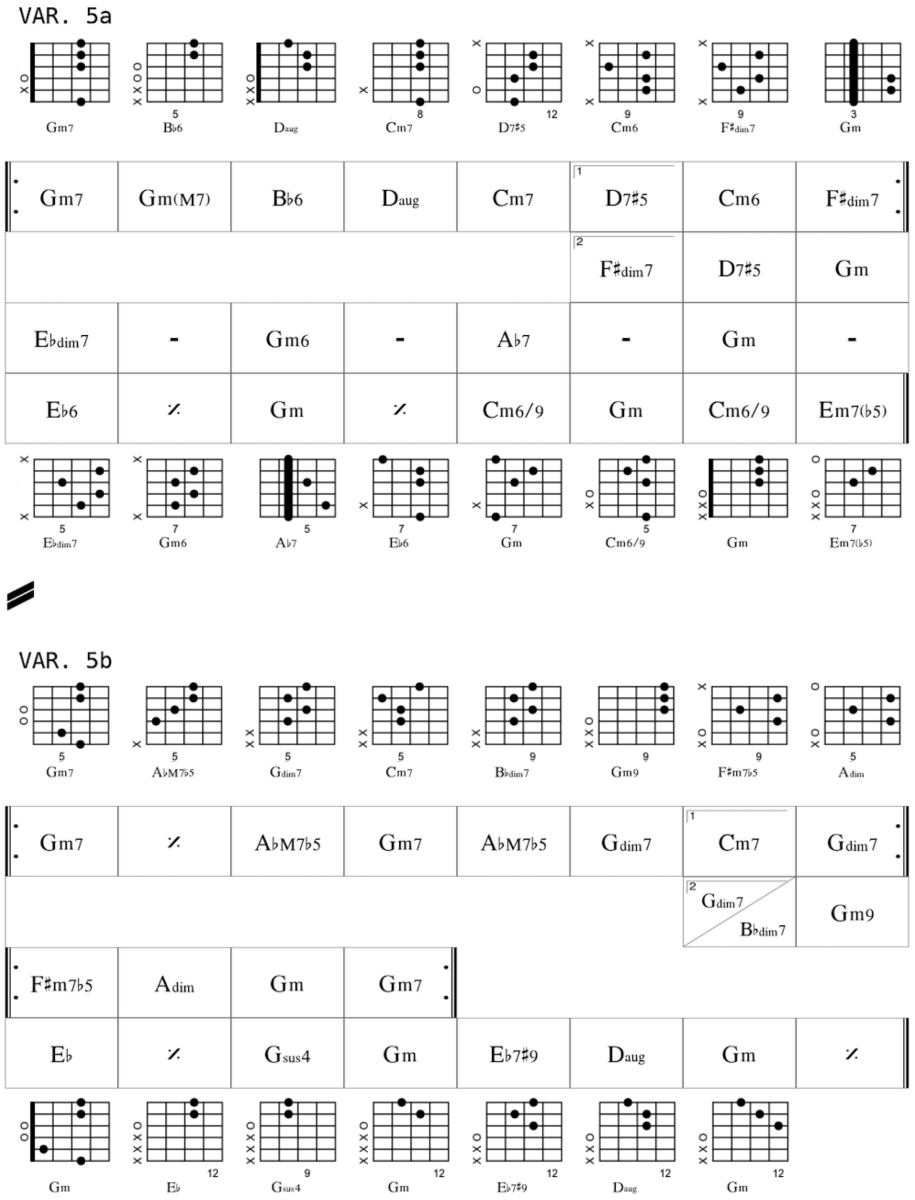
Gm	𝄎	E♭(7)♭5	Gm	E♭(7)♭5	D(6)	1 Cm	D(6)
						2 D(6)	Gm
Adim7/D	𝄎	Gm	𝄎				
E♭	𝄎	Gm	𝄎	E♭(7)♭5	D(6)	Gm	𝄎

VAR. 4

Capodastre 3e case



Gm	𝄎	E♭7M/G	Gm	E♭7M/G	1 Dsus4	Csus2	D7add11
					2 Am7♭5	Dsus4/A	Gm
D/G	𝄎	G(m)	𝄎				
E♭7M/G	𝄎	G(m)	𝄎	E♭7M/G	Gm/D5	Gm	G(m)

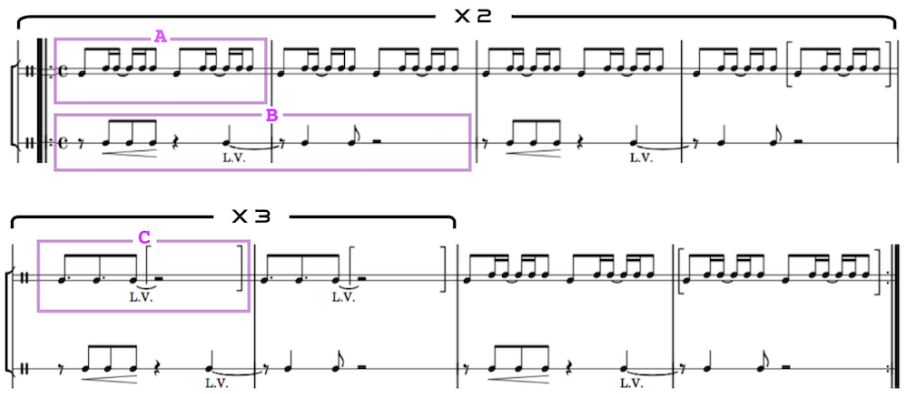


Voici les combinaisons possibles selon les grilles proposées (1 = variation 1, 2 = variation 2, 3 = variation 3 et 4 = variation 4, 5 = variation 5a, et 6 = variation 5b) pour 4 guitaristes:

1 – 2 – 3 – 4 – 5 – 6 – 12 – 13 – 14 – 15 – 16 – 23 – 24 – 25 –26 – 34 – 35 – 36 – 45 – 46 – 56 – 123 – 124 – 134 – 135 – 136 – 234 – 235 – 236 – 345 – 346 – 456 – 1234 – 1235 – 1236 –2345 – 2346 – 3456

Thème Rythmique – djembé/dum-dum

Les percussions sont définies ici – par défaut – par le couple djembé/dumdum, mais peuvent bien entendu être adaptées selon un *instrumentarium* libre et selon des motifs rythmiques autres, mais apparentés.



à l'instar des briques d'une construction, chaque motif rythmique – ici encadré et dénommé A, B et C – est agencé afin d'entrer dans l'élaboration d'une séquence entière, soit 16 mesures.

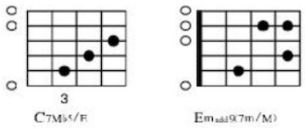
Les parties entre crochets doivent être interprétées en termes d'articulation et/ou de cadence selon le contexte.

Nord

Open Composition Opus 2

Chord diagrams

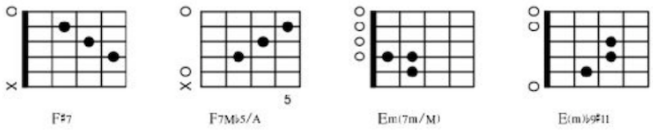
VAR. 1



A	C7Mb5/E	𝄎 (>) 𝄎	𝄎	(>) 𝄎
B	Emadd9 𝄎 +7	𝄎 𝄎 +7M	𝄎 𝄎 +7	Emadd9 (>) 𝄎

```
   (A × n + B × m) × ad. lib.
or (A × n +   -  ) × ad. lib.
or (  -   + B × m) × ad. lib.
or (  -   +   -  ) × ad. lib.
```

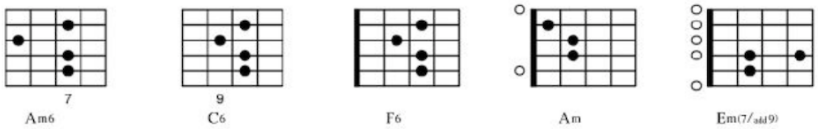
VAR. 2



A_1	F♯7	𝄎 (>) 𝄎	𝄎	(>) 𝄎
A_2	F7Mb9/A	𝄎 (>) 𝄎	𝄎	(>) 𝄎
B	Em 𝄎 +7	𝄎 𝄎 +7M	𝄎 𝄎 +7	Em E(m)b9♯11

```
   ((pA1 + pA2).n/2p + m.B) × ad. lib.
or ((pA1 + pA2).n/2p +  - ) × ad. lib.
or (         -         + m.B) × ad. lib.
or (         -         +  - ) × ad. lib.
```

VAR. 3



The musical phrase **A** (i.e. $q\mathbf{A_1} + q\mathbf{A_2} + q\mathbf{A_3} + q\mathbf{A_4}$) is played ***tremolo***.

$\mathbf{A_1}$	Am6	𝄎	𝄎	𝄎
$\mathbf{A_2}$	C6	𝄎	𝄎	𝄎
$\mathbf{A_3}$	F6	𝄎	𝄎	𝄎
$\mathbf{A_4}$	Am	𝄎	𝄎	𝄎
B	Em	Em7	$\mathrm{Em_{add9}}$ Em	𝄎

$((q\mathbf{A_1} + q\mathbf{A_2} + q\mathbf{A_3} + q\mathbf{A_4}).n/4q + m.\mathbf{B}) \times ad.\ lib.$
or $((q\mathbf{A_1} + q\mathbf{A_2} + q\mathbf{A_3} + q\mathbf{A_4}).n/4q + \text{-}) \times ad.\ lib.$
or $(\text{-} + m.\mathbf{B}) \times ad.\ lib.$
or $(\text{-} + \text{-}) \times ad.\ lib.$

Interlude

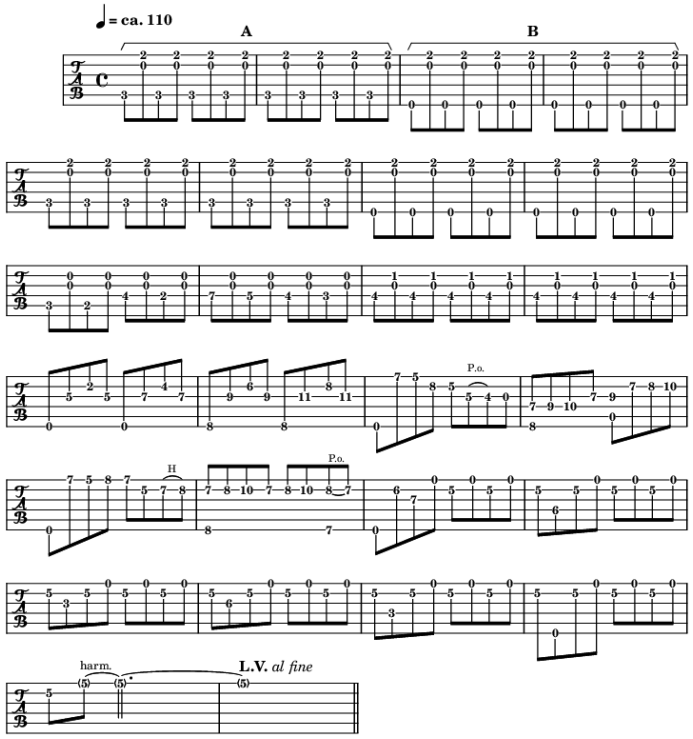
Interludes are the part of Nord written in tablature for two guitars. In fact, there are two distinct parts, therefore two interludes, but which are thematically interconnected. So there is an interlude called *Nomad* composed of three movements, the last movement of which is played as a soloist; and a more autonomous interlude called *Monkey*, which can be played repeatedly, and even mixed in at any moment during the performance.

The cells marked A and B (see also A' and B' for the *Monkey* interlude) in the scores can be played *ad libitum*, being able to serve as digressions or to support a possible interaction depending on the context. Along the same lines, the use of a wind instrument is particularly encouraged.

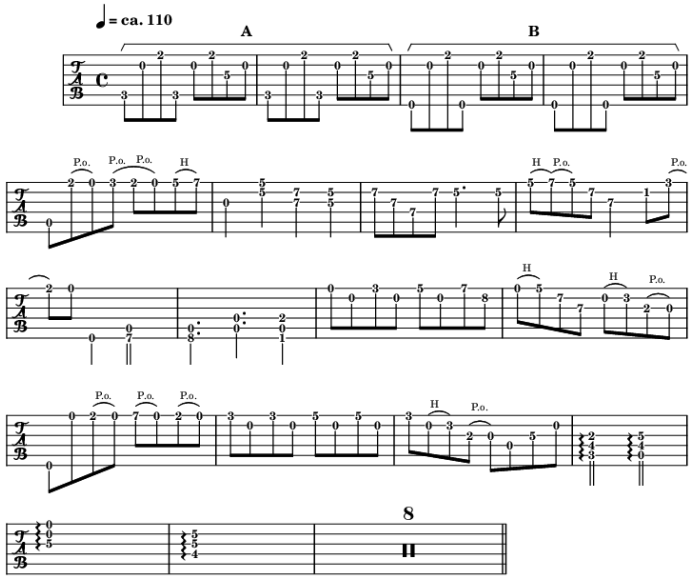
The dynamic aspect must be considered according to the interpreter's feelings. Last, all notes and open strings must resonate as long as possible, more or less understood like a *legatissimo*.

Nomad Movement 1

Guitar 1

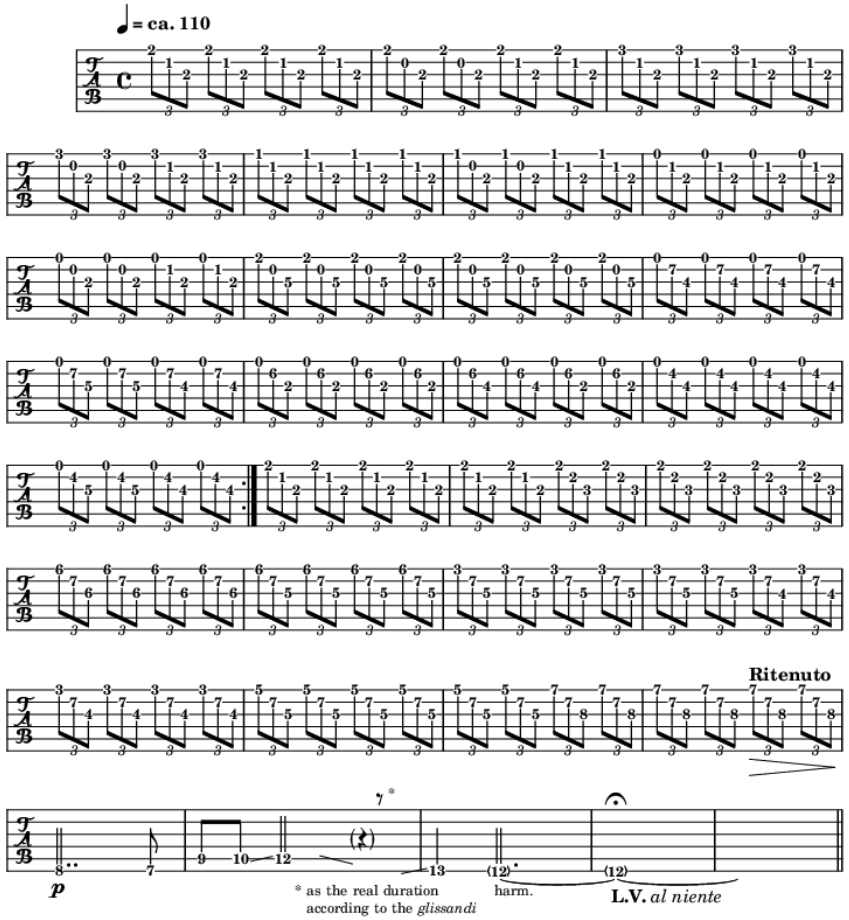


Guitar 2

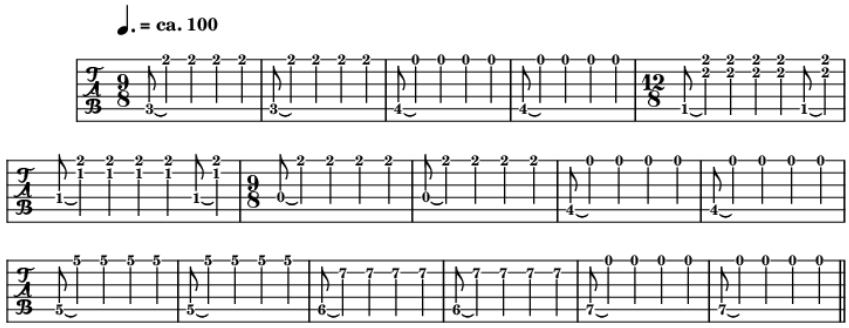


Nomad Movement 2

Guitar 1

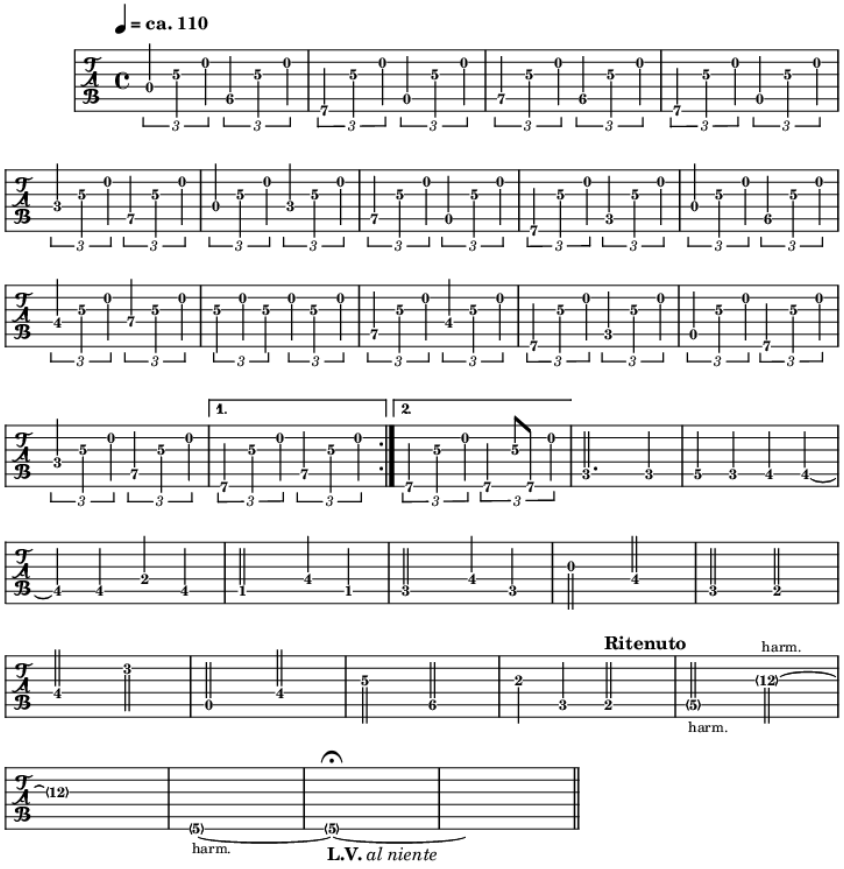


bearing – can be played as a loop anytime but not here.

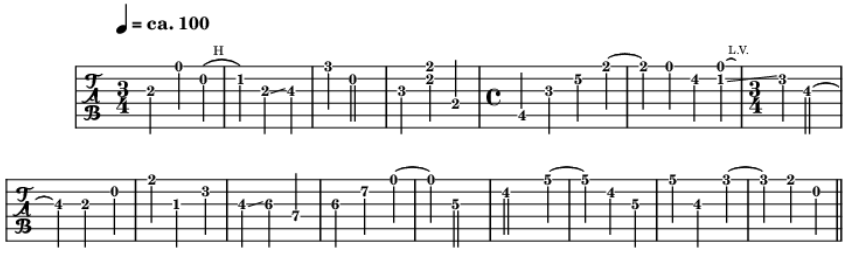


Nomad Movement 2

Guitar 2

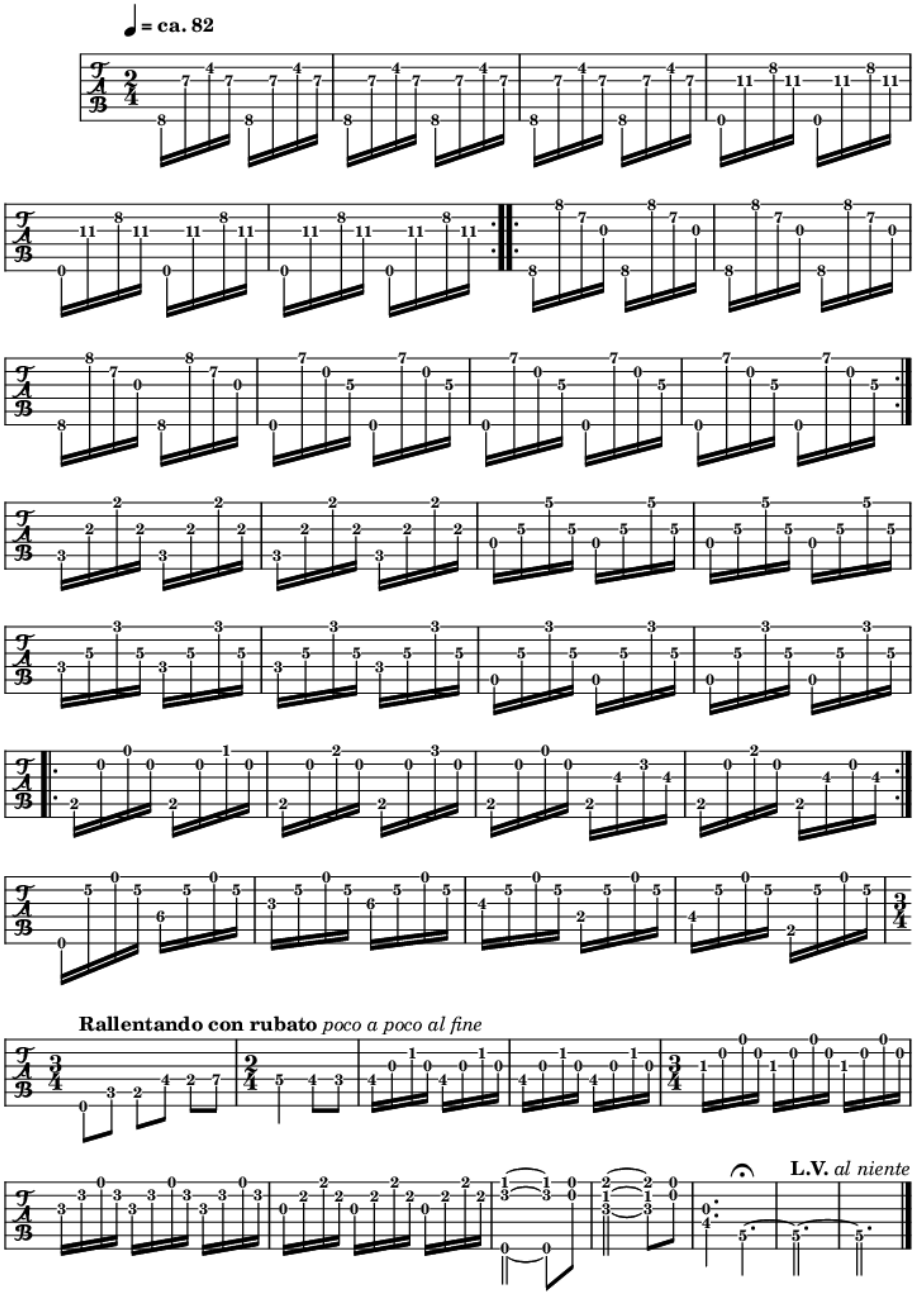


bearing – can be played as a loop anytime but not here.



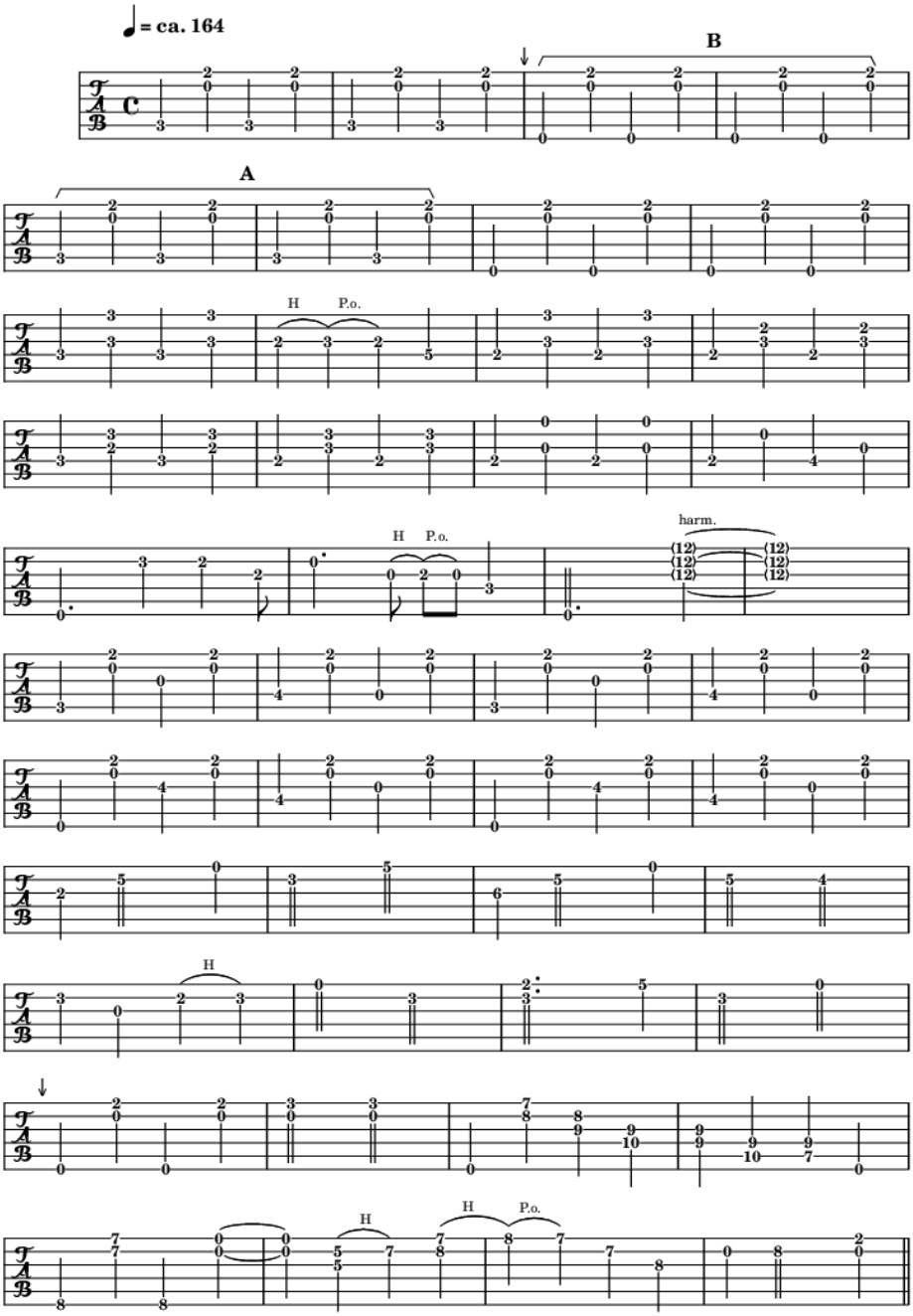
Nomad Movement 3

Guitar solo



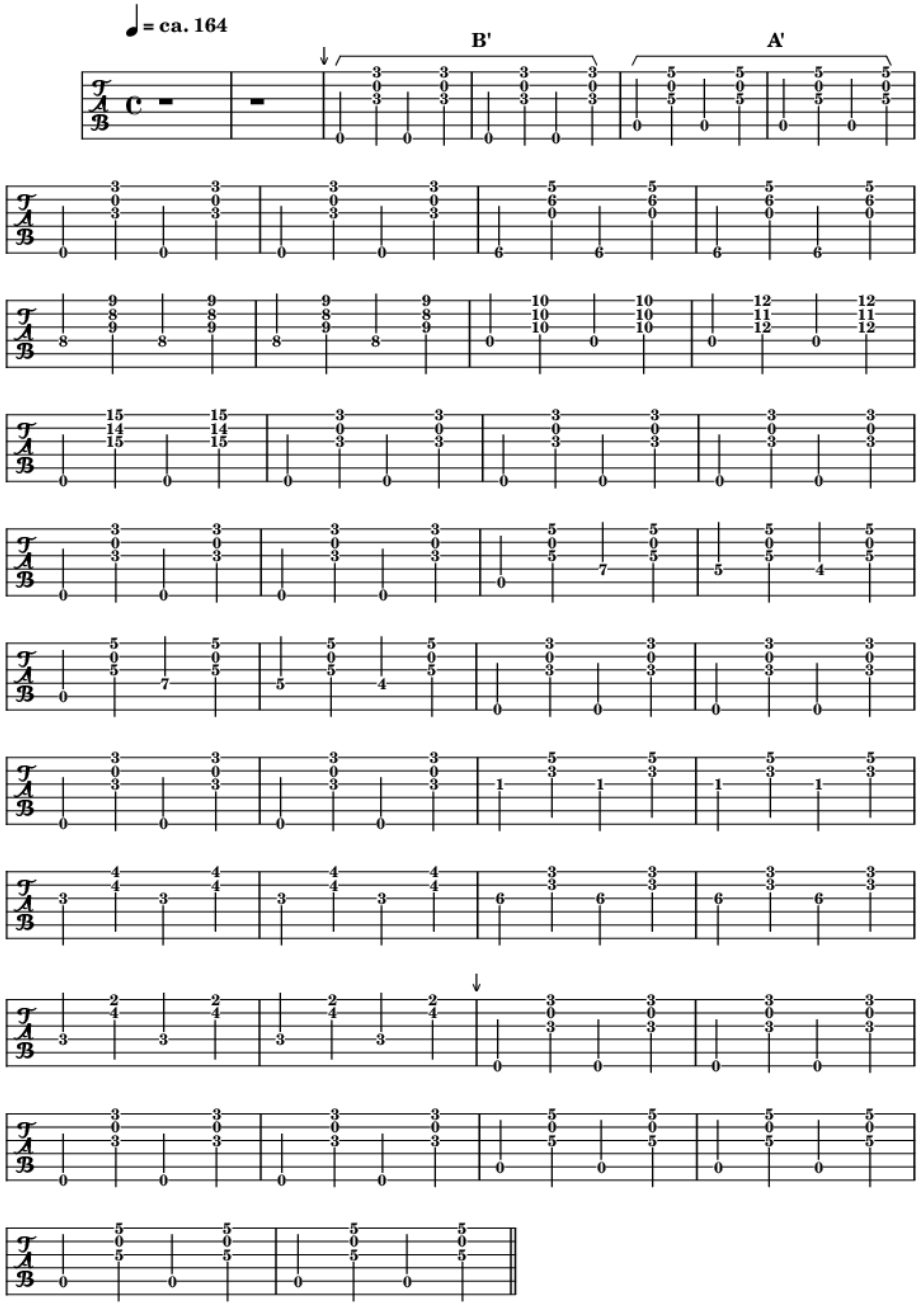
Monkey

Guitar 1



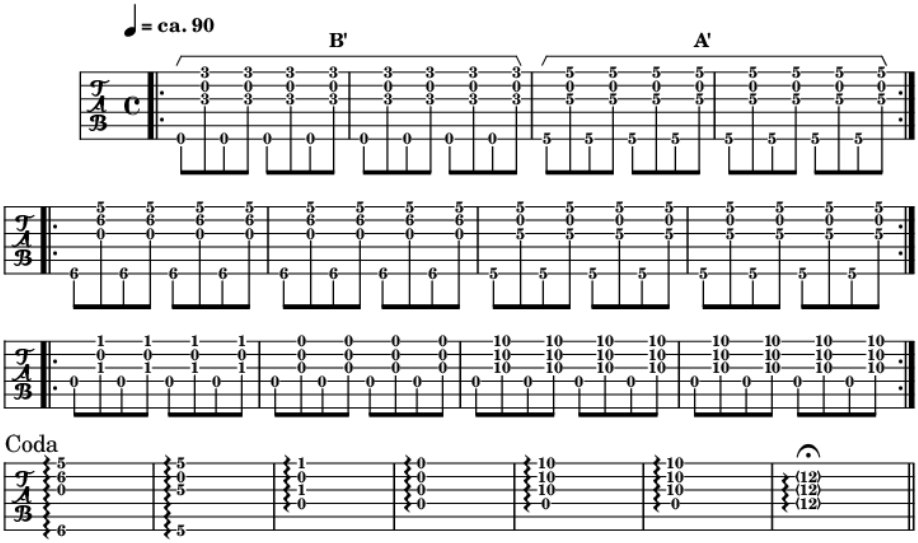
Monkey

Guitar 2

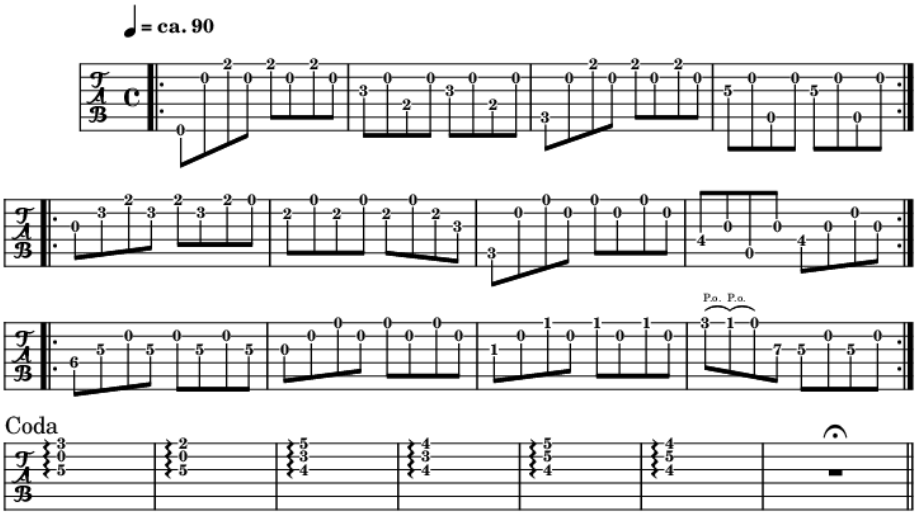


Monkey song

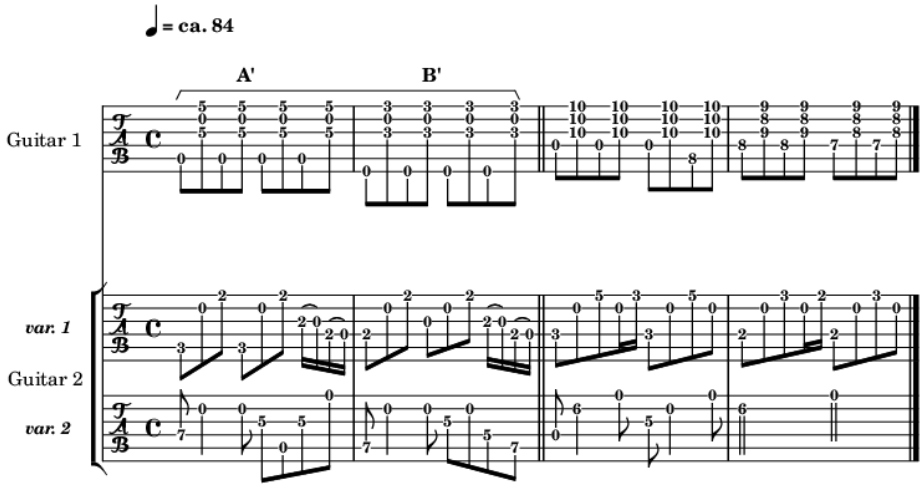
Guitar 1



Guitar 2



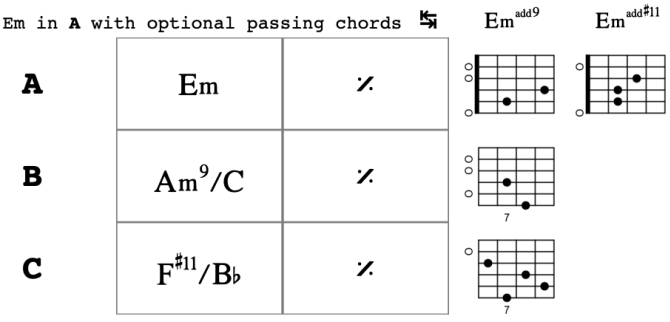
Monkey song



VAR. 4

A and **B** refers to the cells of **Interludes.**

Em in **A** with optional passing chords



Kjølhiea – eller hvor du er

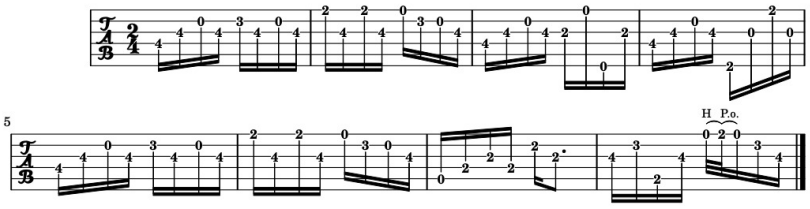
Open Composition Opus 3

for 4 guitars

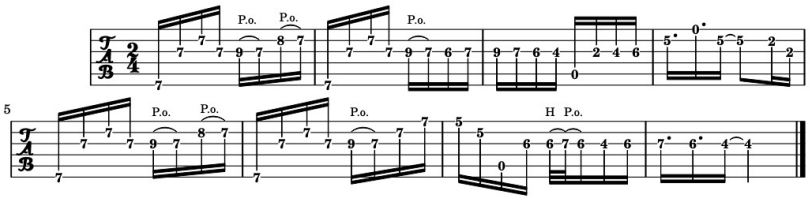
This is the basics. You have to find your own harmony playing with dynamics and rhythms.

♩ ~ 50 ~

Morgen bønn

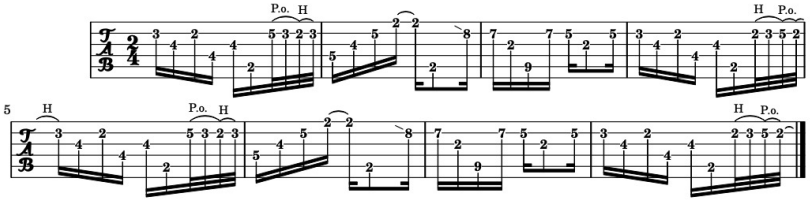


Sunrise



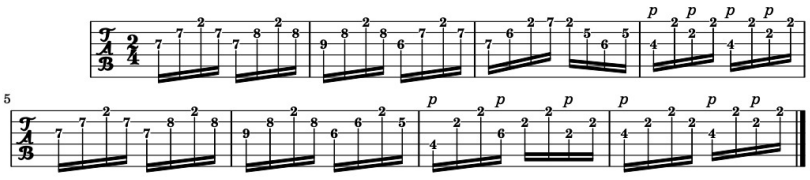
Marche

Capodastro *second fret*



Free Spirit

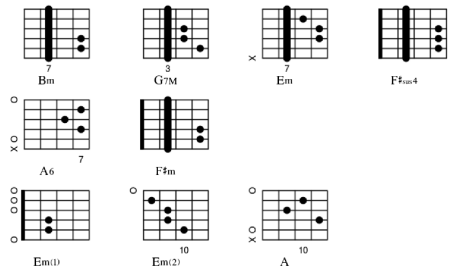
Capodastro *second fret*



Chord diagrams

Bm	𝄎	𝄎	𝄎
1 Bm	A	Bm	𝄎
2 A	𝄎	Bm	𝄎

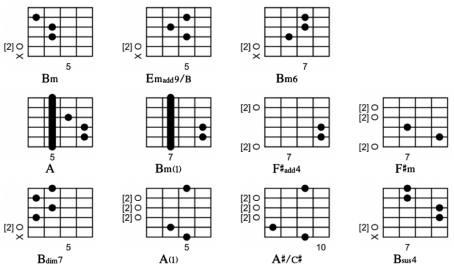
var. 1



Bm	G7M	Em	F♯sus4
Bm	A6	F♯m	𝄎
Bm	G7M	$Em_{(1)}$	$Em_{(2)}$
A6	A	Bm	𝄎

var. 2

Capodastro 2nd fret



Bm	$Em_{add9/B}$	Bm6	$Em_{add9/B}$
Bm	A	$Bm_{(1)}$	$F♯_{add4}$ / F♯m
Bm	$Em_{add9/B}$	Bm6	B_{dim7}
$A_{(1)}$	A/C♯	B_{sus4}	𝄎

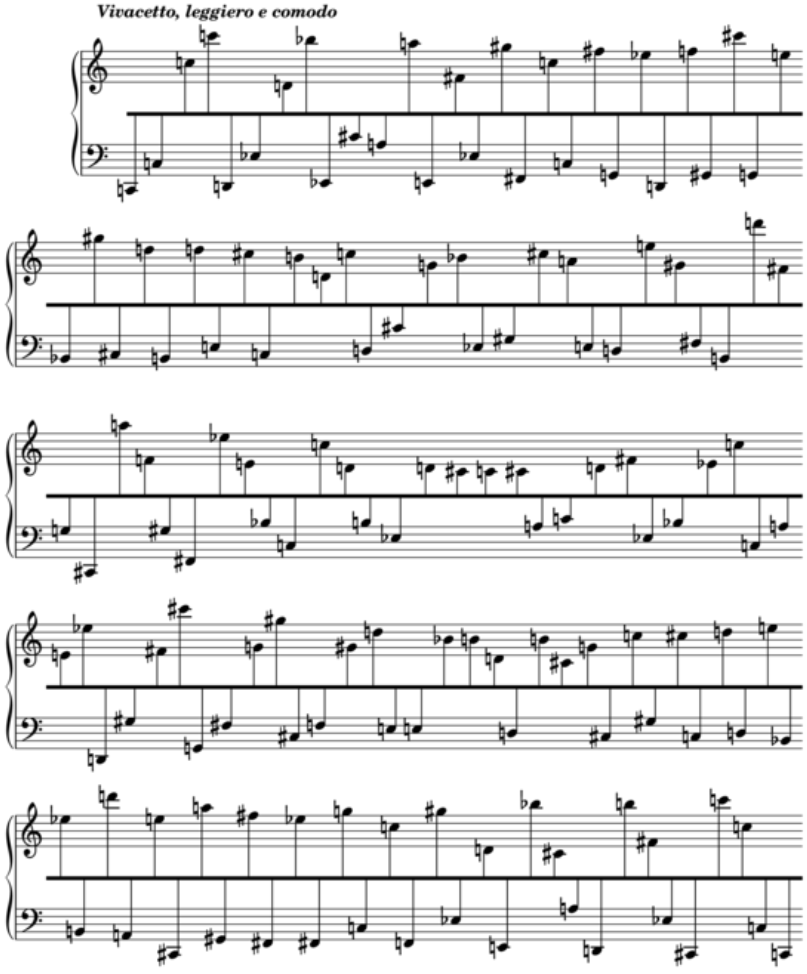
Études

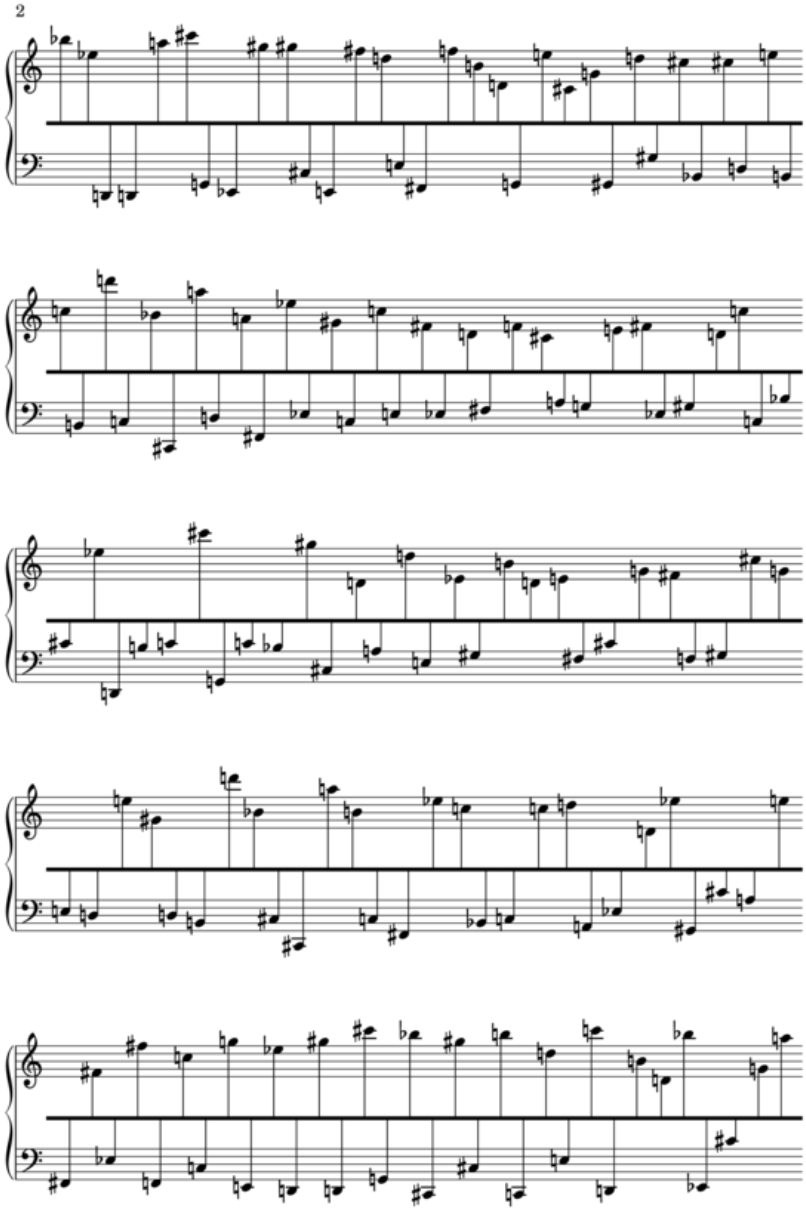
N°1 Entrelacs

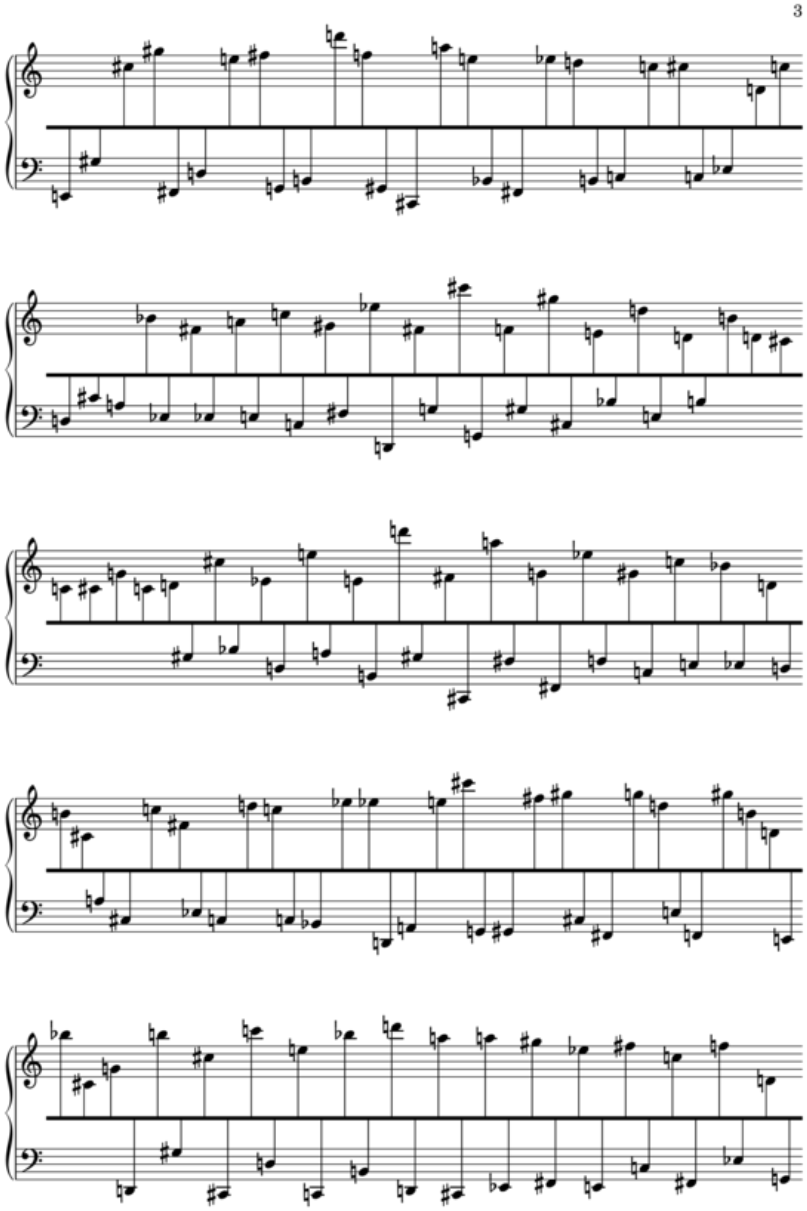
clavicorde

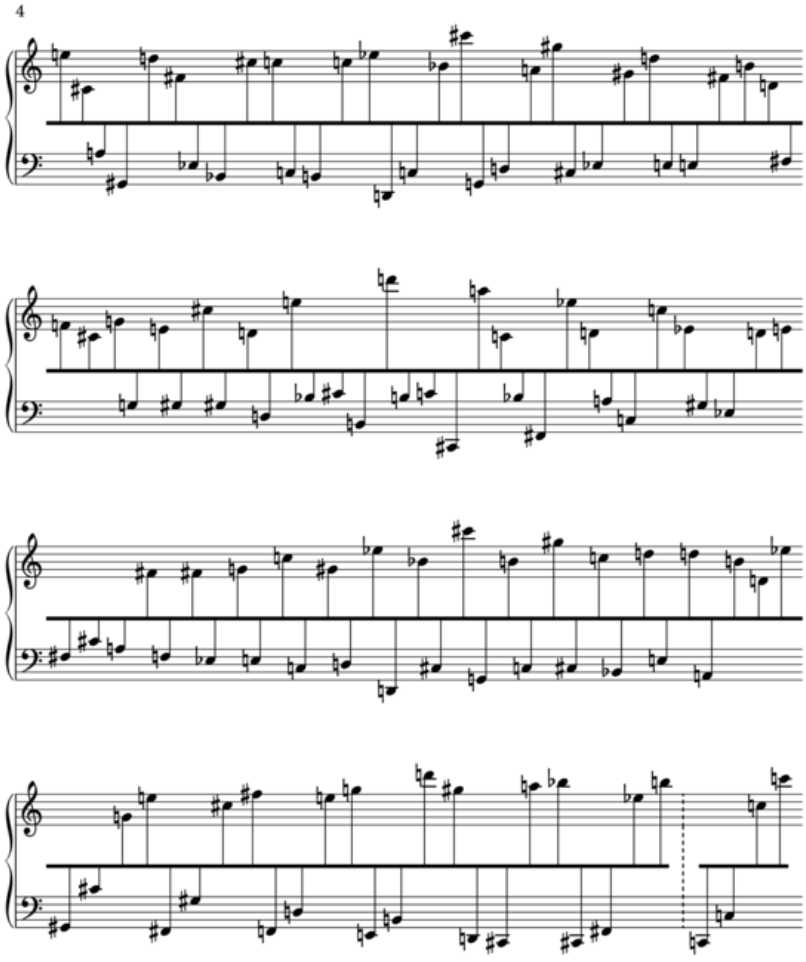
Yann Ics

Vivacetto, leggiero e comodo

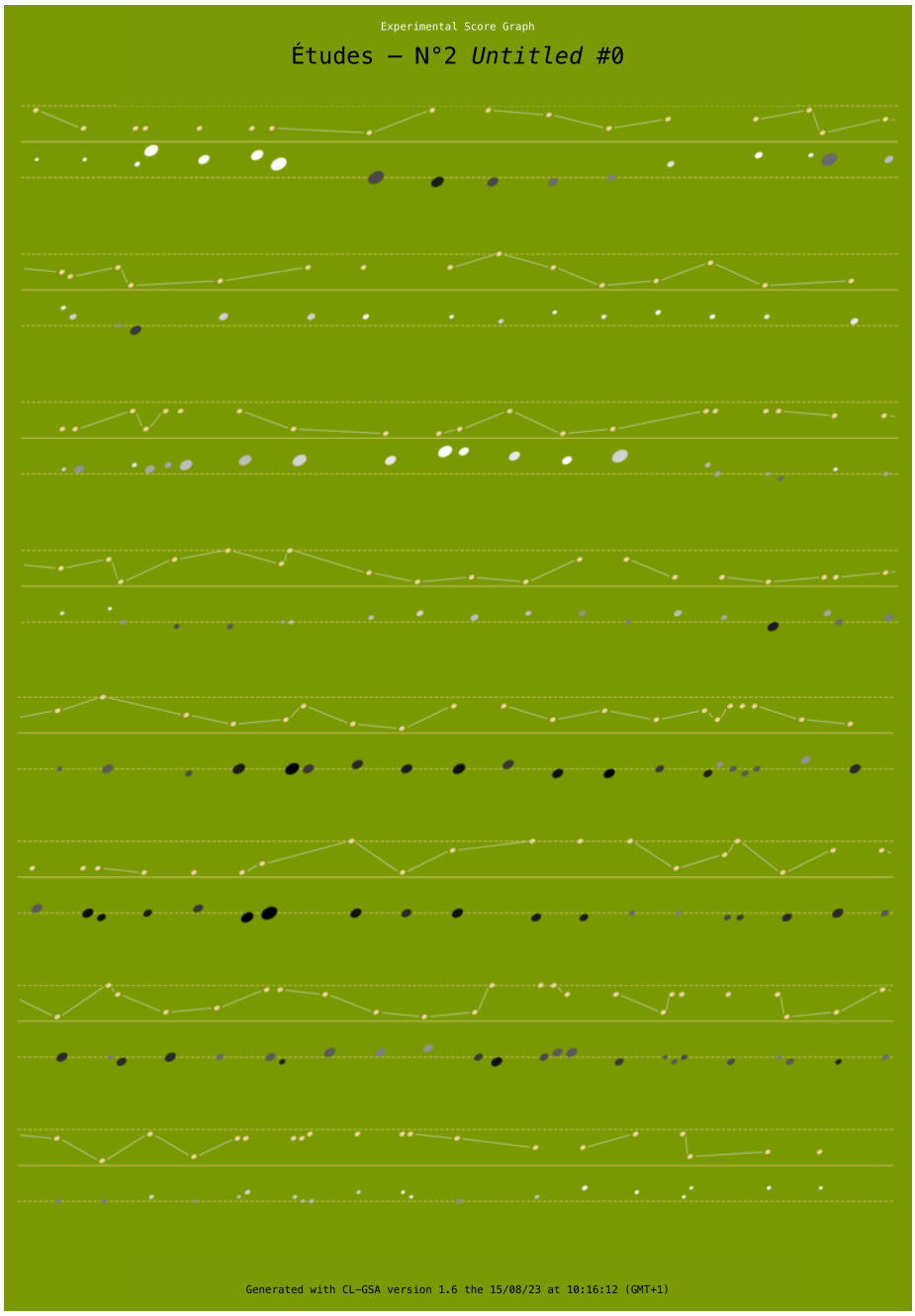



2


3



Cette étude représente un cycle formel. Les éventuelles reprises sont laissées à la discrétion de l'interprète et se font à partir de la barre de mesure en pointillé. L'absence de la double barre de fin indique simplement le caractère cyclique de cette étude et suppose une continuité hors de l'espace audible.


Experimental Score Graph
Études – N°2 *Untitled #0*
Generated with CL-GSA version 1.6 the 15/08/23 at 10:16:12 (GMT+1)

References

→ The references to go further or deeper for advanced readers are prepended by an asterisk.

Analyse Musicale : La forme: unité et multiplicité. Numéro 20, Paris, Juin 1990.

Paolo Aralla. *Morphological Analysis*. in PRISMA 01, Euresis Edizioni, Milan 2002.
http://www.paoloaralla.it/doc-pdf/P.ARALLA-MorphologicalAnalysis.pdf

John Beaulieu. *Music and Sound in the Healing Arts*. Barrytown/Station Hill Press, 1987.

Jean-Yves Bosseur, *Du son au signe. Histoire de la notation musicale*, Éditions Alternatives, 2005.

Sound and Vibration Handbook | Brüel & Kjær, *Acoustics – Sound Attenuation in Air*, Web page accessed 18 May 2021.
https://www.bksv.com/downloads/svpockethandbook/

Alain de Cheveigné. *Formant Bandwidth Affects the Identification of Competing Vowels.*. Web publication, 1999.
http://recherche.ircam.fr/equipes/pcm/cheveign/ps/icphs99.pdf

Marc Deléglise. *Permutations et cycles*. Université Lyon 1, Capes Externe Math, 2010–2011.
http://math.univ-lyon1.fr/capes/IMG/pdf/cycles.pdf

Jacques Derrida. *De la grammatologie*. Les Éditions de Minuit, Paris 1967.
https://www.pileface.com/sollers/IMG/pdf/de_la_grammatologie.pdf

Peter Desain, Henkjan Honing. *The quantization problem: traditional and connectionist approaches*. In *Understanding Music with AI: Perspectives on Music Cognition*, Edited by Mira Balaban, Kemel Ebcioglu and Otto Laske, AAAI Press, 1992, pp. 448 to 463.

Michel Foucault. *Les Mots et les Choses*. Gallimard, 1966.

James Gleick. *La Théorie du Chaos – Vers une nouvelle science*. Traduit de l'anglais par Christian Feanmougin. Flammarion, 1991.

* David Goldberg. *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. Numerical Computation Guide, July 2001, pp. 171 to 264.
https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/02Numerics/Double/paper.pdf

* Tomihisa Kamada, Satoru Kawai. *An Algorithm for Drawing General Undirected Graphs*. Information Processing Letters 31:1, April 1989.
https://pdfs.semanticscholar.org/b8d3/bca50ccc573c5cb99f7d201e8acce6618f04.pdf

Pierre Hanna, Pascal Ferraro, Matthias Robine, Julien Allali. *Recherche de documents musicaux par similarité mélodique*. Document numérique, Volume 11, 2008, pp. 107 to 125.
https://www.cairn.info/revue-document-numerique-2008-3-page-107.htm

Marshall McLuhan. *Understanding Media – The Extensions of Man*. First edition 1964, MIT Press edition, 1994.

* John Makhoul. *Linear Prediction: A Tutorial Review*. Proceedings of the IEEE 63(4), 1975.
http://www.commsp.ee.ic.ac.uk/%7Exl404/papers/Linear prediction A tutorial review.pdf

Stelios Manousakis. *Musical L-Systems*. Sonology Master's Thesis. The Royal Conservatory, The Hague. June 2006.
http://carlosreynoso.com.ar/archivos/manousakis.pdf

Claudie Marcel-Dubois, abbé François Falc'hun. *Les archives de la Mission de folklore musical en Basse-Bretagne de 1939*. Web site retrieved in 2016.
http://bassebretagne-mnatp1939.com/pages/presentation.html

Filippo Tommaso Marinetti. *La nouvelle religion – morale de la vitesse*. Nouveau Manifeste publié dans le premier numéro du journal l'*Italie futuriste*, le 11 mars 1916. Traduction d'Olivier Lahbib.
.../La_nouvelle_religion-morale_de_la_vitesse_1916_Marinetti_traduction_

Anthony Morlot. *Les drogues numériques et ondes binaurales: I-Doser, phénomène de mode ou réel danger?*. Université de Lorraine, Faculté de Pharmacie. Mémoire en vue de l'obtention du Diplôme d'État d'Audioprothésiste, 2012.
http://docnum.univ-lorraine.fr/public/BUPHA_MAUDIO_2012_MORLOT_ANTHONY.pdf

* Alexandre Papadopoulos, Pierre Roy, François Pachet. *Avoiding Plagiarism in Markov Sequence Generation*, UPMC Paris 06, 2014.
https://www.csl.sony.fr/downloads/papers/2014/papadopoulo s-14a.pdf

* Samuel Phillips, Anurag Agarwal, Peter Jordan. *The Sound Produced by a Dripping Tap is Driven by Resonant Oscillations of an Entrapped Air Bubble*. Scientific Reports 8:9515, 2018.
https://www.nature.com/articles/s41598-018-27913-0.pdf

Curtis Roads. *L'audionumérique – Musique et informatique*. Dunod, 2007. Traduction et adaptation de Jean de Reydellet.

Bob Snyder, Robert Snyder. *Music and Memory: An Introduction*. A Bradford Book Mit Press, 2001.
https://monoskop.org/images/f/f3/Snyder_Bob_Music_and_Memory_An_Introduction.pdf

Sophie Stévance. *Les Opérations Musicales Mentales De Duchamp. De La 'Musique En Creux'*. Images Re-Vues, no. 7 (2009).
https://www.academia.edu/5607095

Peter Szendy. *Brian Ferneyhough*. Textes réunis par l'auteur, Ircam–L'Harmattan, 1999.
.../Compositeurs-d-aujourd-hui-Peter-Szendy-Brian-Ferneyhough-L-Harmattan-1999-pdf

Frédéric Voisin, Jacopo Baboni Schilingi. *Morphologie*. IRCAM Troisième édition, novembre 1999.
http://www.fredvoisin.com/IMG/pdf/Morphologie_1999.pdf

Frédéric Voisin. *Dissemblance et espaces compositionnels*. Actes des Journées d'Informatique Musicale 2011, pp. 191 to 197.
http://jim.afim-asso.org/jim11/html/actes/25_41_jim2011_fredvoisin7.pdf

Iannis Xenakis. *Formalized Music*. Stuyvesant, NY: Pendragon Press, 1992.
.../Xenakis_Iannis_Formalized_Music_Thought_and_Mathematics_in_Composition.pdf

Hans Holbein's <Dance of death>
woodcut <The New-Married Lady>