Some anatomical insights into *Structure verticale*

Kai Yves Linden

ircam Forum

Ateliers du Forum IRCAM 2021

Overview

Timeline

Concepts and technical set-up

Electronic «instruments»

Formal structure

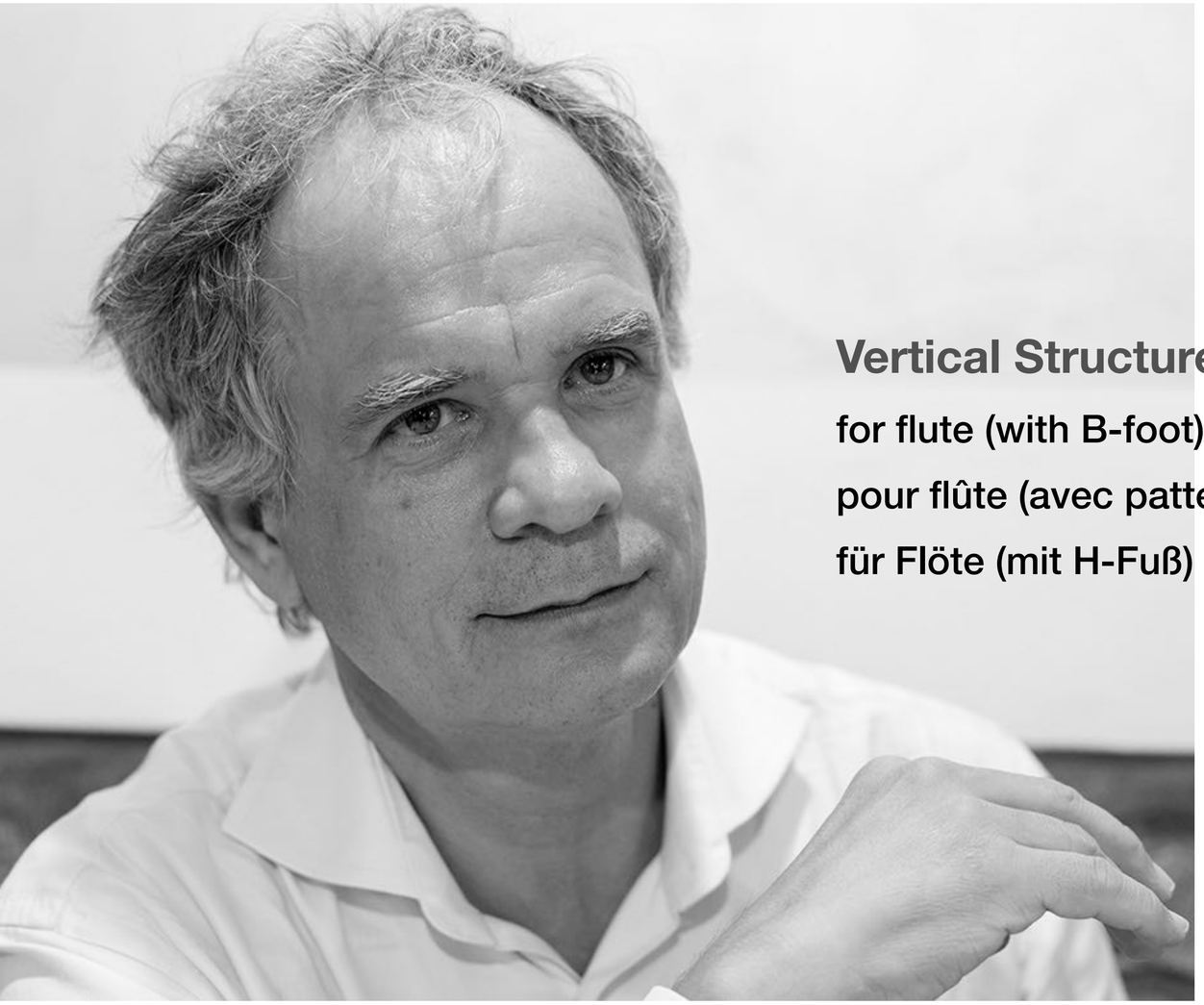
Tempo and delay

Onset detection

Practical insights

Conclusion

Kai Yves Linden



born 1960 in Düsseldorf, Germany, studied composition with Wolfgang Hufschmidt at Folkwang University of Arts in Essen, Germany, from 1982 to 1986. Currently working as software engineer in a standardization institute he pursues compositional projects in his free time. His compositions comprise chamber works for solo instruments and ensembles and vocal music.

Vertical Structure · Structure verticale · Vertikale Struktur

for flute (with B-foot) and electronic extension in real time

pour flûte (avec patte de si) et extension éléctronique en temps réel

für Flöte (mit H-Fuß) und elektronische Erweiterung in Echtzeit

2017/2021

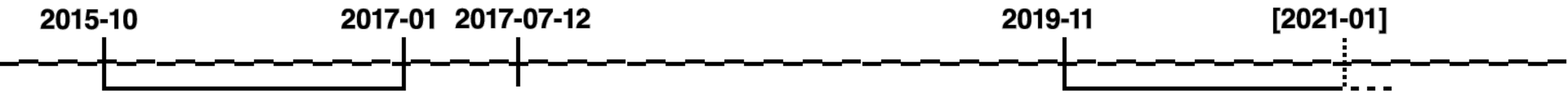
"Vertical Structure" is a mixed music for flute and electronics depending on IRCAM technologies Antescofo, Spat and MuBu/PiPo. In my presentation I will explain the conceptual principles of the piece and dissect some excerpts from its 19 minutes duration. I will also consider practical strategies in the compositional process to gain productivity and creativity when working with a complex software set-up.

Timeline

2015-10 — 2017-01 — 2017-07-12 — 2019-11 — [2021-01]

- 2015-2017 first version
  - Collaboration with flutist Anne Horstmann
  - First performance 12 July 2017 Margarethenkapelle, Dortmund
  - Notes on web page: https://kymbala.de/musik/oeuvre/verticale/
- 2019-2021 second version
  - rewriting mostly the electronic part
  - adding ca. 7 minutes (giving a total duration of ca. 19 minutes)
  - final editorial work ongoing

Timeline



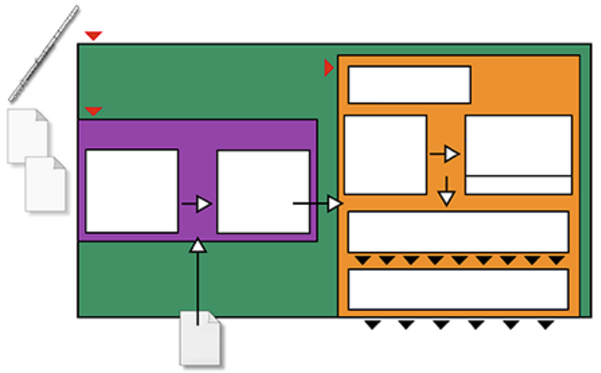
É c r i r e e t r é é c r i r e …

(write and re-write) – favored working style

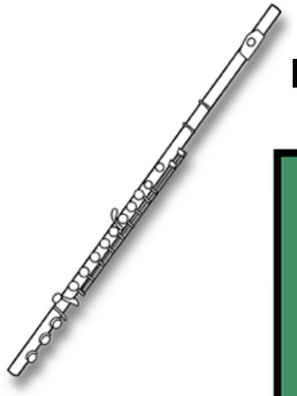
“Vertical Structure” is perhaps my most important work – at least in terms of hours I have invested into it. It was also a decisive experience in finding out what I actually want to do in music. Although the revision added more than a third of the final duration, I still consider it the same work, now in its definite form. (Yes, one has to finish and leave things sometime…)

Concepts and technical set-up

- The notes played by the **flute** are followed by the **score follower**.
- **Messages** from the score are sent to **receivers** in the sound processing environment.
- The entire performance is continuously **recorded** so parts of it can be **replayed** with arbitrary **delay**.
- Live and delayed signals are **routed** to and between **modulators** and **generators**.
- The gain (volume) of **internal signal lines** is modulated and pre-balanced.
- These are passed to the **spatialization** and **reverberation** module for the final mix.



Technical set-up



Microphone (attached to instrument)

Performer's part

Result score

Signal processing environment

Sequencing

IRCAM Antescofo

Score following

Listening machine

Action management

Scheduling machine

Max

Structure verticale

Sound processing

Recording unit

buf-rec

Signal path control

inroute

Modulators

VZ (delay/playback)

TP (transposers)

RM (ring modulators)

SIN (sine generator)

Internal output gain control

inmix

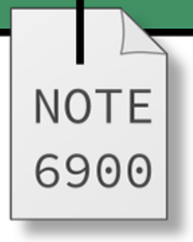
Spatial distribution

IRCAM Spat

spat

1 2 3 4 5 6

(4, 6 or 8) Loudspeakers



Action score
(machine-readable)

Electronic «instruments»

- Bounded number of signal processing objects by disposition
  - based on fundamental electronic modulation methods
- «Personalities» – on the sound stage and *“under the surface”*
- Multiform «mirror» of the flute

«Instrument» types

Core

(Modulators)

VZ: buffer playbacks

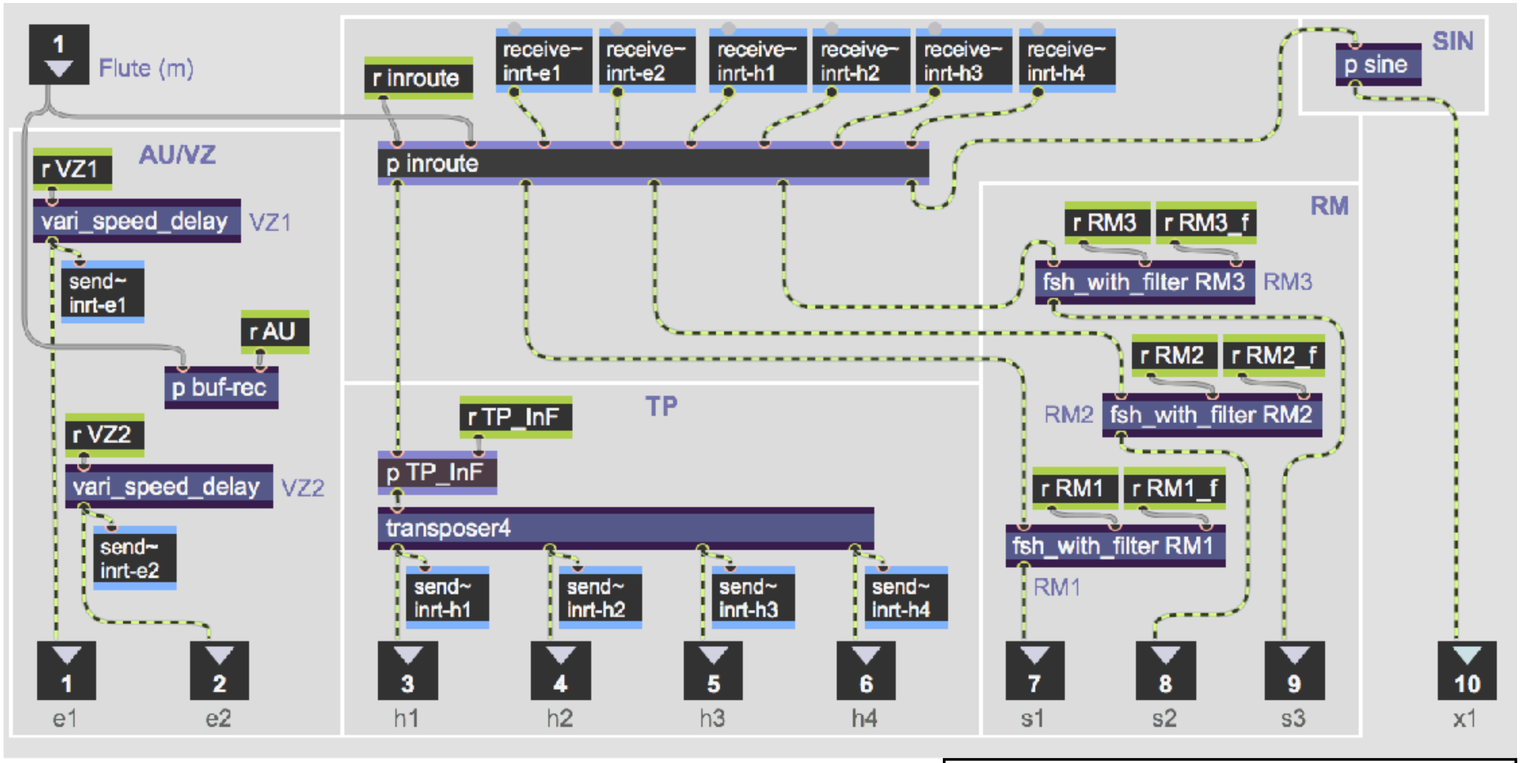
TP: transposers

RM: ring modulators

Sideline

(Generators)

SIN: sine wave oscillator



Max patcher ***soundprocessing***

Electronic «instruments»

- (1.) **Polarity** between
- the instrumentalist (the human actor) and
- the electronic sounds (the machine)

- (2.) **Differentiation** within
- the electronic sounds («instruments») and
- in their relation to the sounds produced by the instrumentalist

«Instrument» types

Core

(Modulators)

VZ: buffer playbacks

TP: transposers

RM: ring modulators

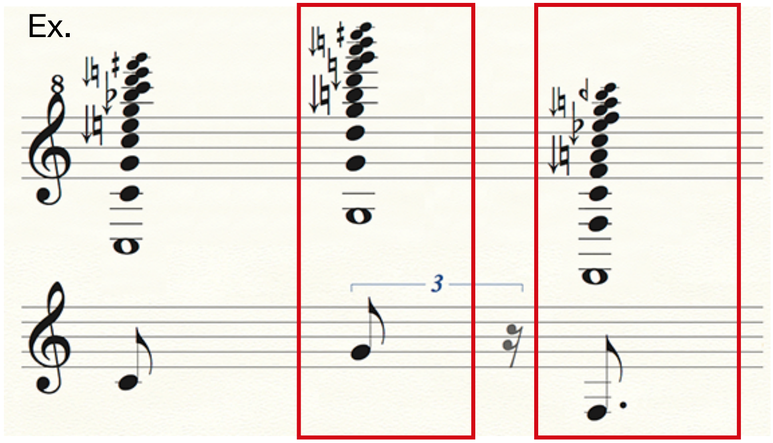
Sideline

(Generators)

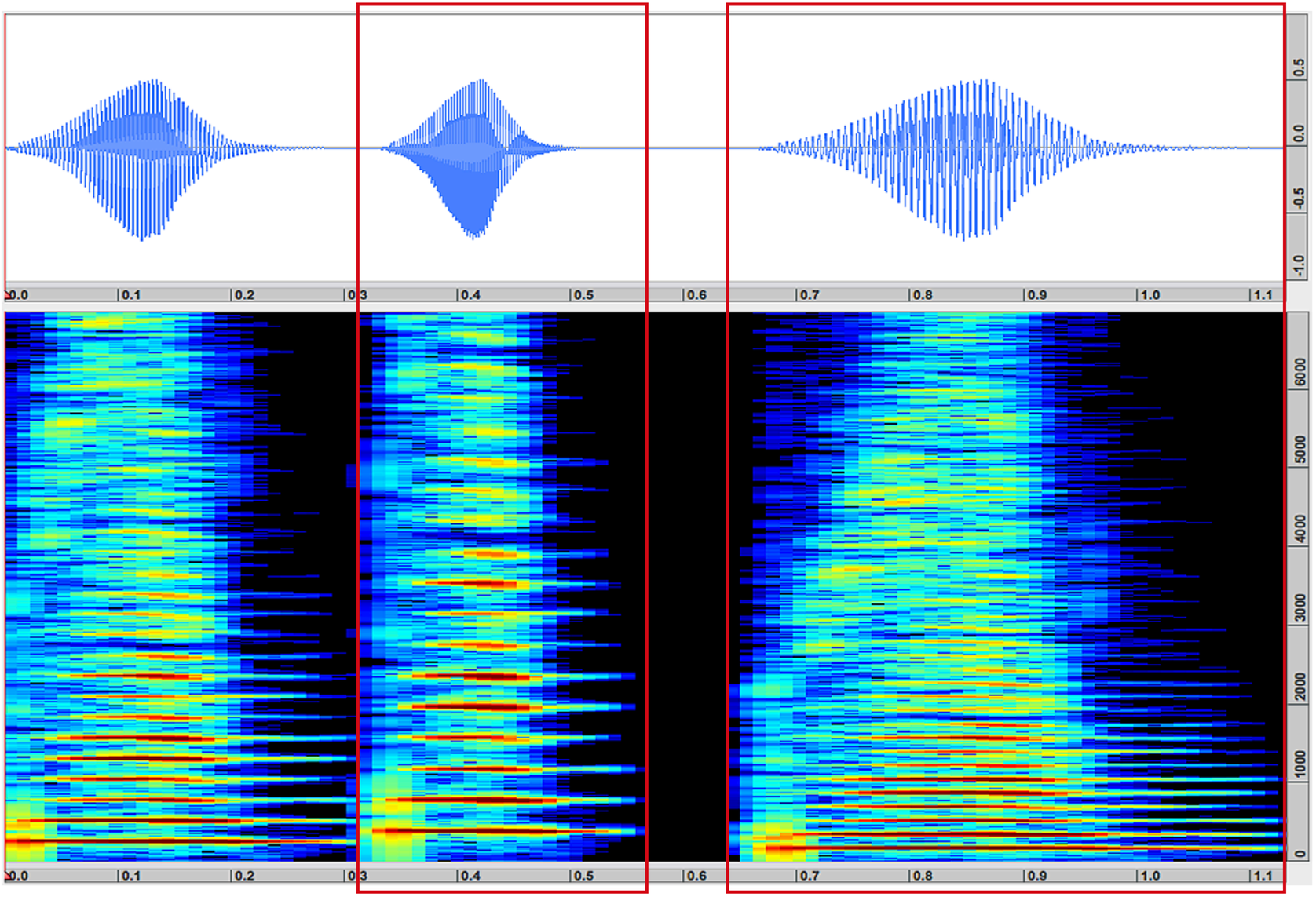
SIN: sine wave oscillator

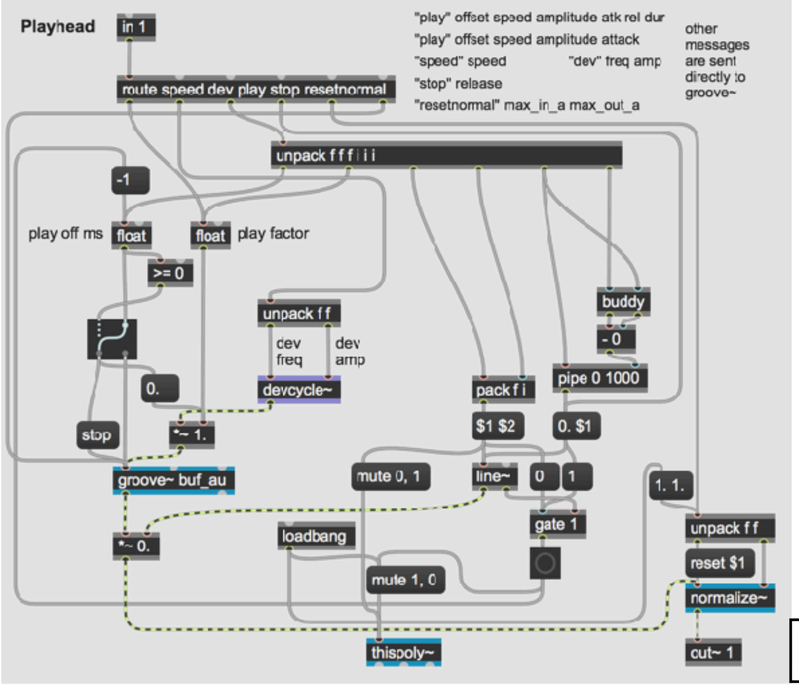
The three main types of electronic «instruments» modulate the flute sounds by changing the pitch or frequency, each in a different way.

Three types of pitch/frequency change: **1. Playback speed** – *Changing pitch and duration*



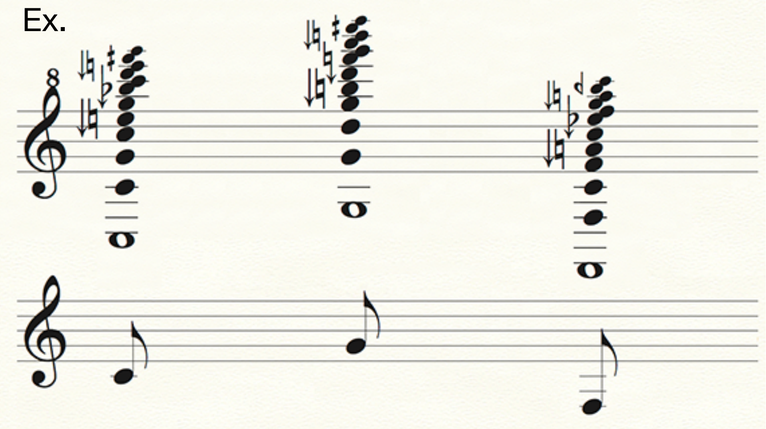
[original] f × 3/2 f × 2/3



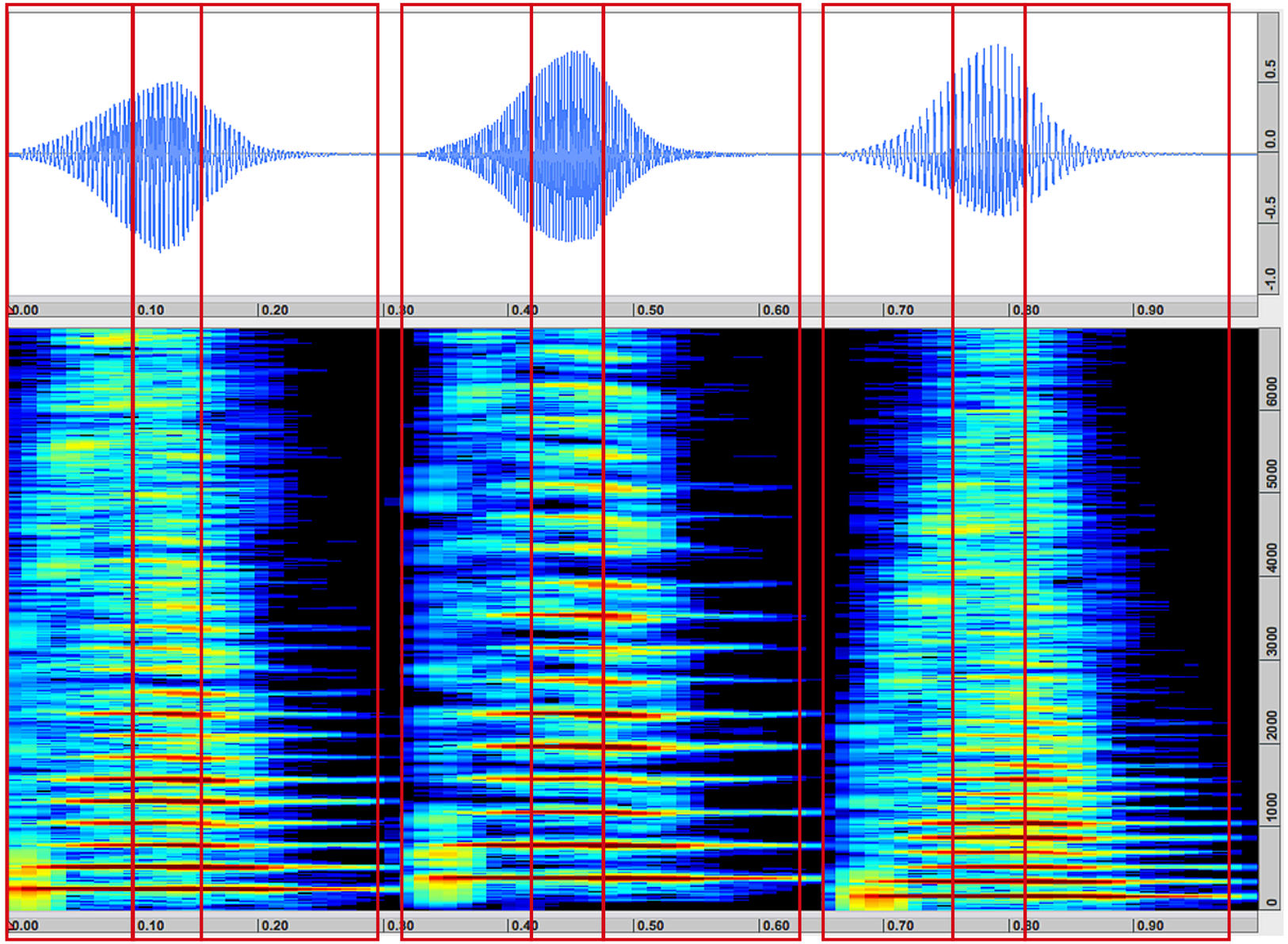


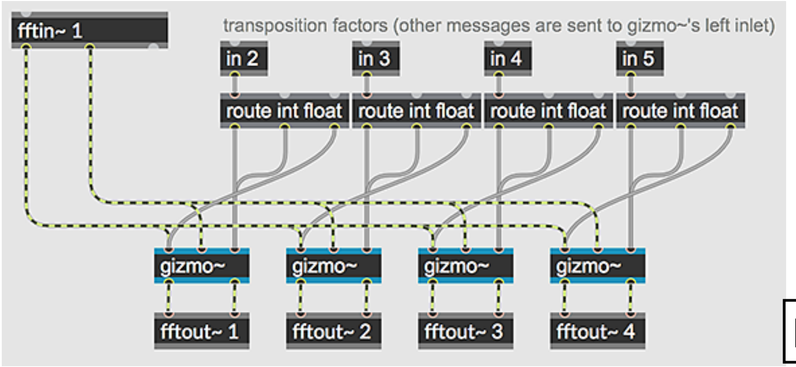
Max patcher ***polyplayhead*** in sound processing unit **AU/VZ**

Three types of pitch/frequency change: **2. Pitch shift in the frequency domain** – ***Transposing the spectrum***



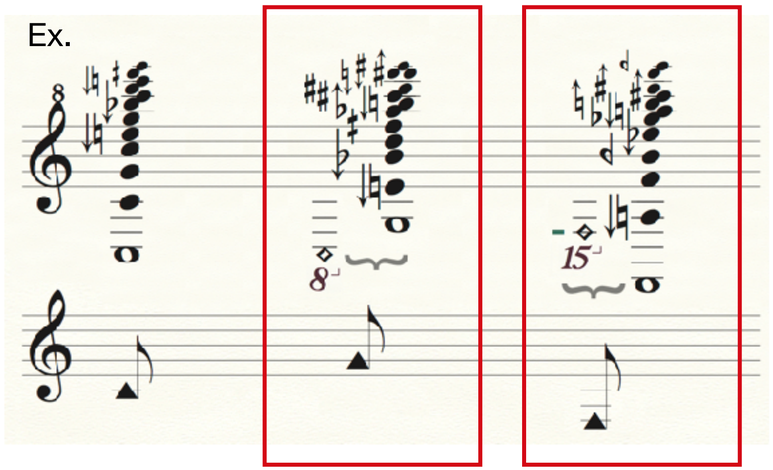
f × 1 f × 3/2 f × 2/3



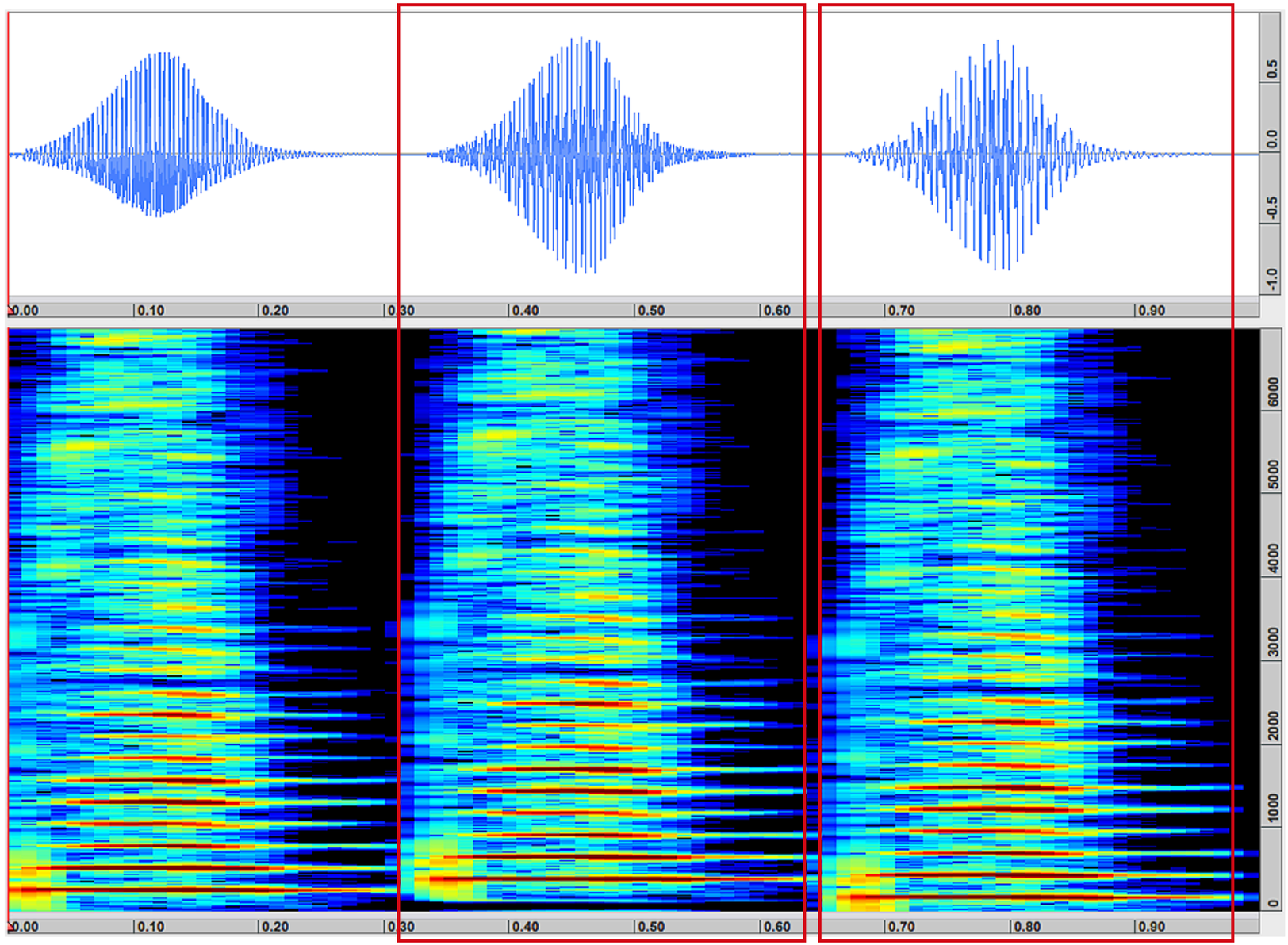


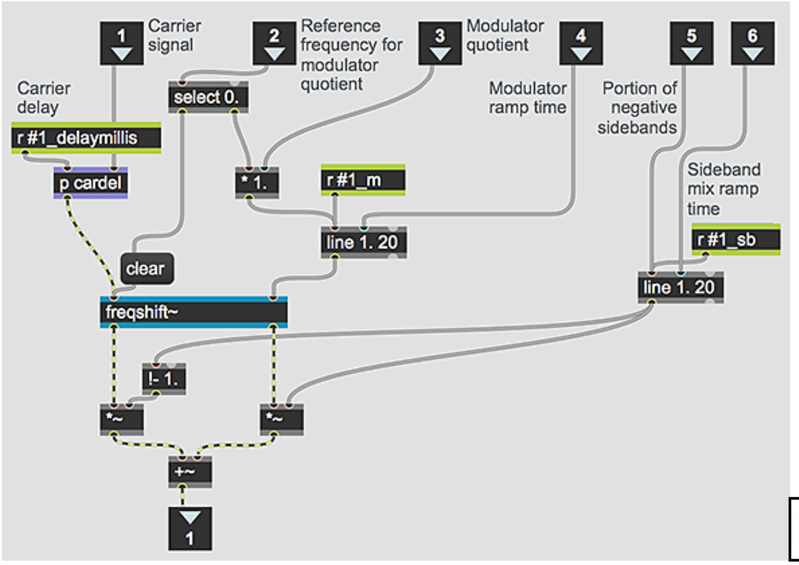
Max patcher ***pfft_tp4*** in sound processing unit **TP**

Three types of pitch/frequency change: 3. Frequency shift (ring modulation) – *Distorting the spectrum*



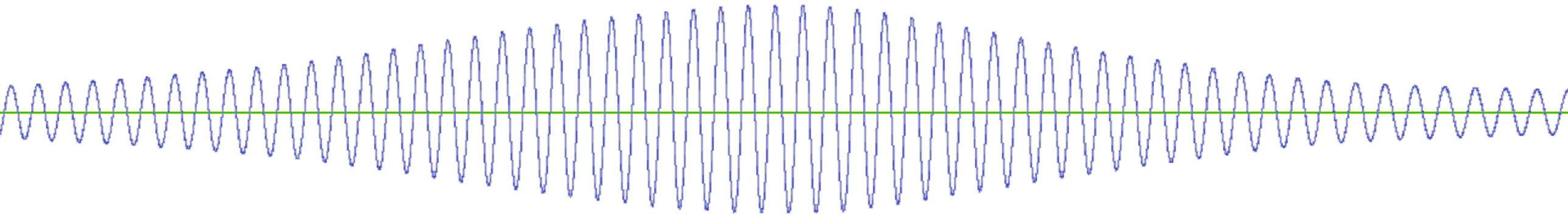
f + 0 f + f × 1/2 f + f × -1/3

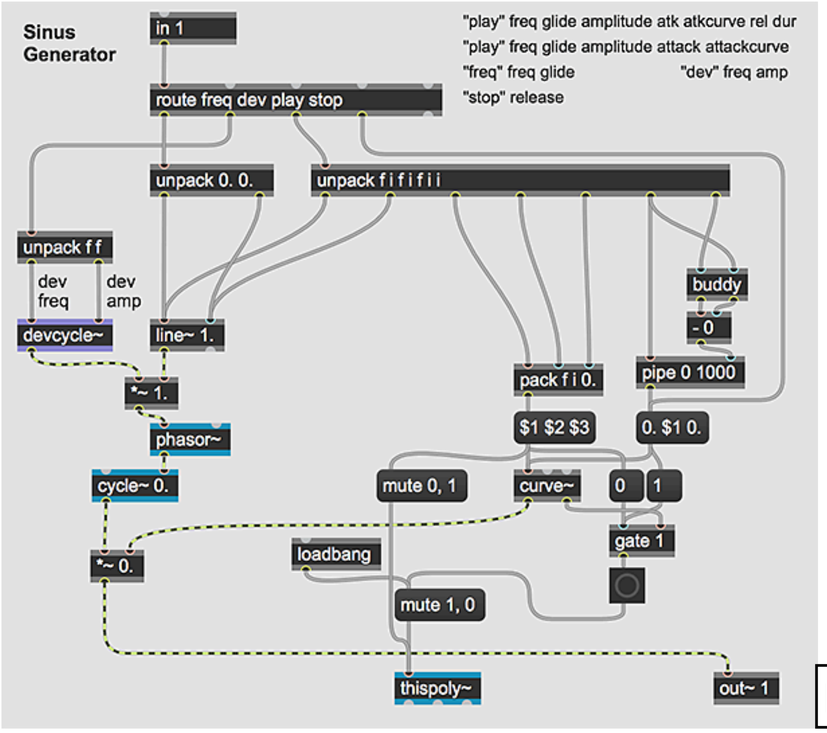




Max patcher *fsh* in sound processing unit **RM**

Additional unit: Sine wave oscillator





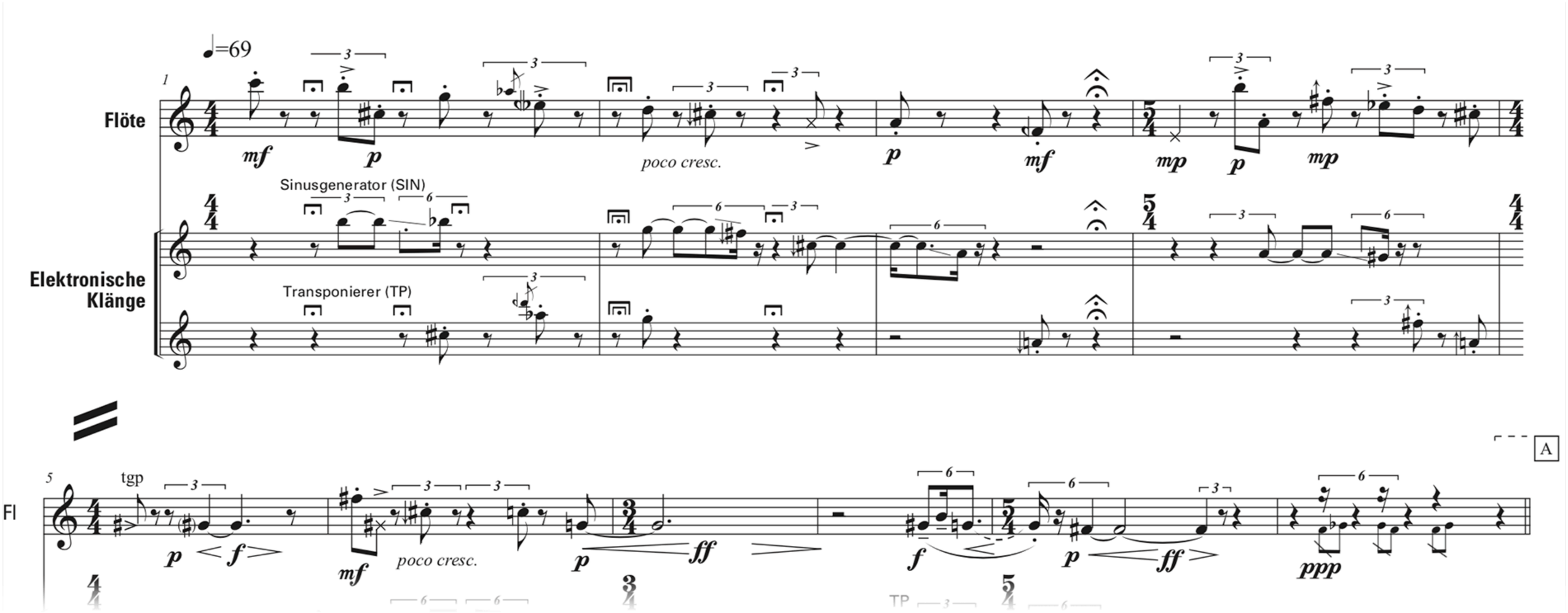
- *"Outside"* the disposition
  - however, already present as modulator in RM
- Roles:
  - "sustaining pedal" ("sympathetic vibrations")
  - "[pre-]echo"
  - "screen" (envelope co-modulated by flute or other sources)

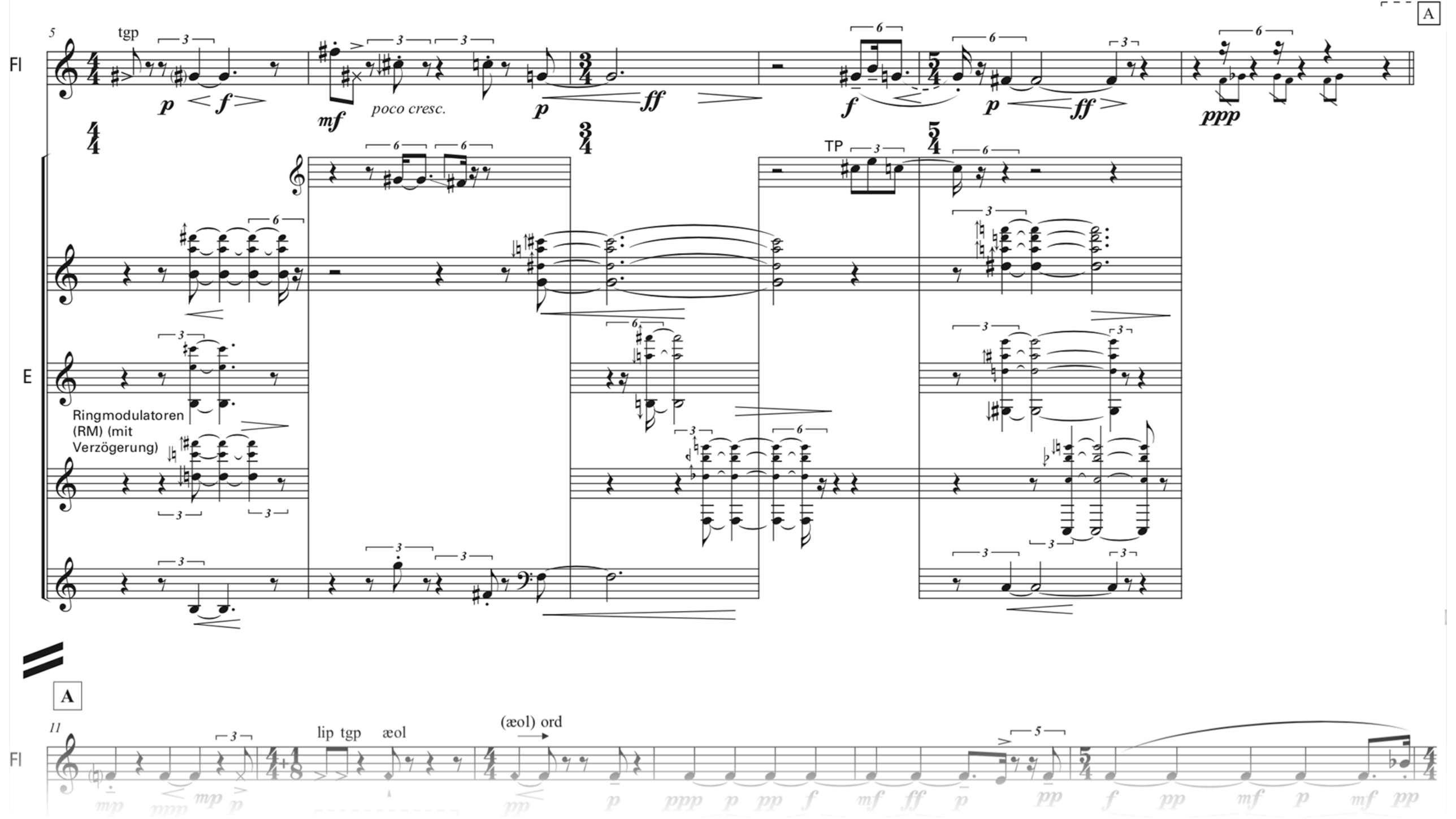
Max patcher *polysin* in sound processing unit **SIN**

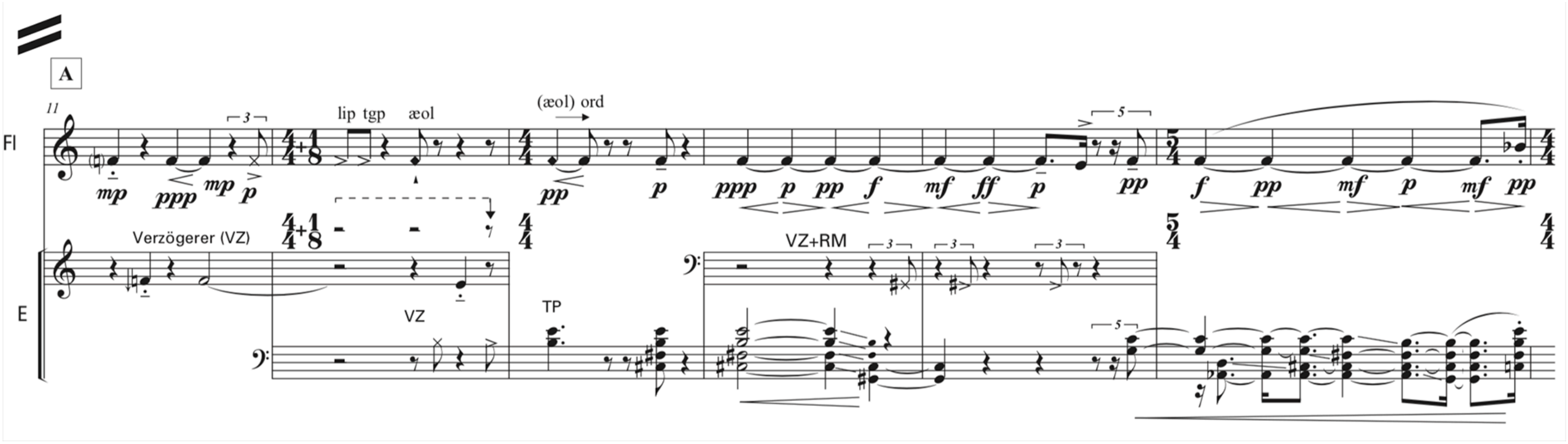
The first minute

All «instruments» are introduced, by and by, in the opening sixty and so seconds.

Excerpt m1-16




A
5
tgp
Fl
p
f
mf
poco cresc.
p
ff
f
p
ff
ppp
TP
E
Ringmodulatoren (RM) (mit Verzögerung)
A
11
lip tgp
æol
(æol) ord
Fl


A
11
Fl
E
lip tgp
æol
(æol) ord
Verzögerer (VZ)
VZ
TP
VZ+RM
mp
ppp
mp
p
pp
p
ppp
p
pp
f
mf
ff
p
pp
f
pp
mf
p
mf
pp

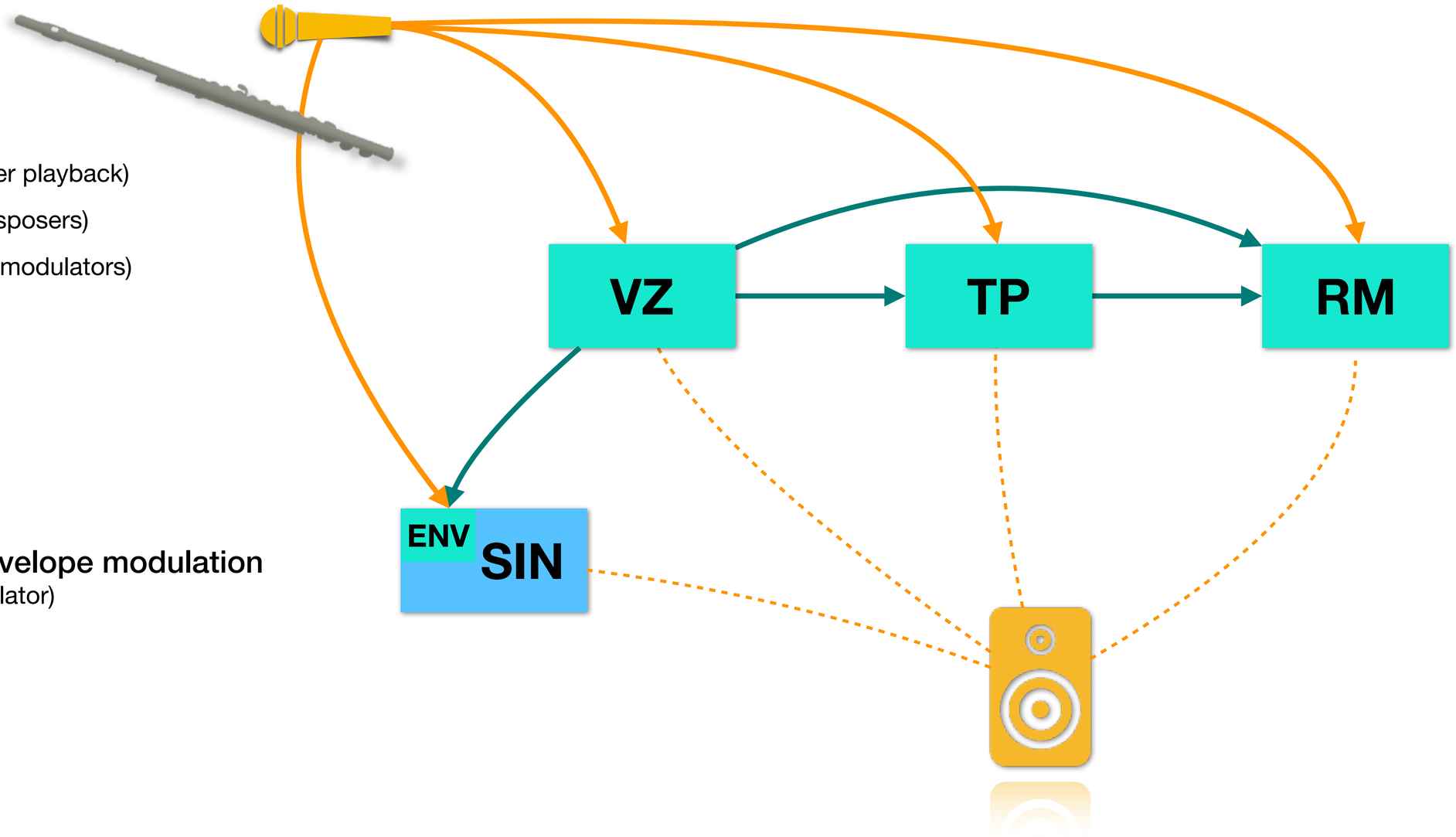
Input routing: Overview of signal feed lines

microphone → **VZ** (buffer playback)
microphone → **TP** (transposers)
microphone → **RM** (ring modulators)

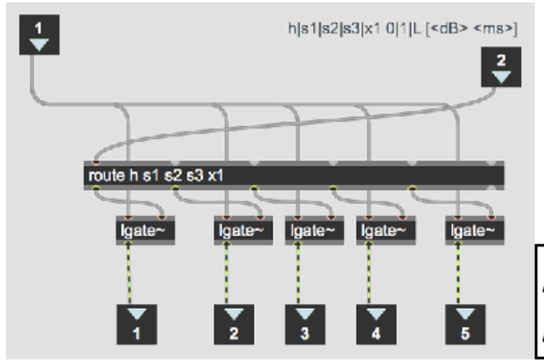
VZ → **TP**
VZ → **RM**

TP → **RM**

microphone → **SIN** | envelope modulation
VZ → **SIN** | envelope modulation
(sine oscillator)

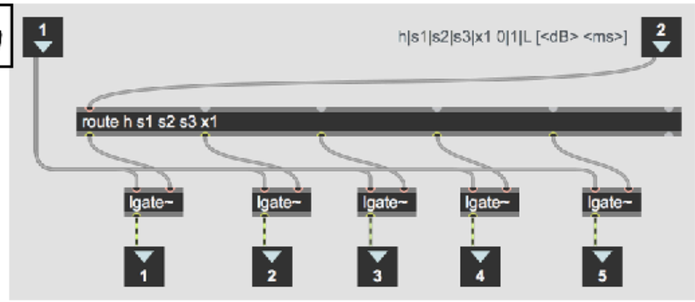


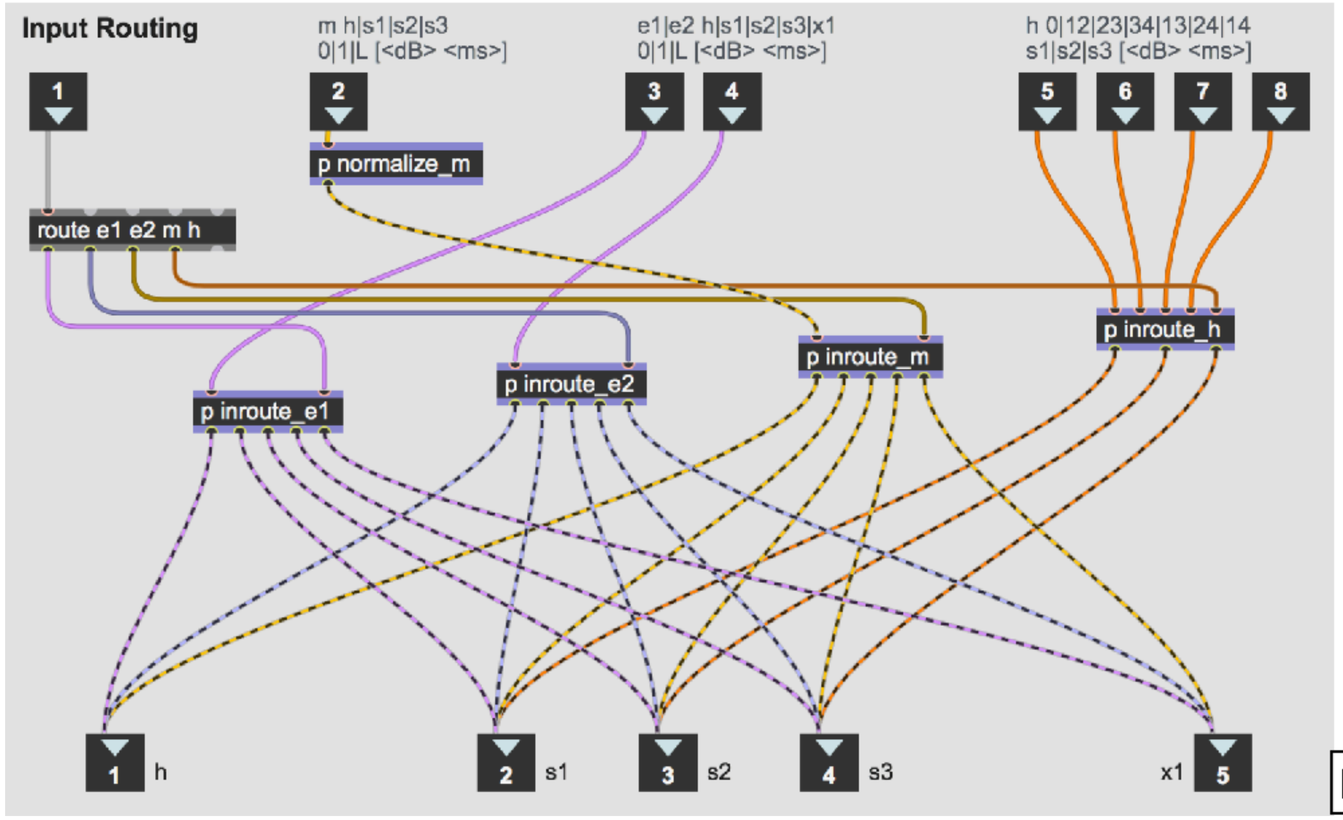
Input routing: Signal feed lines (Max patchers)



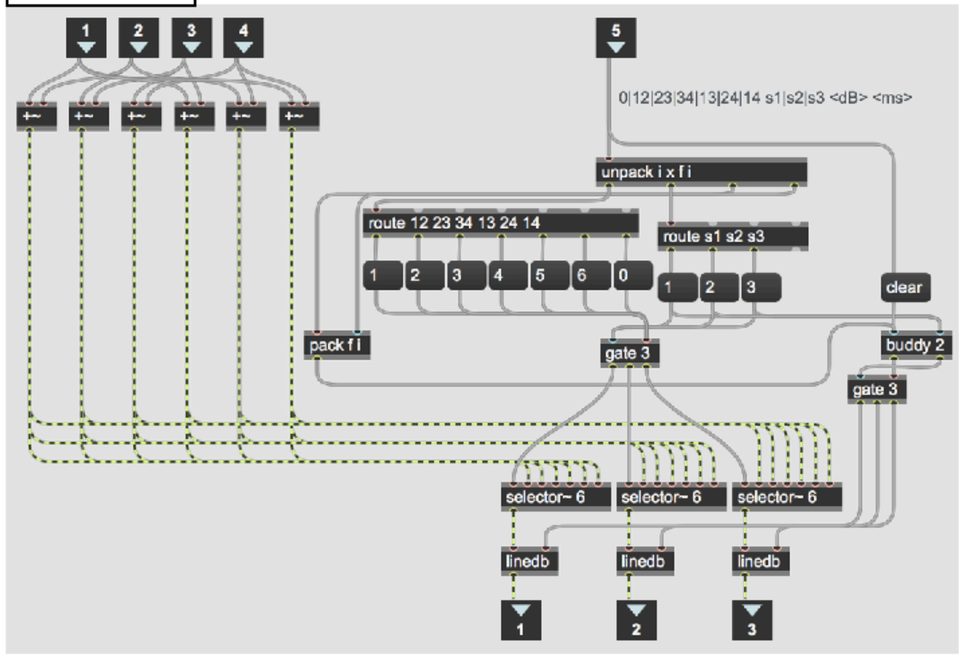
inroute_e1
inroute_e2

inroute_m



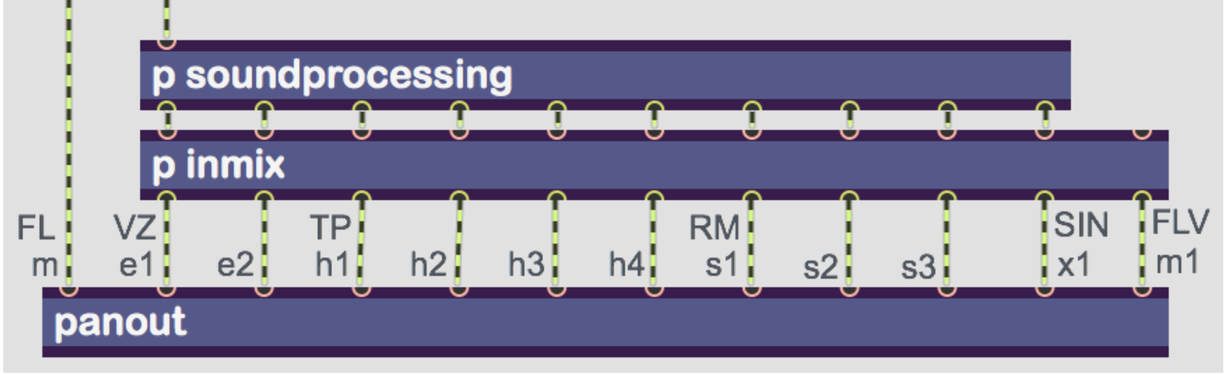


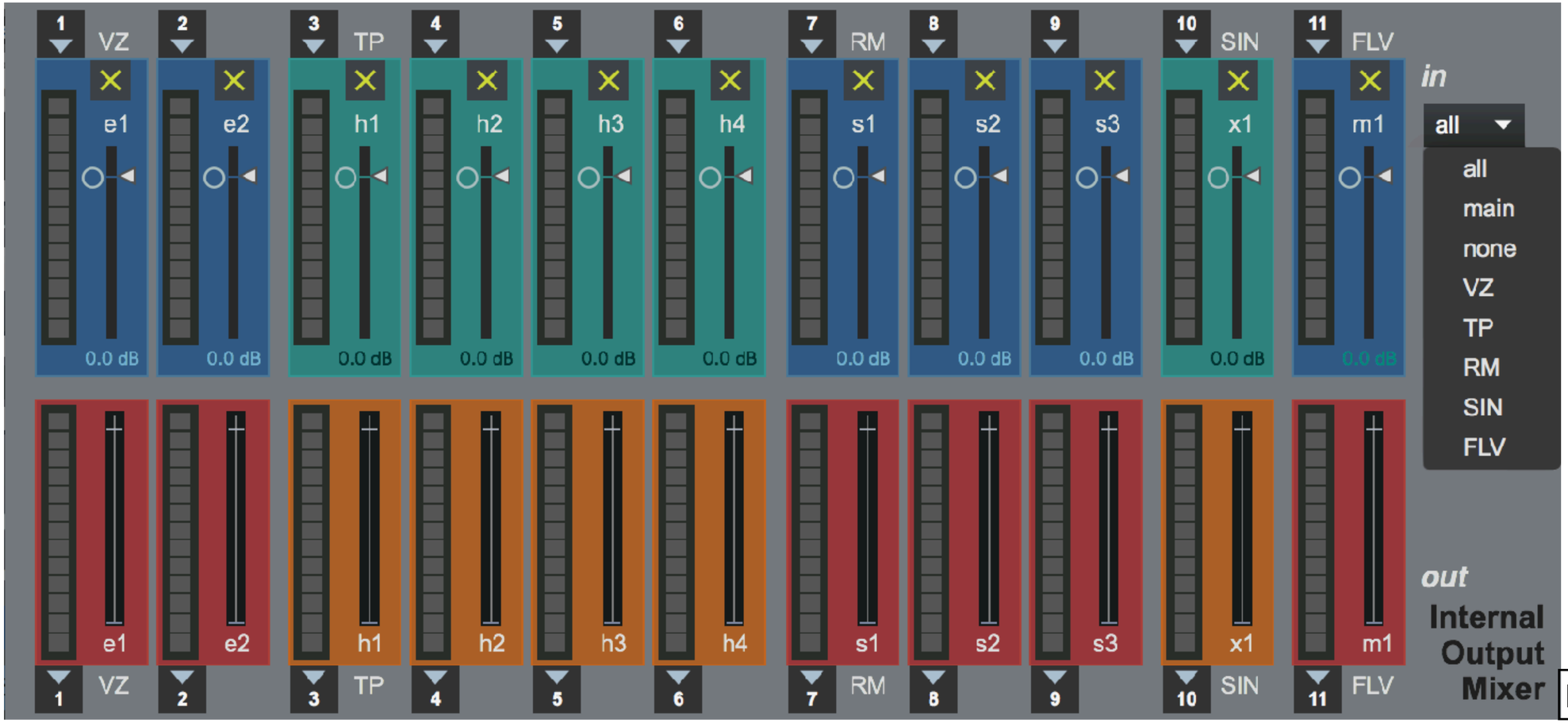
inroute_h



Max patcher *inroute*

Internal output mixer





incoming levels

faders for manual adjustment

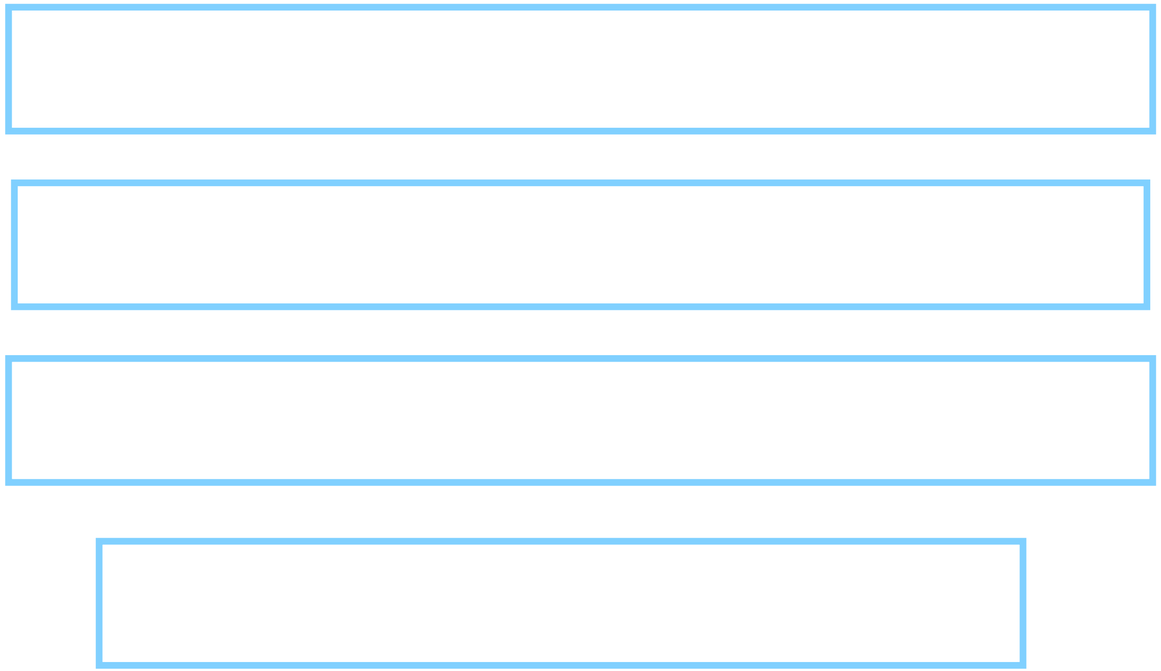
gains controlled by action score

outgoing levels

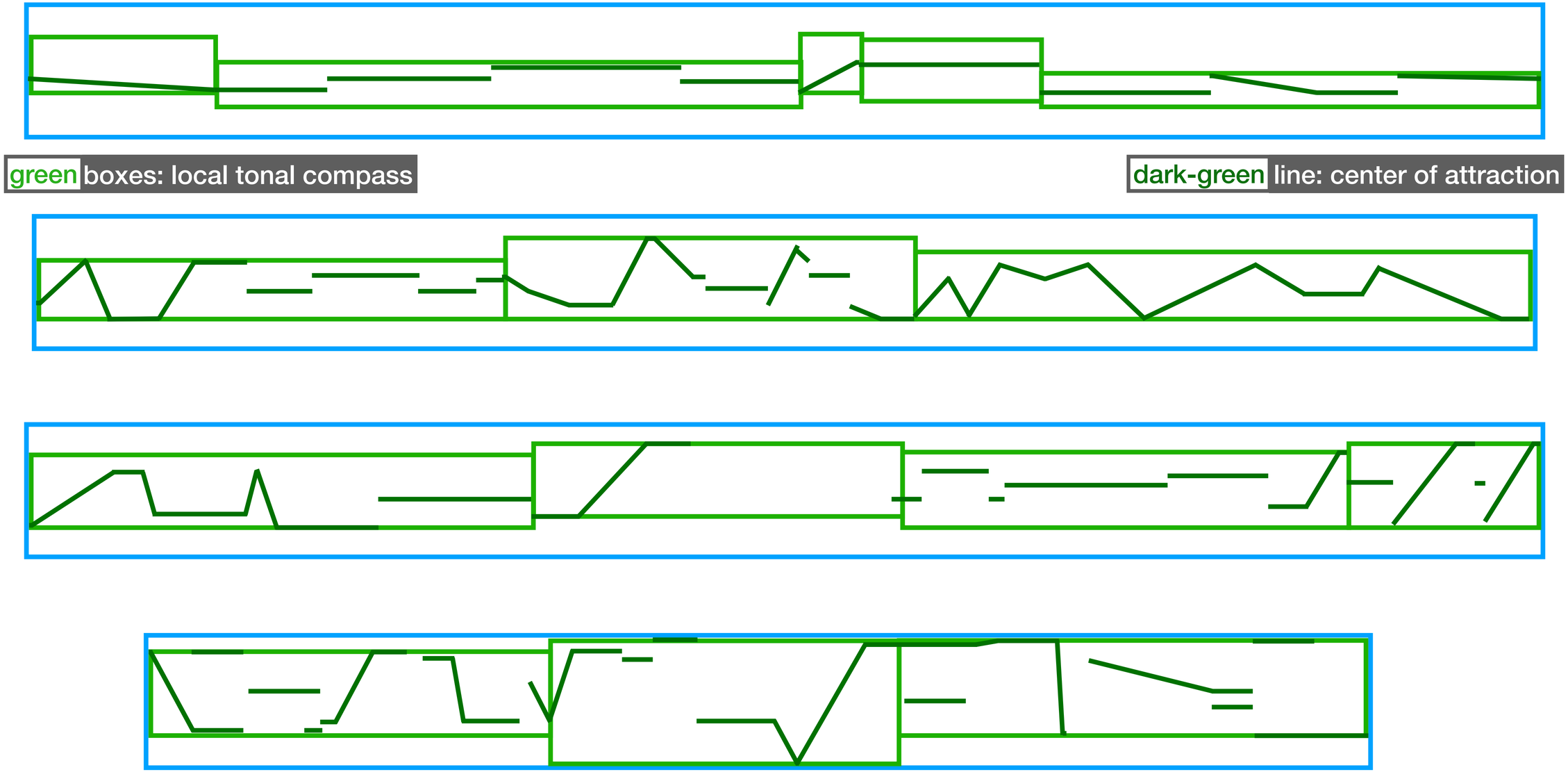
Max patcher *inmix*

Formal structure and tonal space

- “Verticale Structure” is not built on a blueprint.
- The composition was the adventure of an expedition without map.
- I’m fascinated by the ideas of temporality and contingency.
- The diagram shown here is the result of an retrospective breakdown.



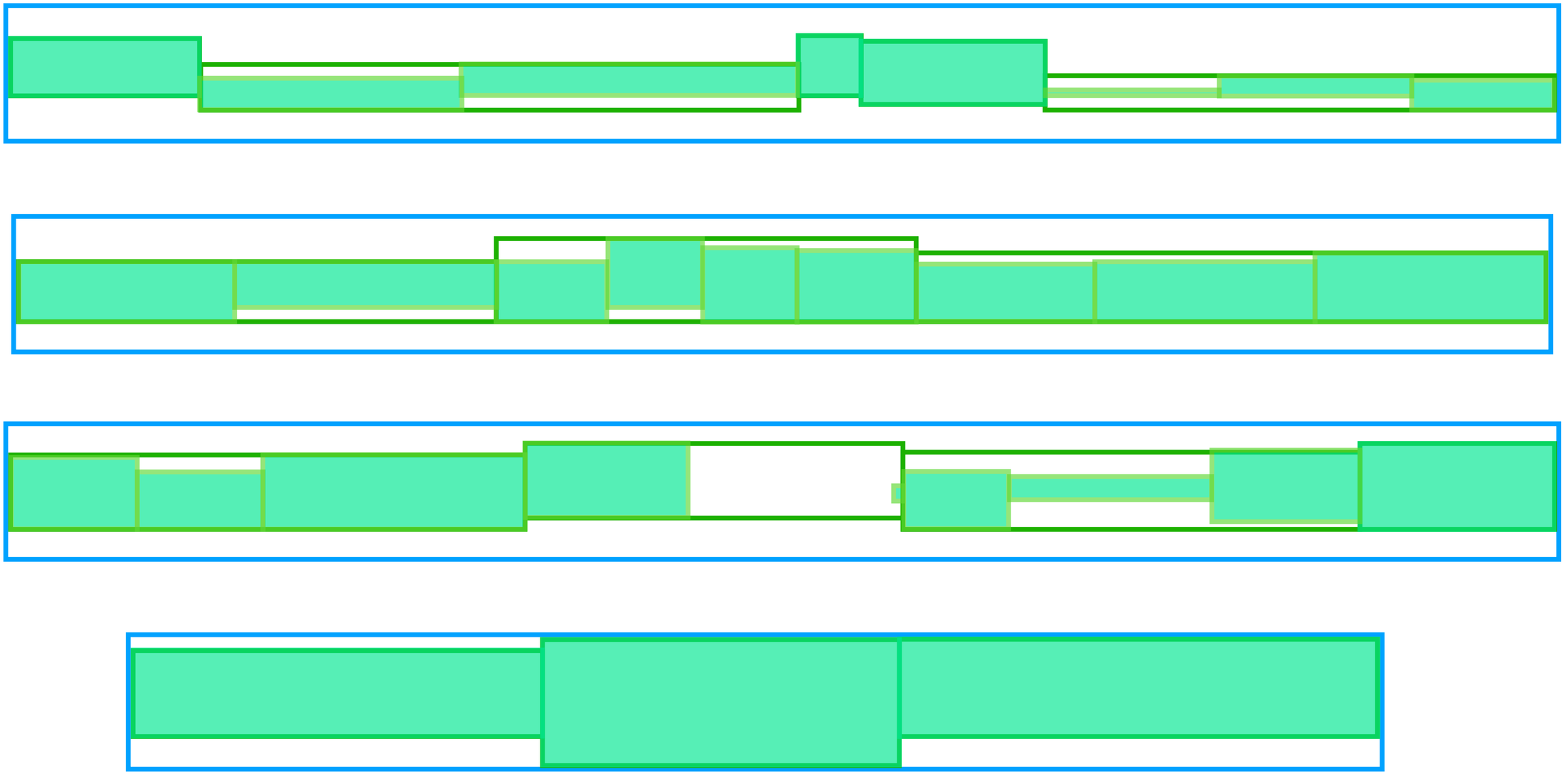
Formal structure and tonal space: **Movement in the tonal space (flute part)**



green boxes: local tonal compass

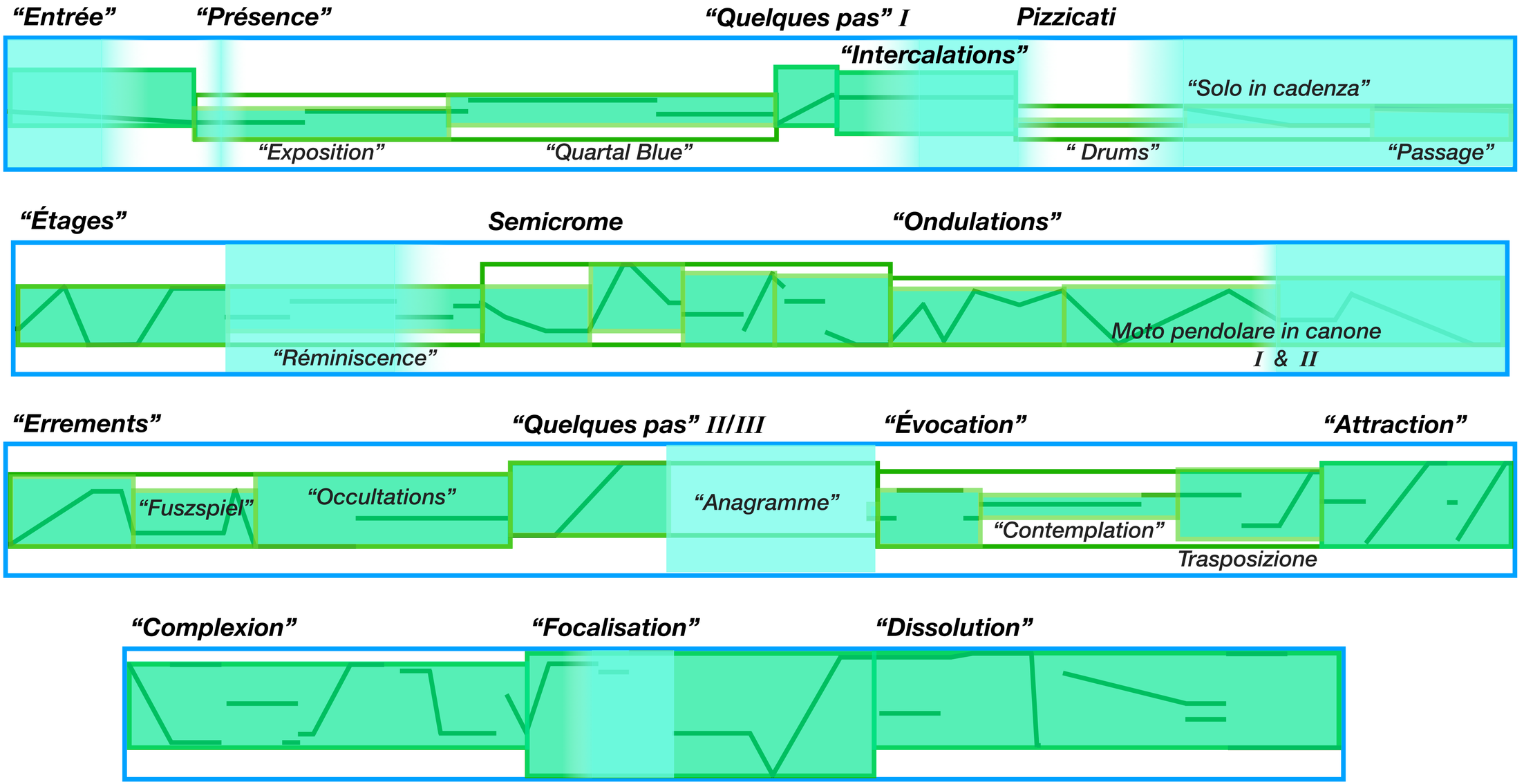
dark-green line: center of attraction

Formal structure and tonal space: **Structure formants* (defining episodes and variants)**



* property-sets regarding pitches, durational patterns, texture, generative tendencies, stylistic models, …

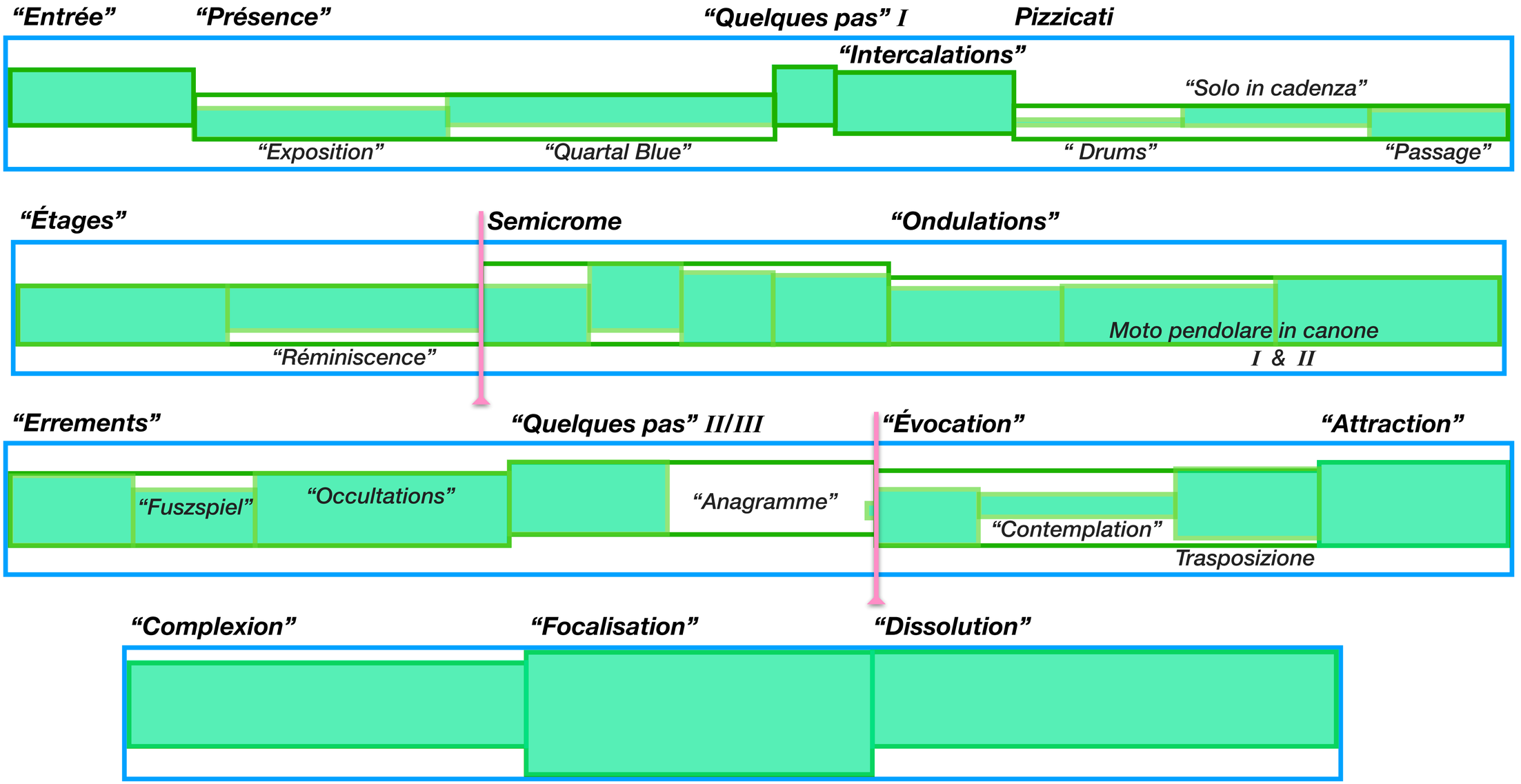
Formal structure and tonal space: **Sections and sub-sections**



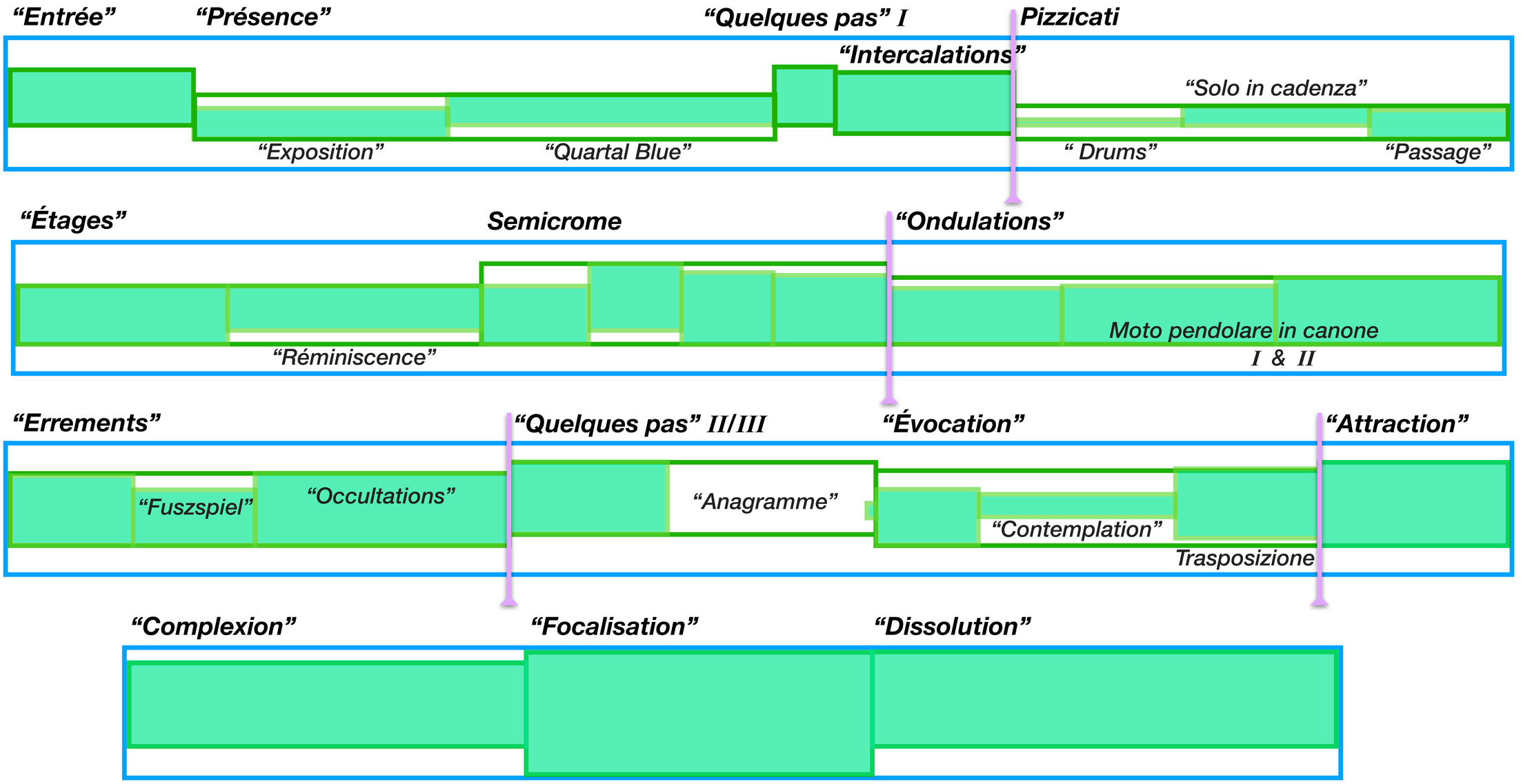
(The section names are working titles for easier location.)

indigo overlay: sections added or extended in the revision

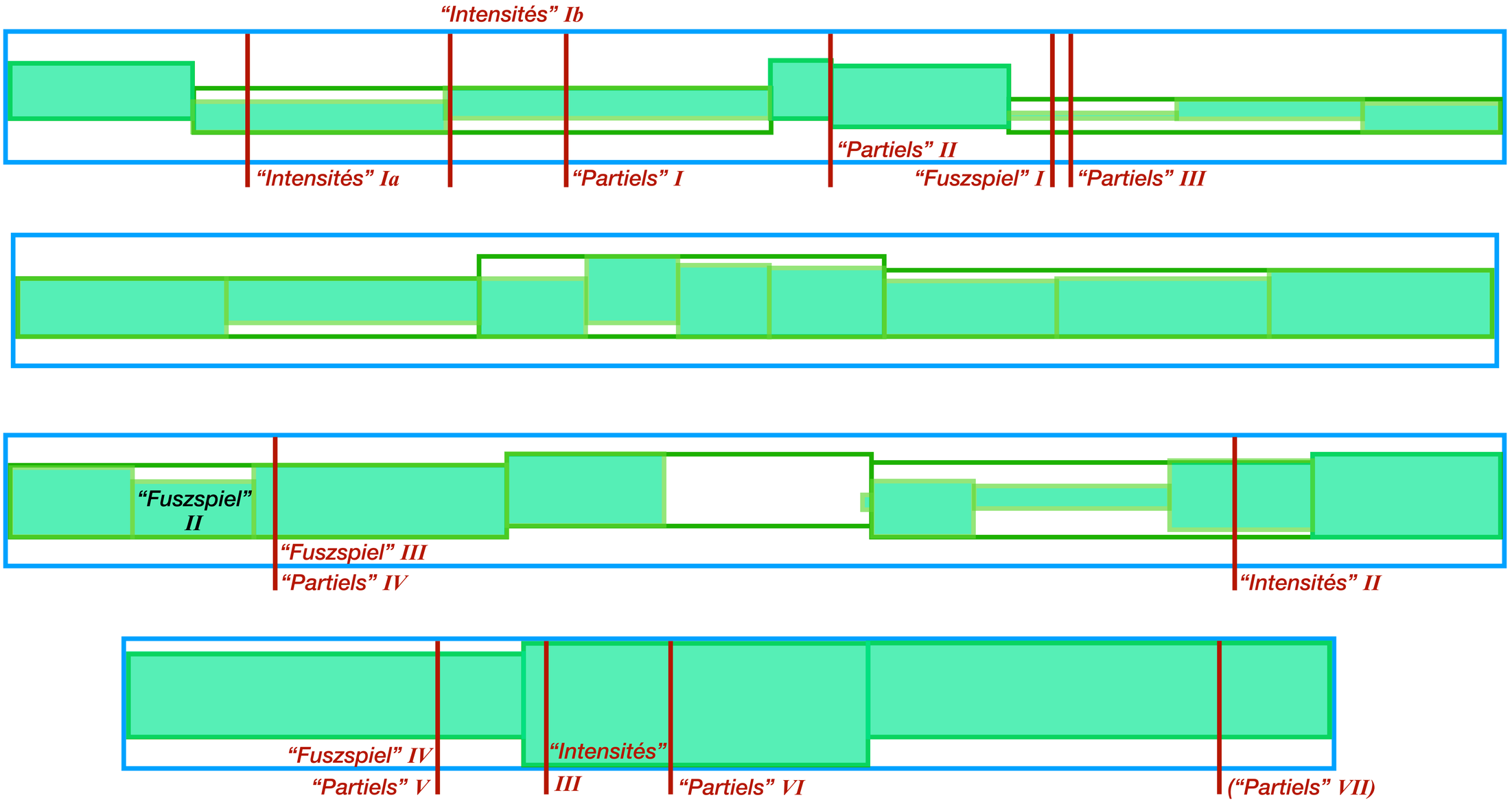
Formal structure and tonal space: Grouping into three divisions pink markers



Formal structure and tonal space: **Alternative grouping into five divisions** violet markers

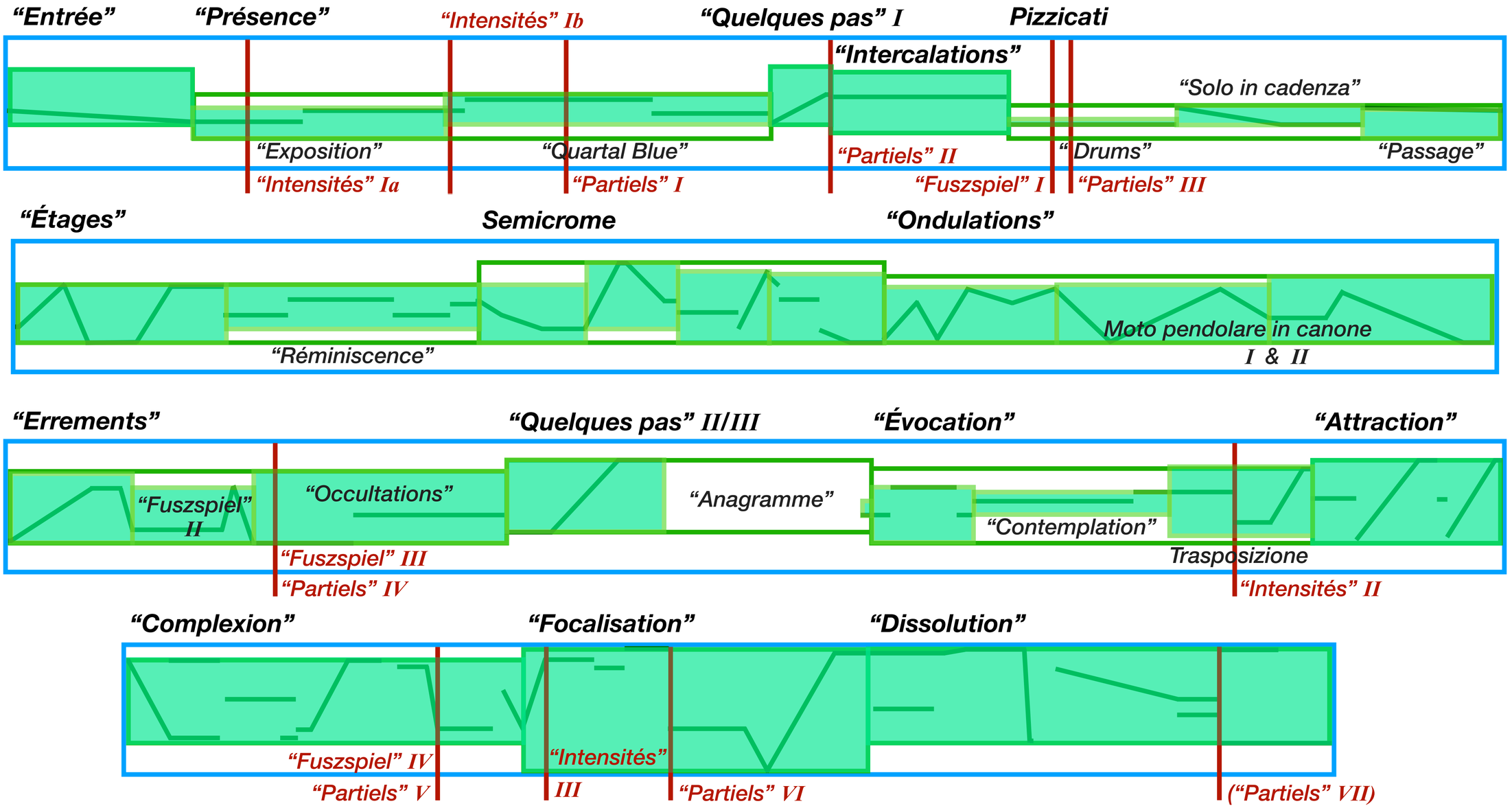


Formal structure and tonal space: **Recurring phenomenal types which add a further layer**

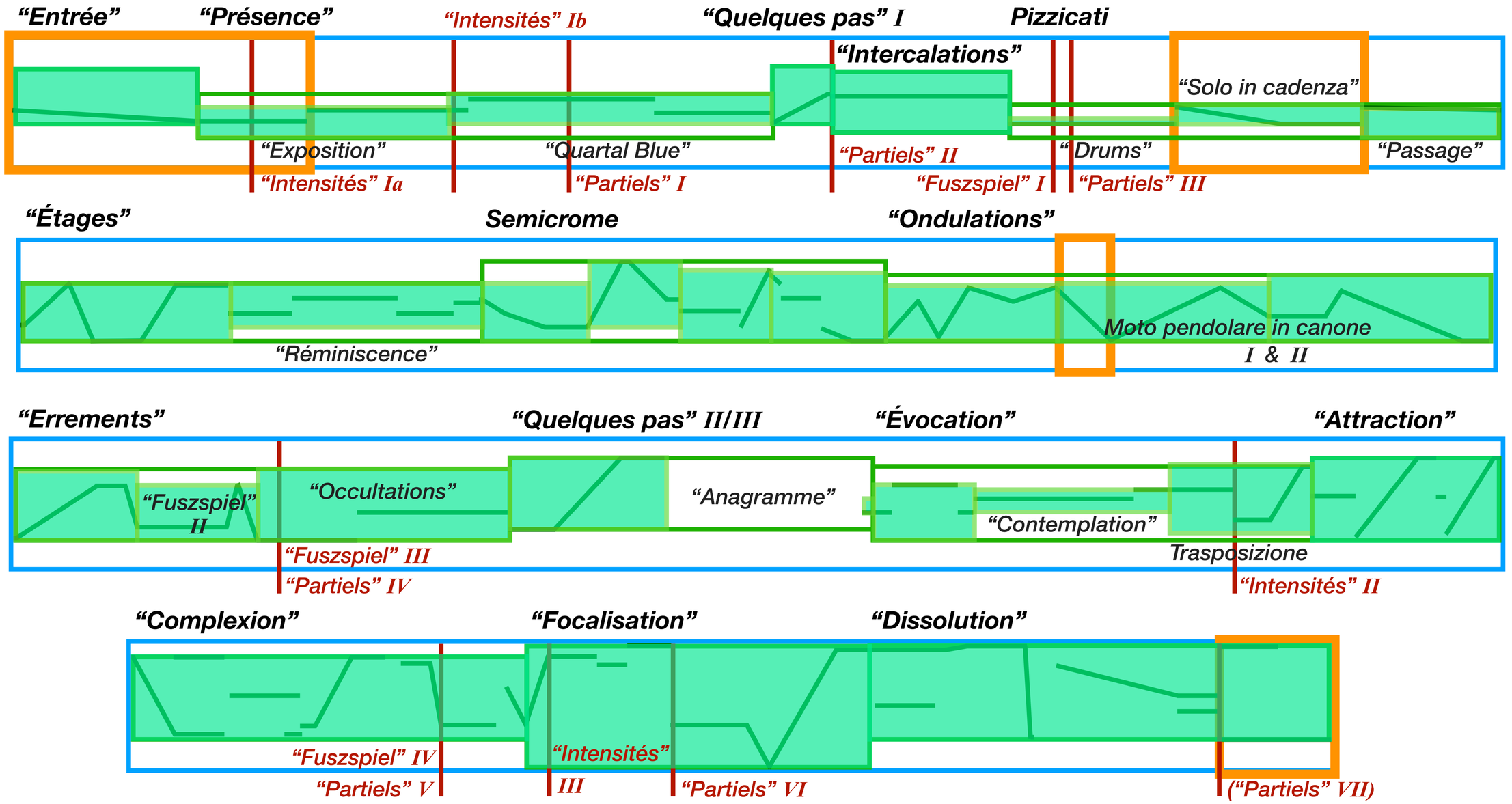


red lines: start points of phenotype occurrences

Formal structure and tonal space: A “dream narrative”



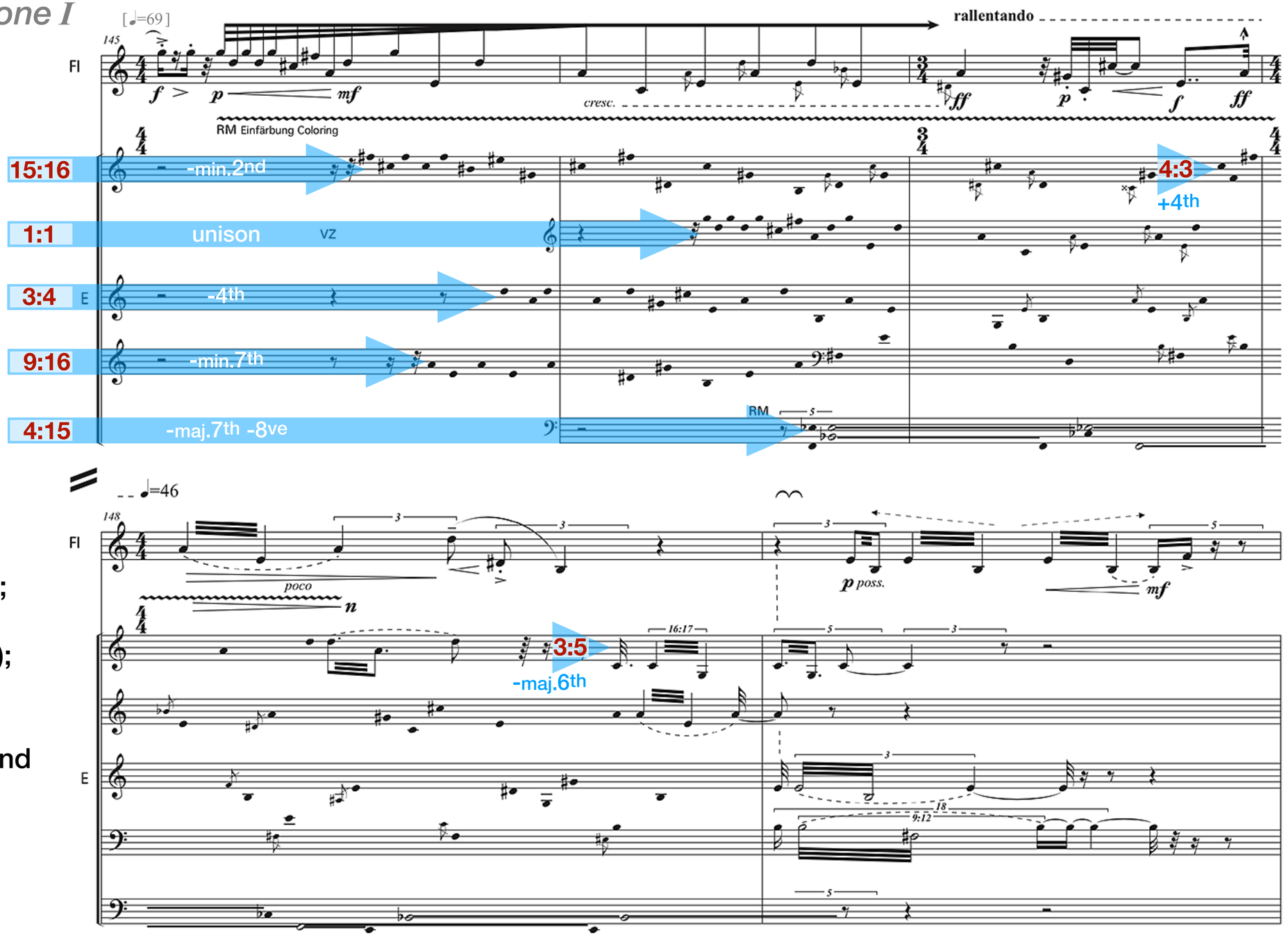
Formal structure and tonal space: **Examples (excerpts) in this presentation** orange outlines



Tempo and delay

Canon with five comes
(speed transpositions)

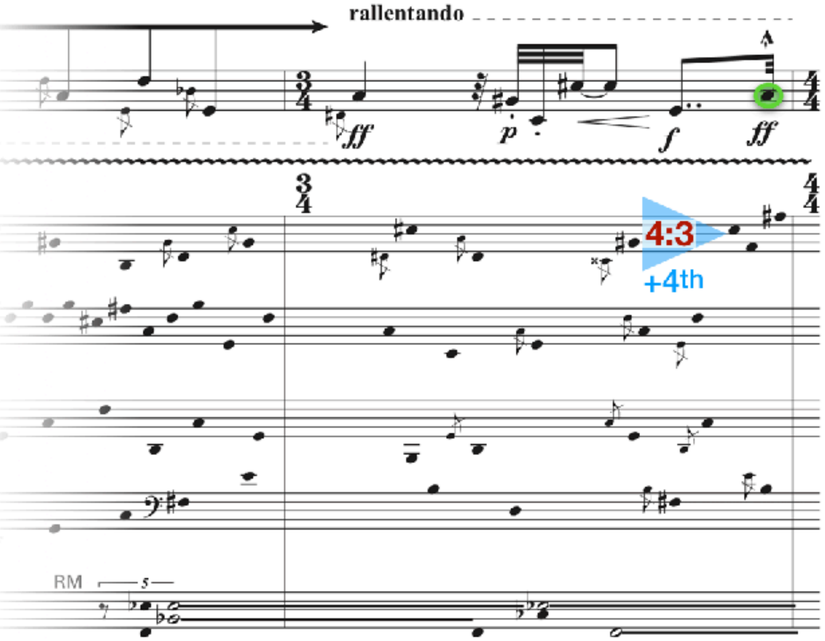
- *Comes mutabilis*: changes transposition, temporarily in diminution (accelerated);
- Comes in unison;
- A fourth slower (and lower);
- A minor seventh below;
- *Comes bassus duobus*: almost four times slower and deeper, middle portion of specimen in interlaced doubling, enriched by ring modulation.



Tempo and delay

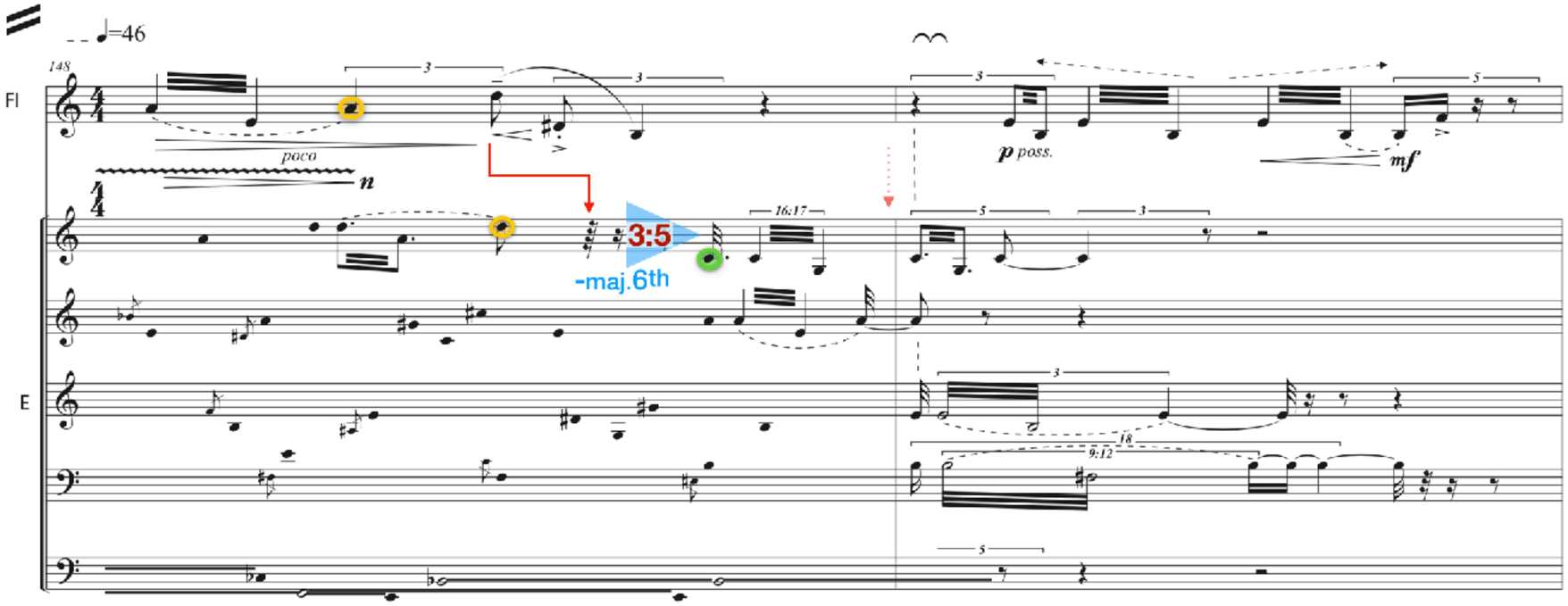
When increasing the playback rate we must take care not to look in the past for events which did not yet occur.

At a speed rate of 4:3 *Comes mutabilis* consumes time 33.3 % faster than the original. In measure 148 it is about to pass beyond the time horizon. With a skip down to a speed rate of 3:5 the voice leaps back, so to say, into the past (behind the original).



current offset

time horizon



Moto pendolare in canone I

$$t_r = t_e + (t_k - t_a) \times (1/s - 1)$$

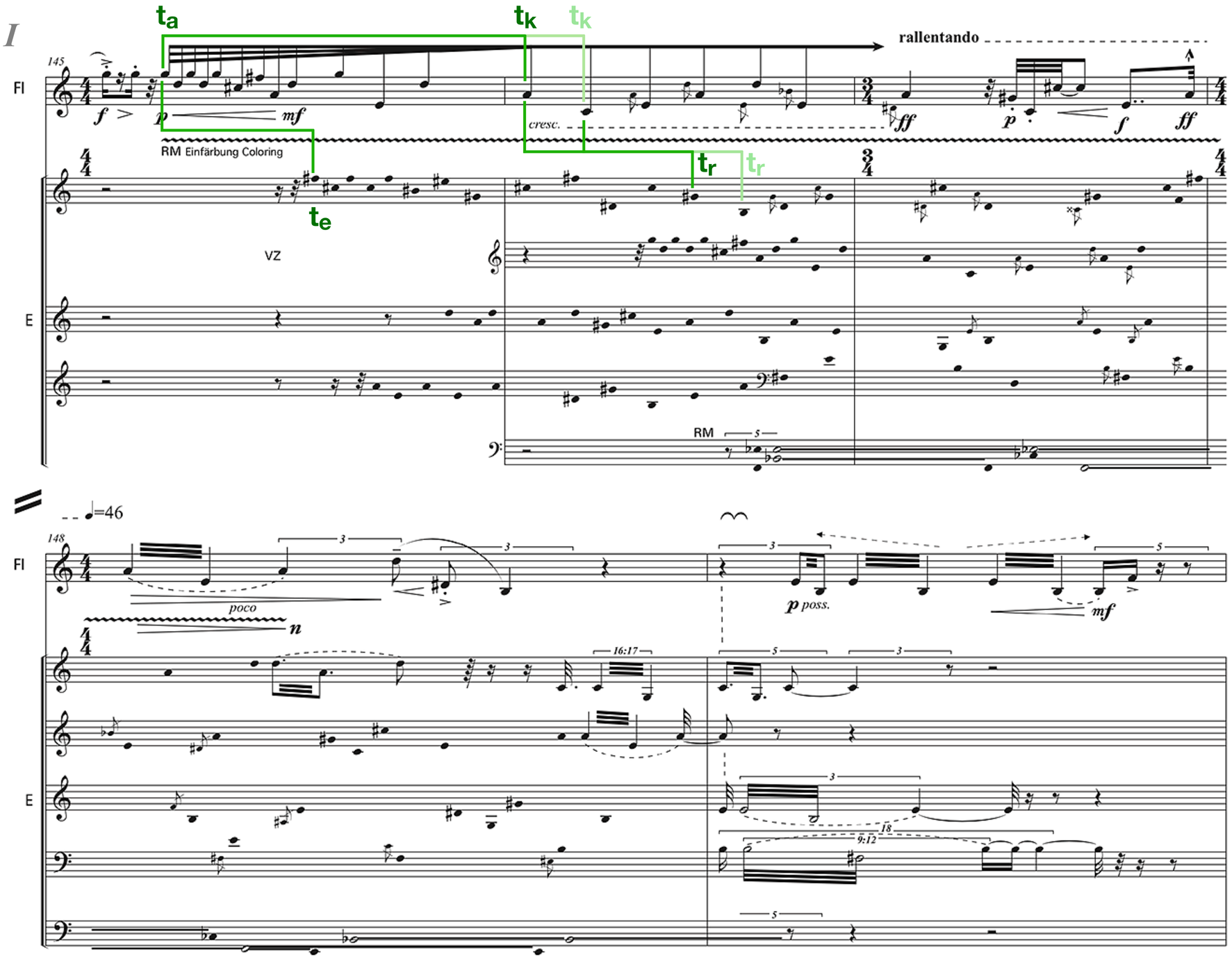
where

- t_r the remaining time
- t_e the entry distance
- t_k the current time
- t_a the start time
- $1/s$ the inverse of the playback speed

Formula to compute the distance of equivalent time points.

current offset

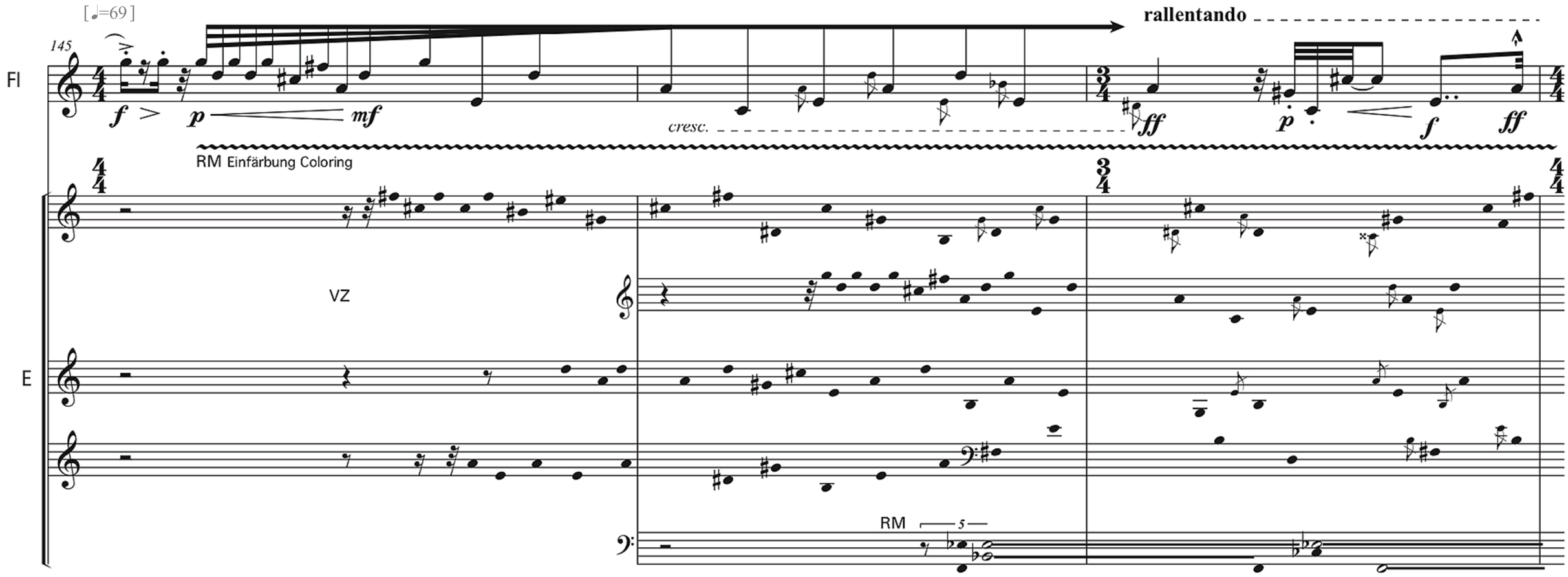
next offset

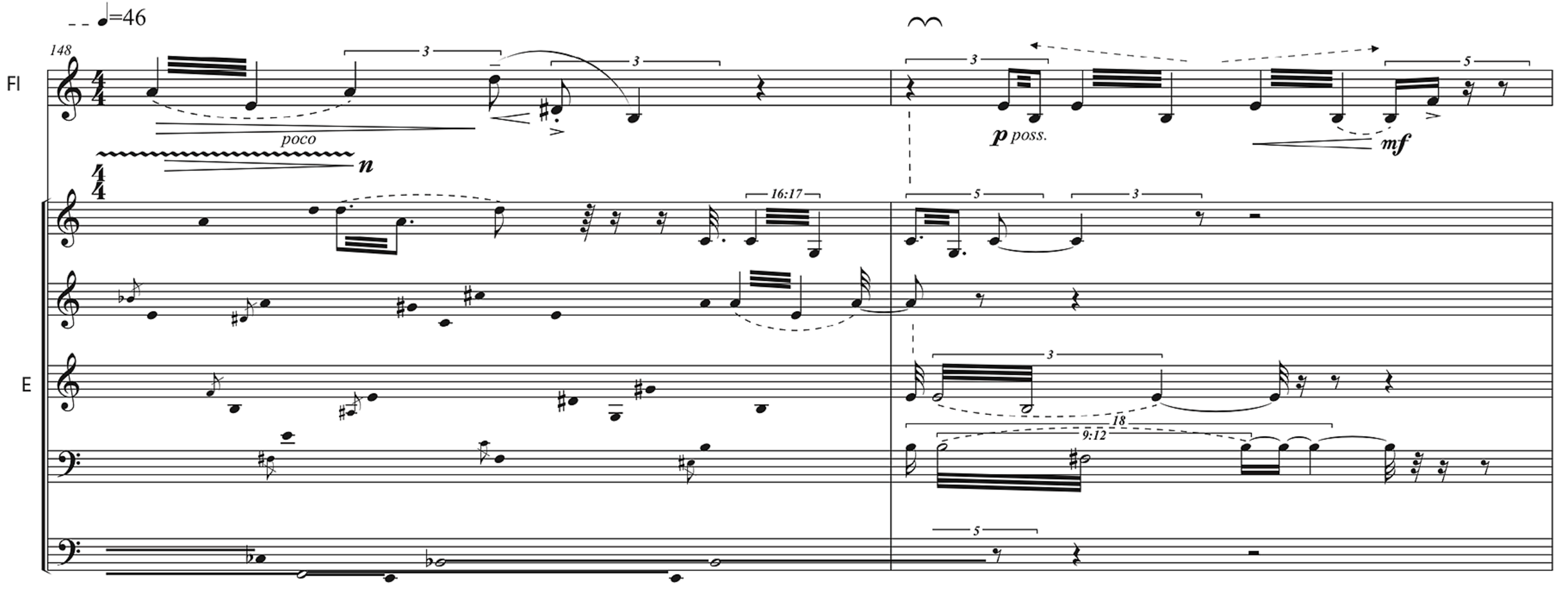


Moto pendolare in canone I

Excerpt m144-150

Although a polyphonic structure, it's not about distinct lines. This mirroring of the flute, with its crossing of wide intervals, aims at densification, entanglement and confusion, like a cloud of fog revealing the flute again when it passes by.




♩=46
148
Fl
poco
n
p poss.
mf
16:17
E
18
9:12

Synchronisation and…

Baseline

- Antescofo follows pitch changes (considering energy).
  - Latency is inherent in pitch detection.

Reminder

- Antescofo = ***anticipatory** score follower*
  - If possible, align actions towards events in the future.

synchronisation
- perform something at the same time or rate
- align actions to events

anticipation
- imagine or experience an event in advance

latency
- time delay between cause and effect
- in FFT the duration of a frame (FFT size)
- the more precise (larger FFT size), the later

…onset detection

onset
- start time of the initial transient of a sound

To get information about onsets, an additional tool is needed.

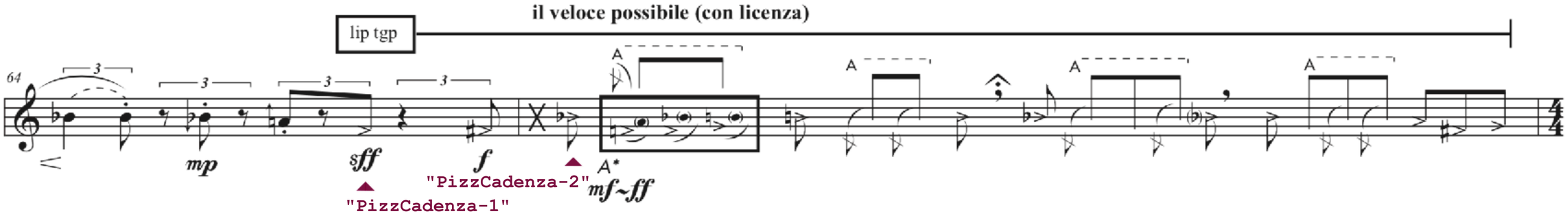
IRCAM's MuBu/PiPo provides it.

Caveat

- Onset detection parameters must be adapted to characteristics of the sound to be analyzed.

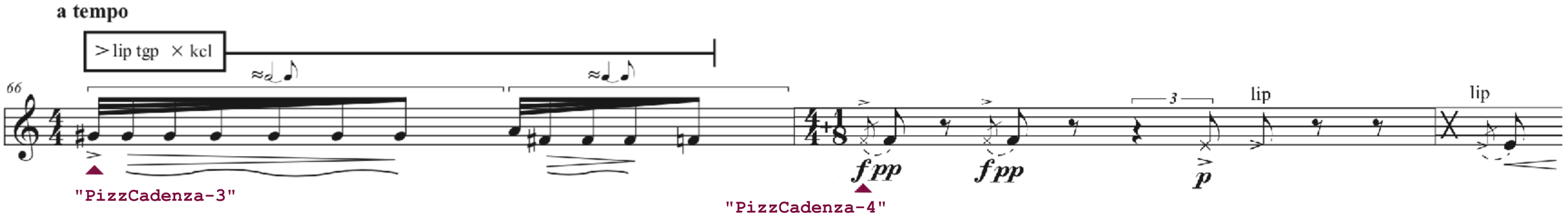
- fast or slow attack?
- stationary sustain or fluctuations?
- how strong do events stand out from the background?
- …

Onset detection



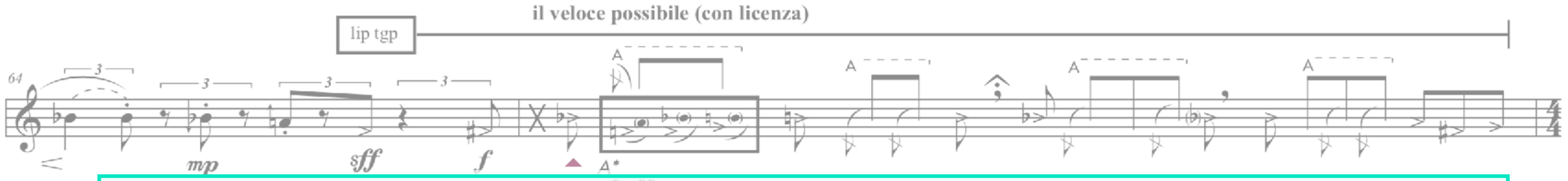
In the Antescofo score of “Vertical Structure” the detection times of selected events are stored as offsets into the recording buffer, for later reference.

The four points shown here, for example, indicated by purple arrowheads, are named *PizzCadenza-1, PizzCadenza-2, PizzCadenza-3, PizzCadenza-4*.

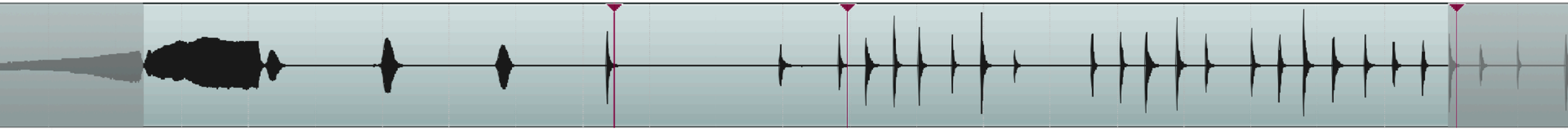


Due to the percussive character of the flute pizzicati, Antescofo’s detection lags a bit.

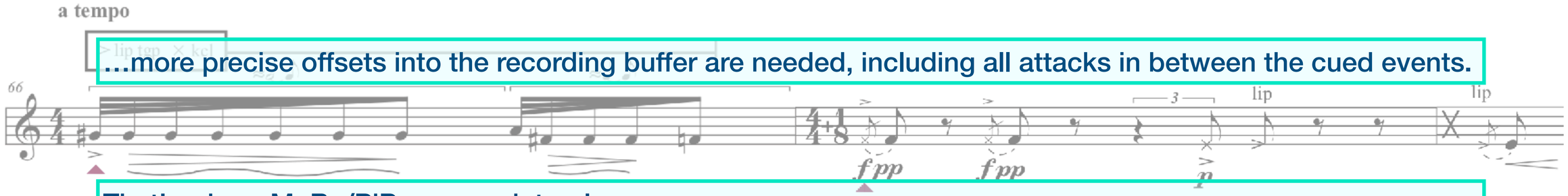
Onset detection



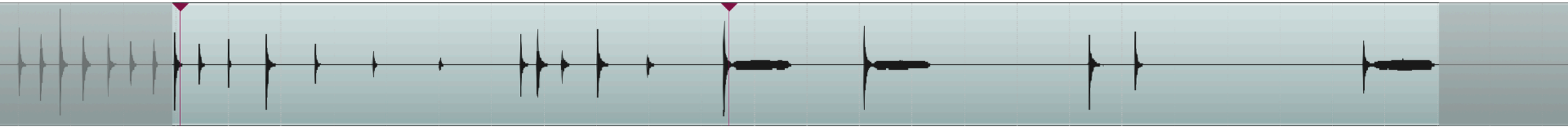
As our intention is to replay the air pizzicati in different order and timing…



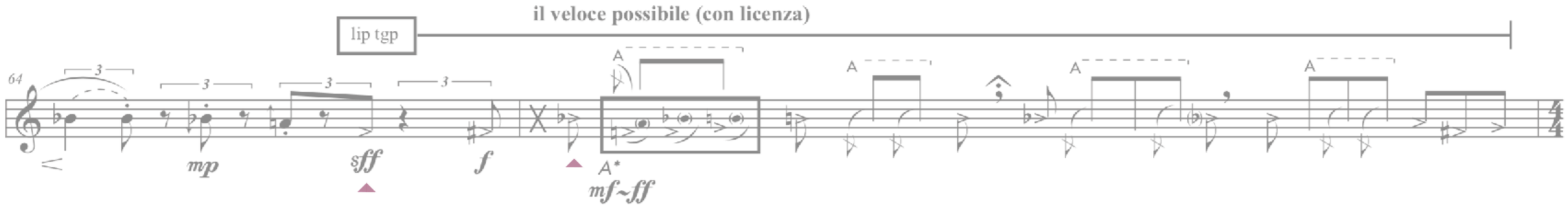
…more precise offsets into the recording buffer are needed, including all attacks in between the cued events.

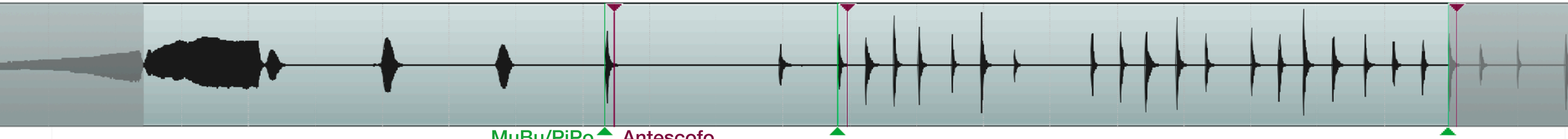


That’s where MuBu/PiPo comes into play.



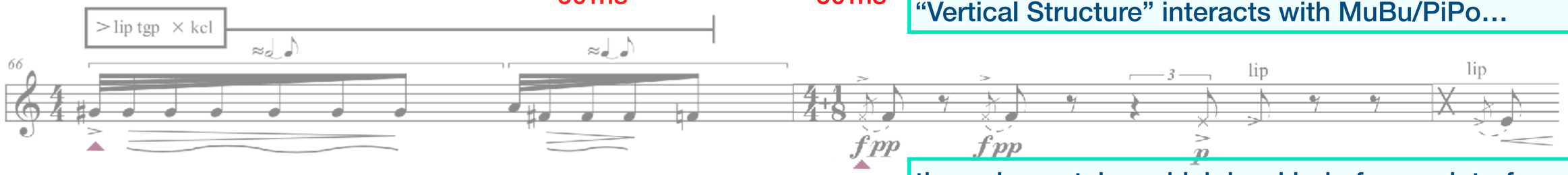
Onset detection





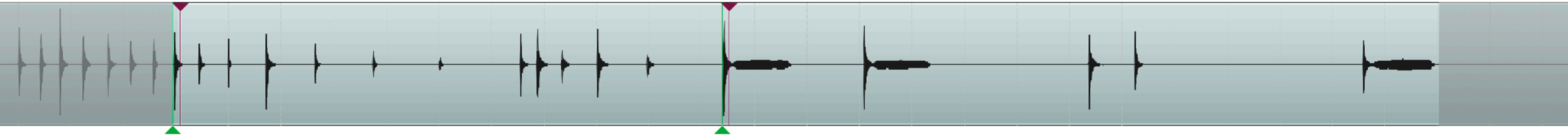
MuBu/PiPo Antescofo

60 ms 60 ms



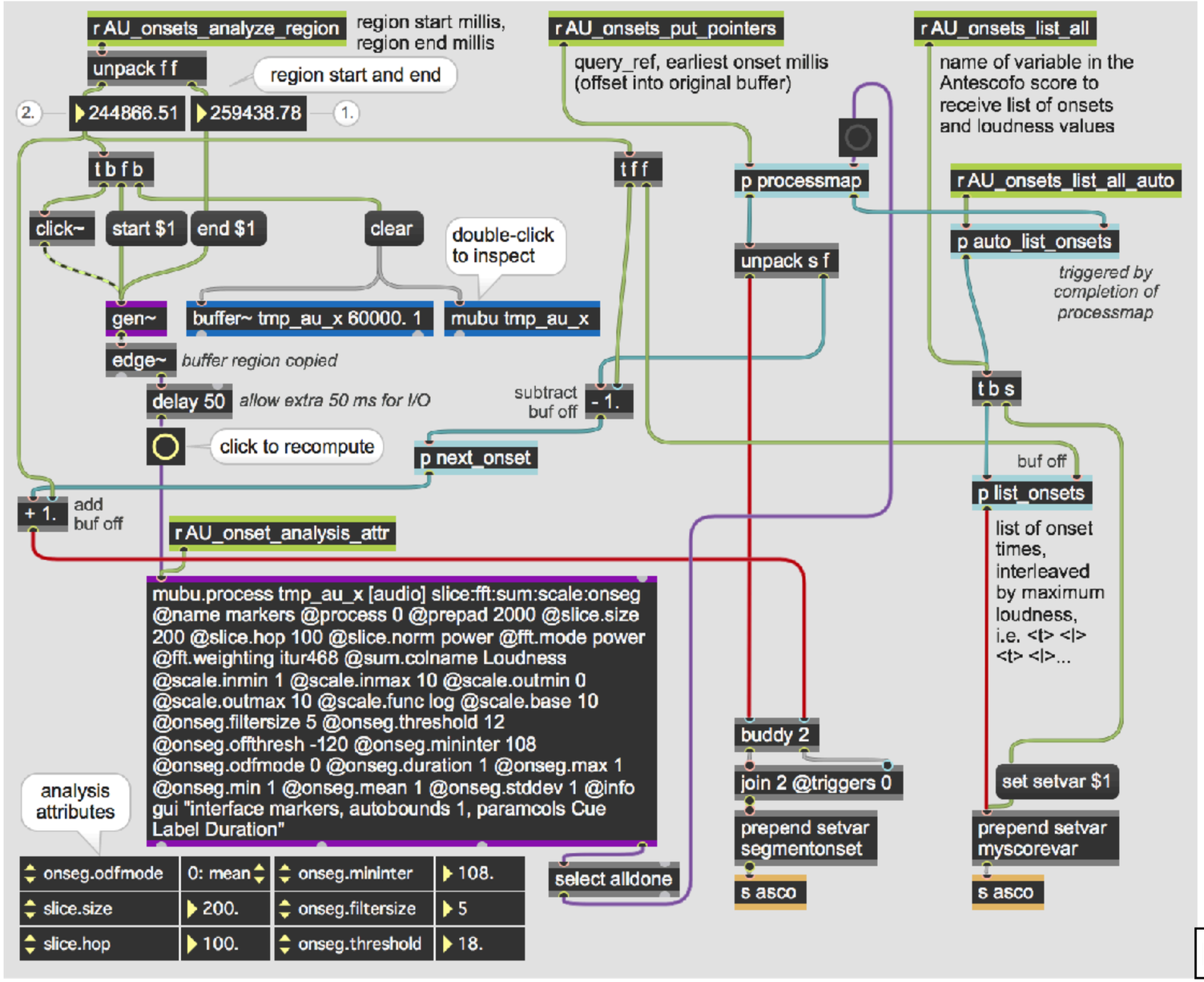
"Vertical Structure" interacts with MuBu/PiPo…

through a patcher which is a kind of query interface…



57 ms 56 ms

Onset analysis queries



We send an analysis request to the patcher and receive the result in an asynchronous callback.

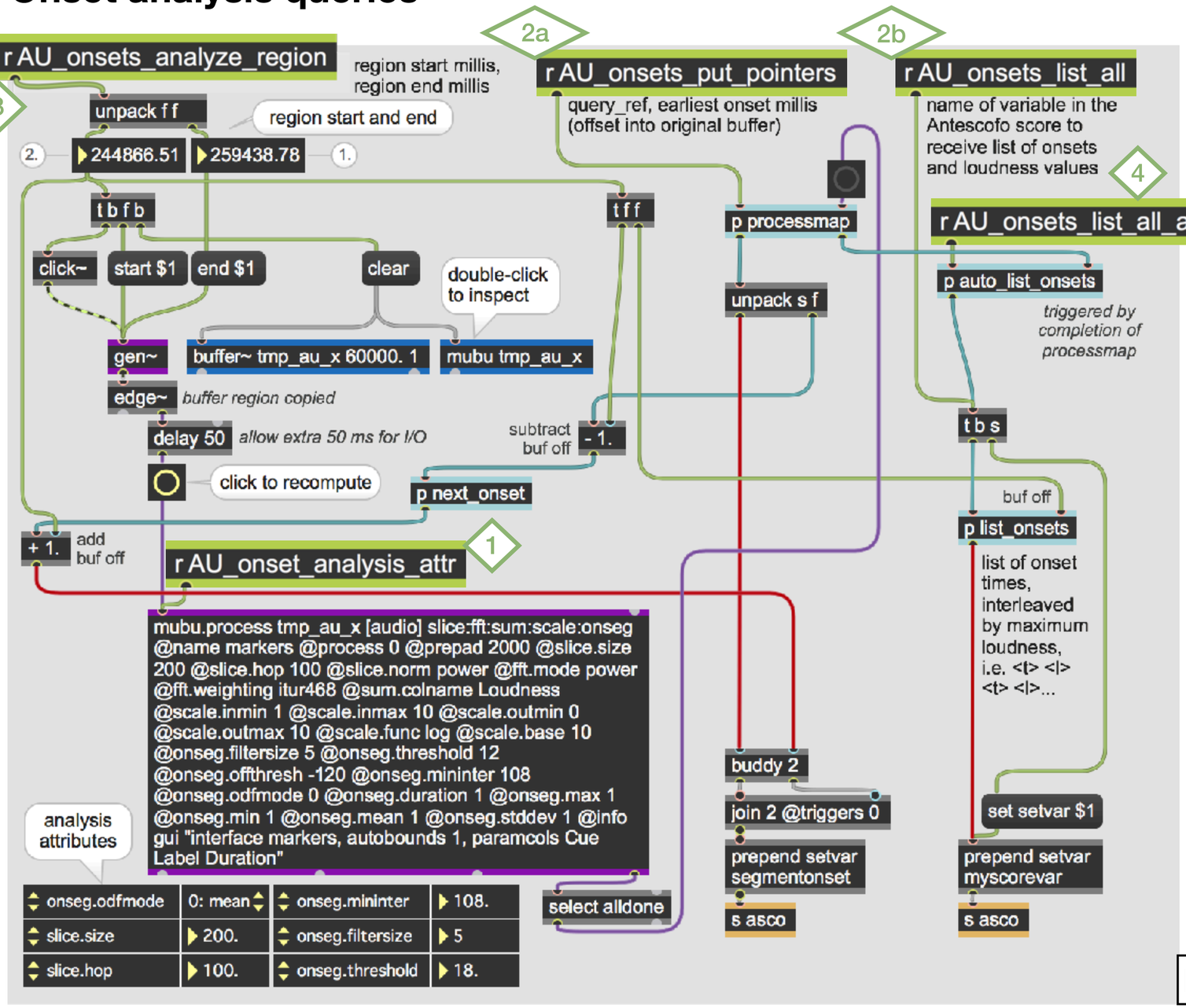
It's likely that analysis parameters (attributes) have to be adjusted. The values listed at the bottom left here are those used for the flute pizzicati.

References to be used in the callbacks may be registered before triggering the analysis. These may be key/value pairs which associate a symbol with an earliest onset time or the name of a variable to hold a complete list of all onsets found in the region. The latter may also be requested after processing the region.

The region is copied to a local temporary buffer and then processed by MuBu. On completion of the analysis all open callbacks are executed.

Max patcher *onset_analysis*

Onset analysis queries



1

eventually fine-tune analysis attributes (of pipo.slice and pipo.onseg)

2

a put a list of named pointers (results will be sent to variable **`segmentonset`**)

and|or

b to get ***all*** onsets in the region specify the name of a variable to hold the list

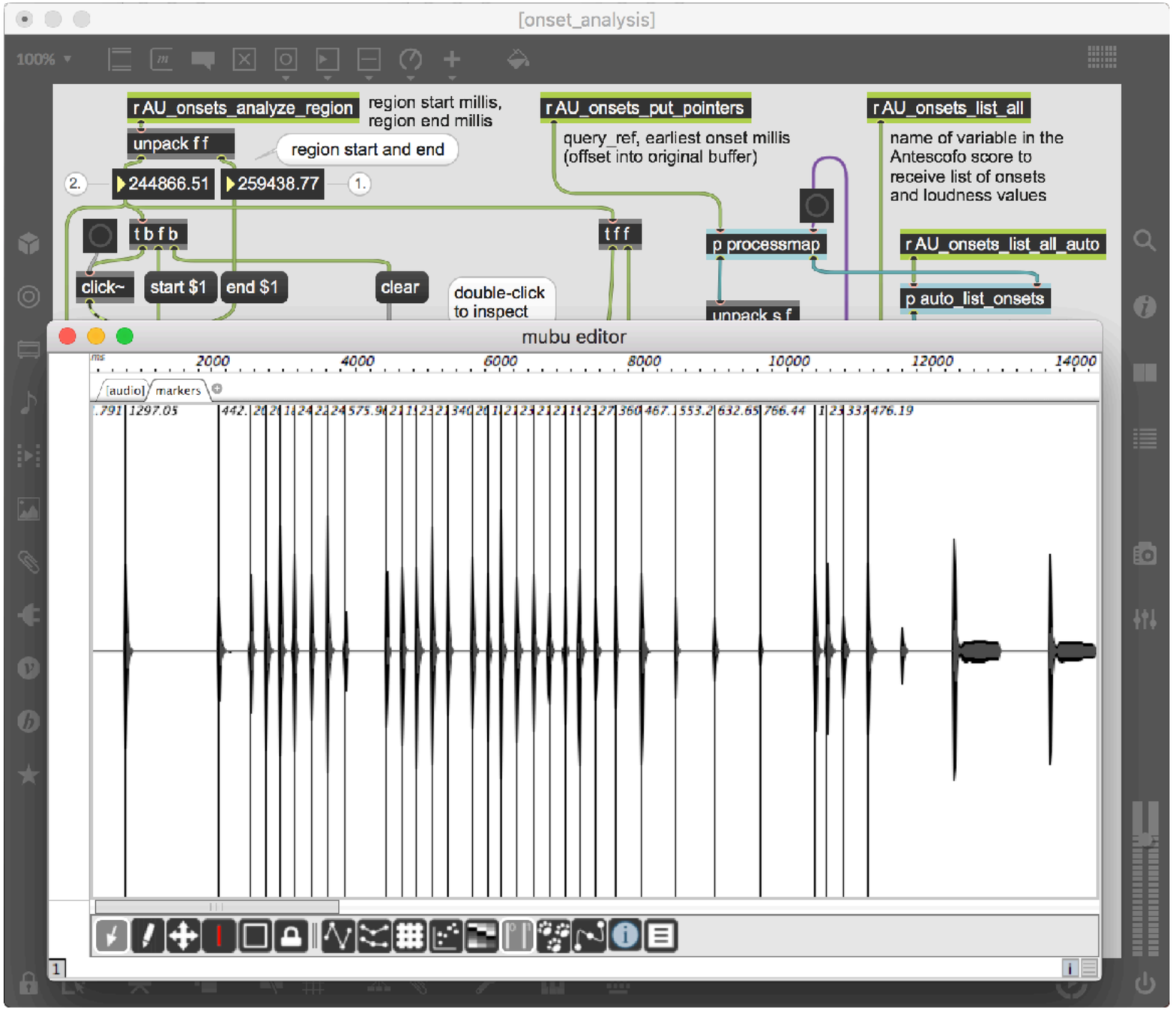
3

trigger analysis of a given region in the recording buffer

4 request immediate callback with the list of all onsets

Max patcher ***onset_analysis***

Onset analysis queries



Max patcher *onset_analysis* with **mubu** window

Onset analysis queries

The mechanism in Antescofo which enables us to react to the variable being filled with the result of the query is the powerful ***whenever*** construct.

A whenever watches a variable. Execution of its body is triggered on every change to the variable.

Here, when the value of the variable **`$pizzcad_onsets_raw`** is updated with the list of onsets, the list is reshaped and reordered to create an accelerated and randomized recombination of the flute's pizzicati.

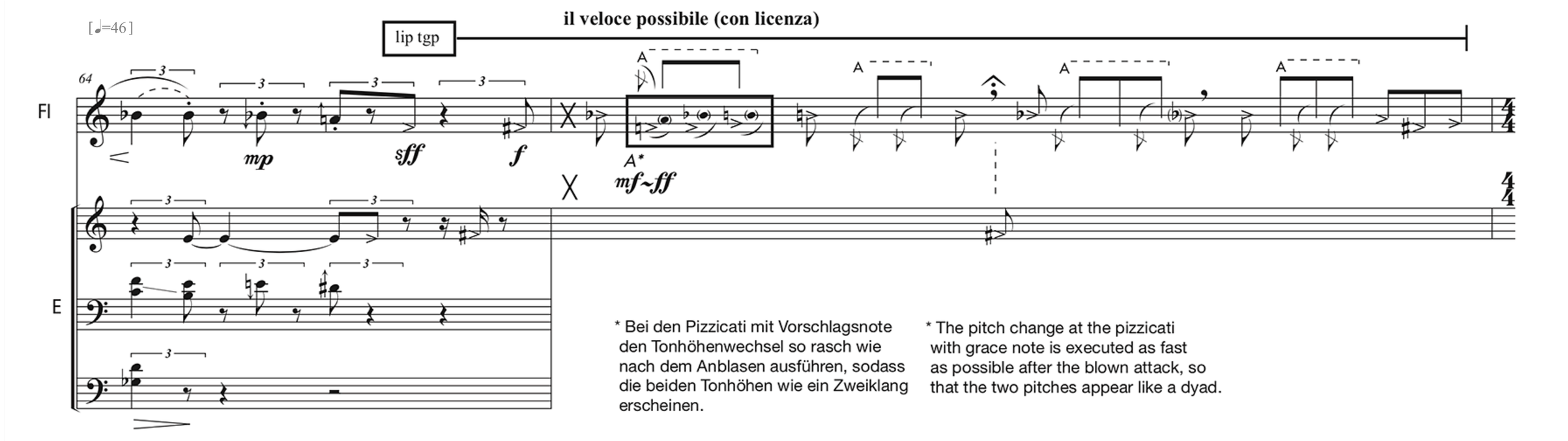
Antescofo score

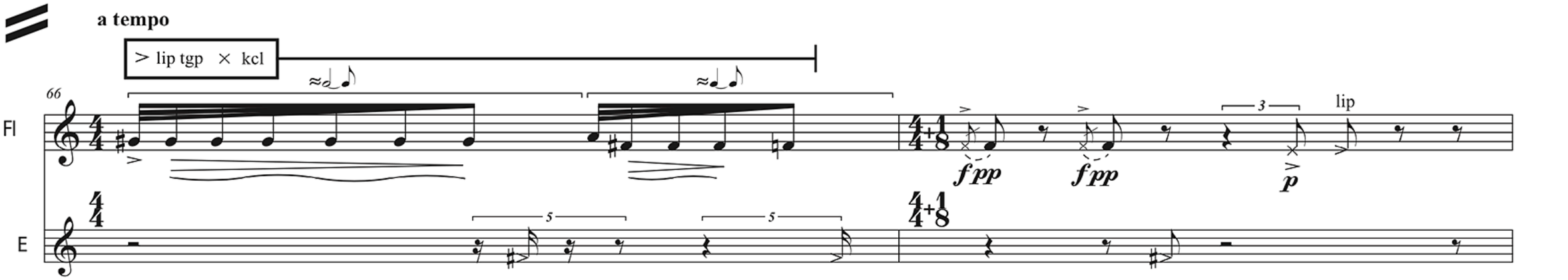
```
  }
NOTE 0  1/2
NOTE 0  2/3
NOTE E4 1/3 @pizz @staccato ; key-click
  group OnsetRegionAnalyze_PizzCadenza {
    @local $rgn_start, $rgn_end, $pointers, $flattened
    let $pointers := tab ["PizzCadenza-1", "PizzCadenza-2",
                          "PizzCadenza-3", "PizzCadenza-4"]
    let $rgn_start := @au0ptr_pos("PizzCadenza-1") - @b2ms_r(1)
    let $rgn_end := @au0ptr_pos("PizzCadenza-4") + @b2ms_r(5/2)
    AU_onset_analysis_attr onseg.threshold 18 // not too many onsets
    let $flattened := @lace([ $pointers, @map(@au0ptr_pos_for_onsetsearch,
                                              $pointers) ],
                            (@size($pointers) * 2))
    AU_onsets_put_pointers $flattened
    AU_onsets_analyze_region $rgn_start $rgn_end
  }
NOTE E4 1/2 @pizz @staccato ; ‹pizzicato› (lip)
NOTE 0  1
NOTE E4 1/2 ; ‹pizzicato› (lip) + ord.
  @global $pizzcad_onsets_raw
  group PizzCadenzaReplay {
    AU_onsets_list_all pizzcad_onsets_raw
    whenever PizzCadenzaReplay ($pizzcad_onsets_raw) {
      @local $earliest, $earliest_i, $i, $onsets
      let $earliest := @offms("PizzCadenza-1")
      let $i := 0
      let $earliest_i := -1
      let $onsets := @reshape($pizzcad_onsets_raw,
                              [@size($pizzcad_onsets_raw) / 2, 2]))
      forall $x in $onsets {
        if ($earliest_i < 0 && $x[0] >= $earliest) {
          let $earliest_i := $i
        }
        let $i := $i + 1
```

Replay with onset data

In this passage the computer disregards the physiological conditions of a flute pizzicato...

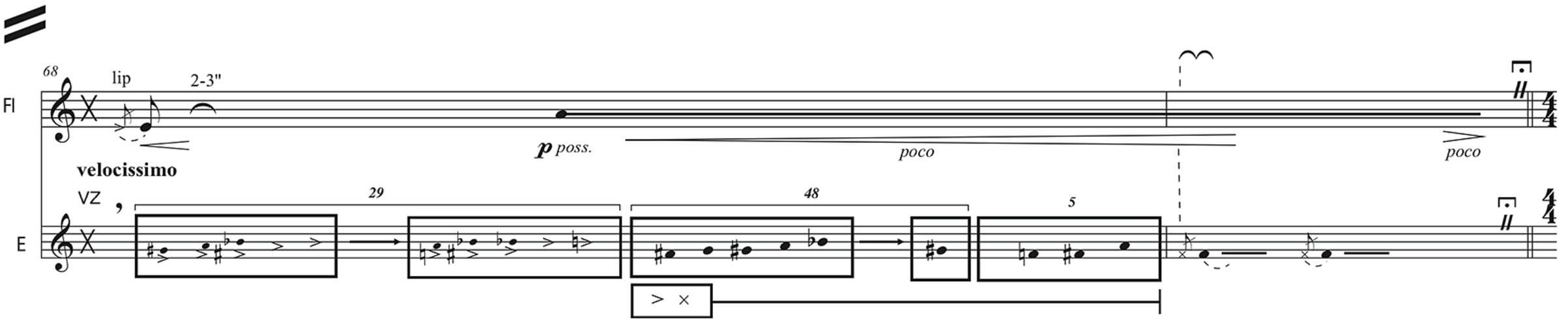
Excerpt m.63-70





Replay with onset data

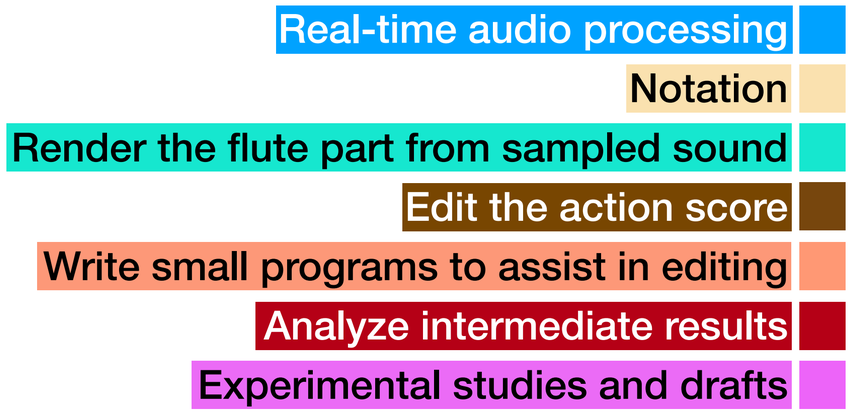
...to create an exaggerative parody of an occasional surge of fury expressed by the flute.



Practical insights

With the exception of scribbled notes, *kritzelzettel* in German, and first sketches on paper, all my work takes place on a mid-2015 Mac Book…

Software used for “Vertical Structure”



Max/MSP

Externals and abstractions:

IRCAM Antescofo

IRCAM MuBu/PiPo

IRCAM Spat

Finale

Sample library:

IRCAM Solo Instruments

Sublime Text 3

with Antescofo package (syntax highlighting)

Eclipse (Java)

IRCAM AudioSculpt, Sonic Visualizer

Reaper

A composer’s work today, especially when she employs sound processing and/or analytical or generative algorithms on the computer, not only takes place at a digital workstation but in fact resembles that of a software developer.

Typically, she uses – at different, partly iterative stages of the compositional process – a couple of different applications and, a lot of files are being produced.

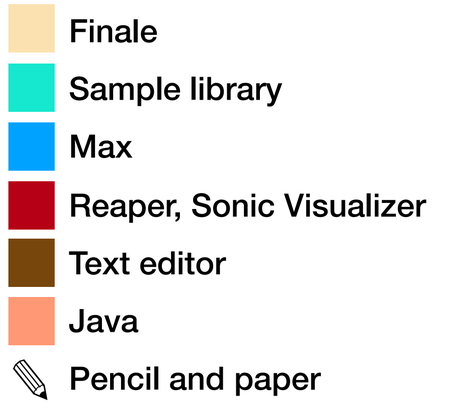
It seems a good idea to adopt tools and best practices used in software development.

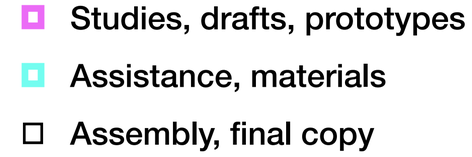
Typical work stages for a section/passage of “Vertical Structure”

- ❖ **Write down key words, outline, graphical or notation sketches** ✎
- ❖ ***Eventually* write studies in Finale**
- ❖ **Write/notate flute part in Finale**
- ❖ **Render flute part from sampled sound**
- ❖ ***Eventually* create new patcher or extend existing one in Max**
- ❖ ***Eventually* make studies for the electronic part**
  - ✤ creating patchers in Max to try out variants and tweak parameters
  - ✤ using pitch transposition and time stretch functionality in Reaper
- ❖ ***Eventuall*y write a small program in Java to compute parameters**
- ❖ **Make notes to retain intermediate results** ✎
- ❖ **Write/notate result score of electronic part in Finale**
  - ✤ omitting dynamics, spatialisation and other details
- ❖ **Write actions/instructions for electronic part in Antescofo score** [+]
- ❖ **Test in (home) studio***
  - ✤ analyzing the output
  - ✤ amending details if needed
- ❖ **Re-examine in the context of the whole**

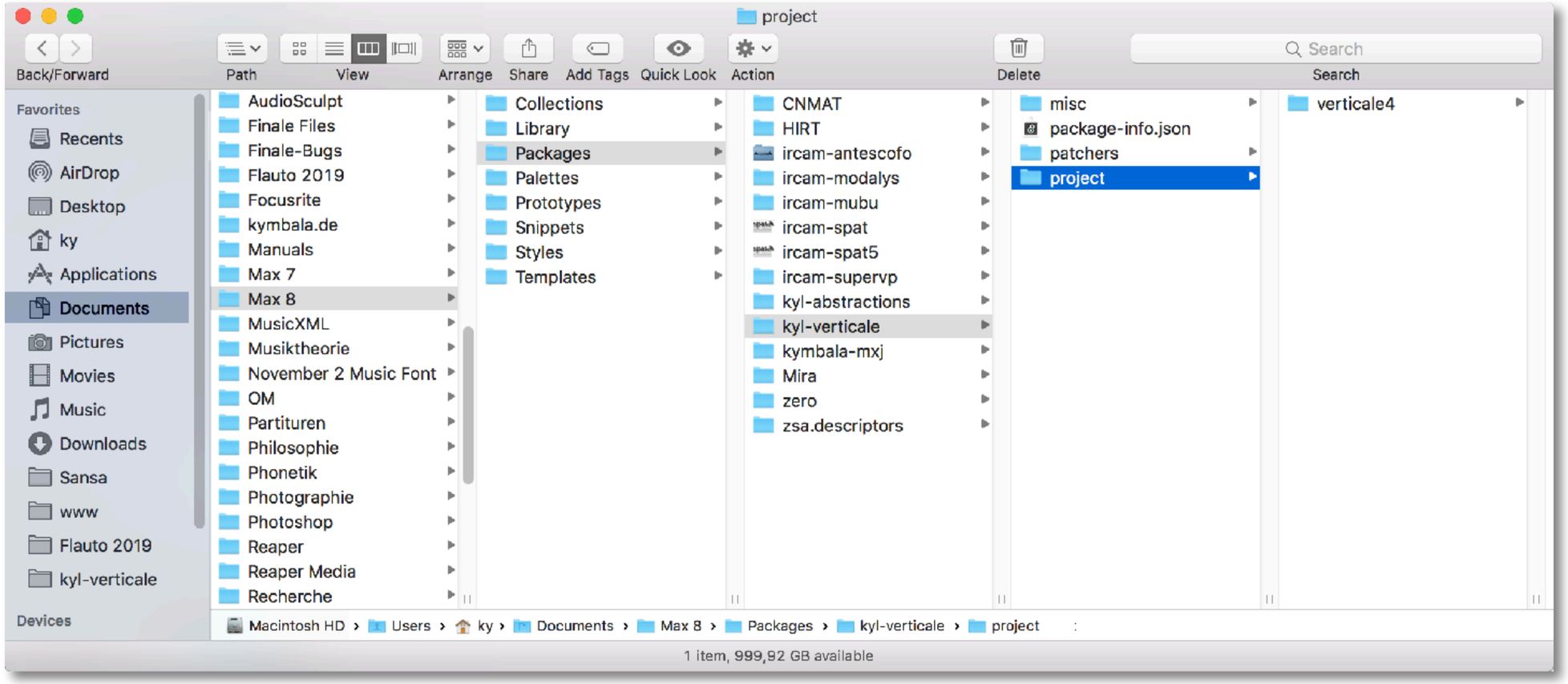
*audio interface and four loudspeakers in my small 8 m² study

Share results, collaborate with others, rehearse with musician, go into production





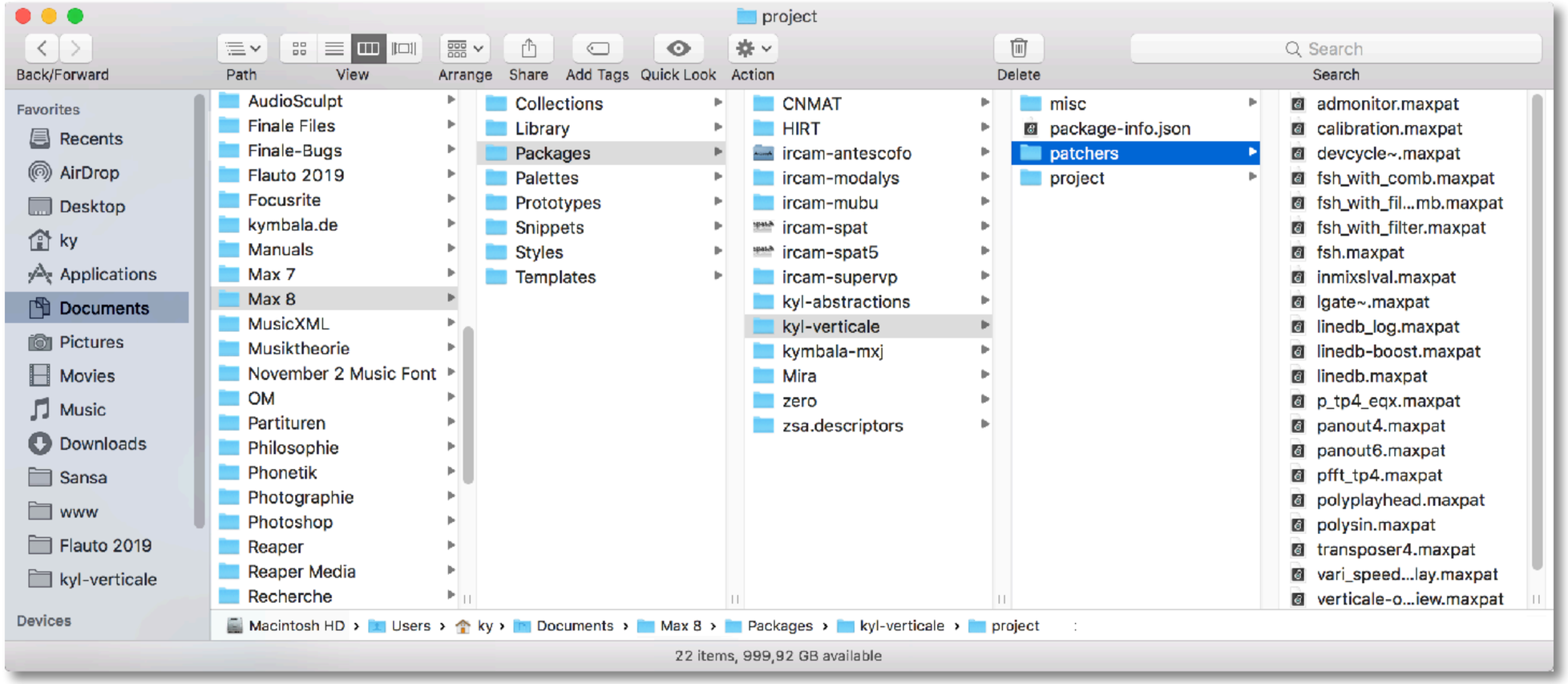
File organisation: All in one place



- Package `kyl-verticale`
- Project `verticale4` (4-channel version)
  - Project inside package: all in one folder

I like the package and project concepts of Max and for “Vertical Structure” ended up combining both by placing the project folder inside a dedicated package for the project.

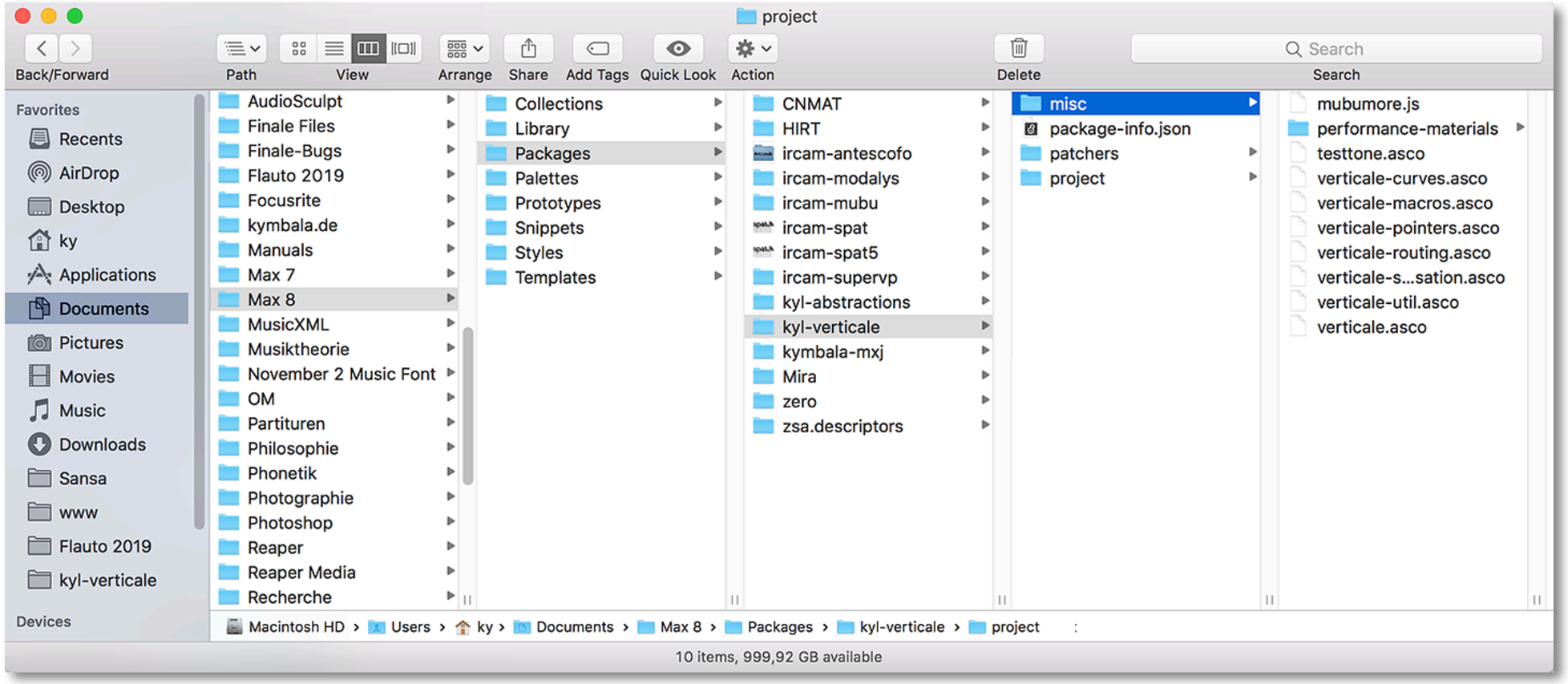
File organisation: Max patchers



- Patchers
  - Generic patchers in package `kyl-verticale`
  - Main patcher in project (`verticale4.maxpat`)

Max patcher files are text-only files. Because of this, not only they can be shared easily but searched and compared with standard text-based tools, e.g. version control management.

File organisation: Miscellaneous



- `misc`
  - Files used within but not edited with Max: Antescofo score
  - Files not handled by Max: e.g. Finale files, documentation

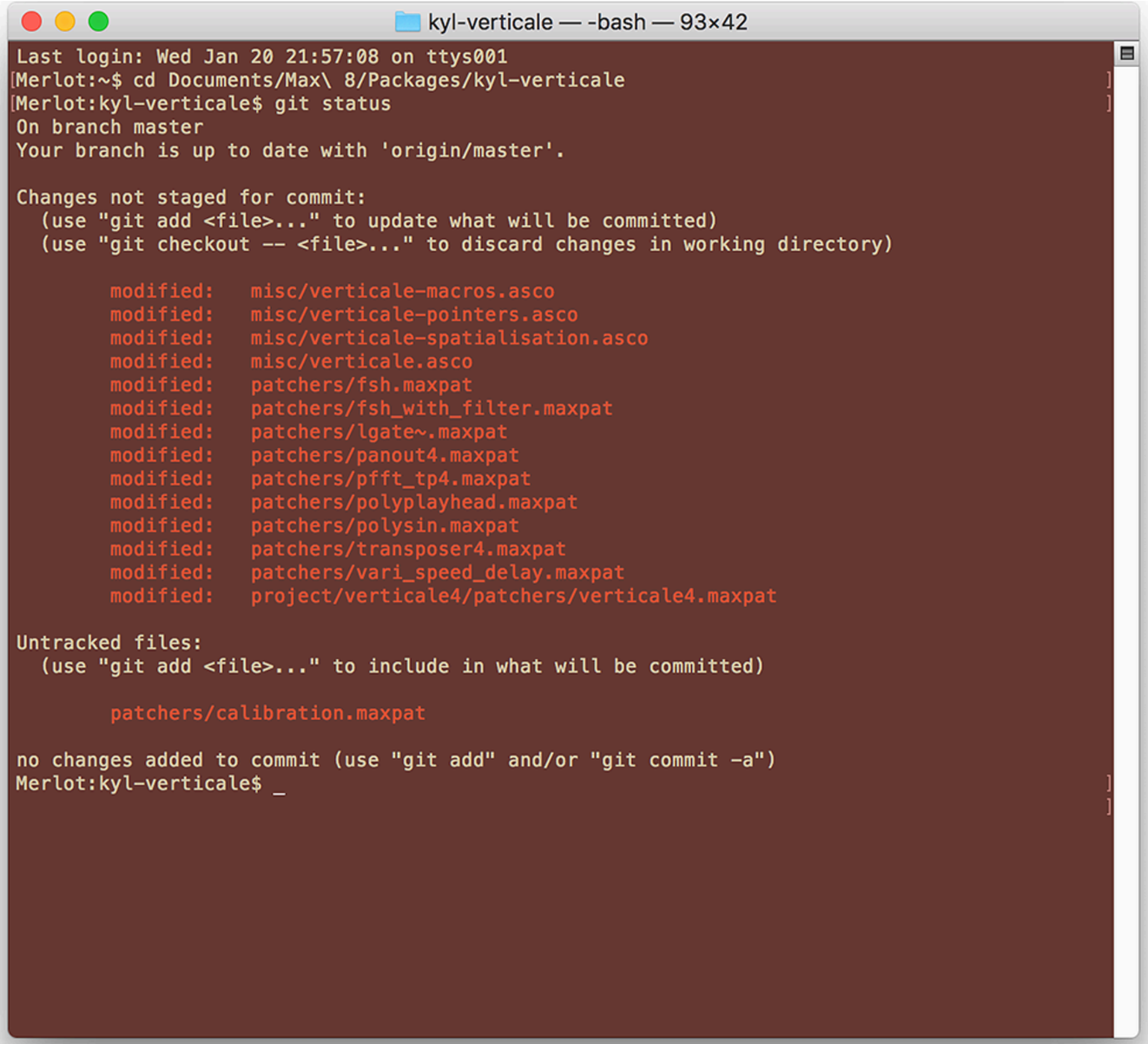
You can put any type of file in a Max package. Max regards them as additional resources or ignores them.

Tracking changes with Git

Once you have all source files – Max patchers, Antescofo score etc. – in one folder, it's easy to set up version control management. The standard tool in the open source world today for tracking changes is Git.

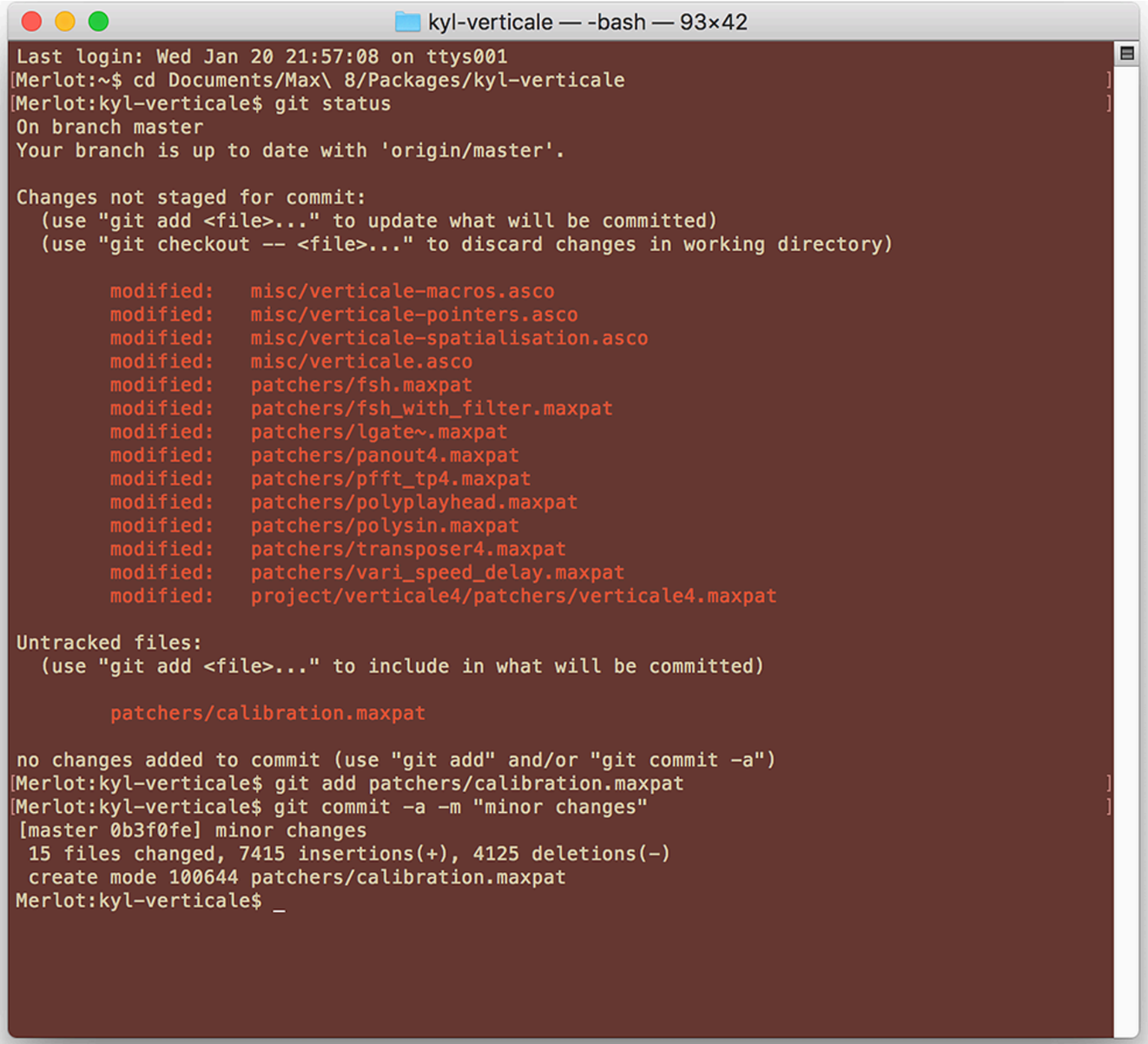
Git tracks your changes and can show differences between revisions and, in the event, let's you go back to a revision. You can try out things on a branch and, if they work, merge the changes to the master tree.

If you're not into the console, there are graphical clients, and furthermore, the web interfaces of repository hosting services.

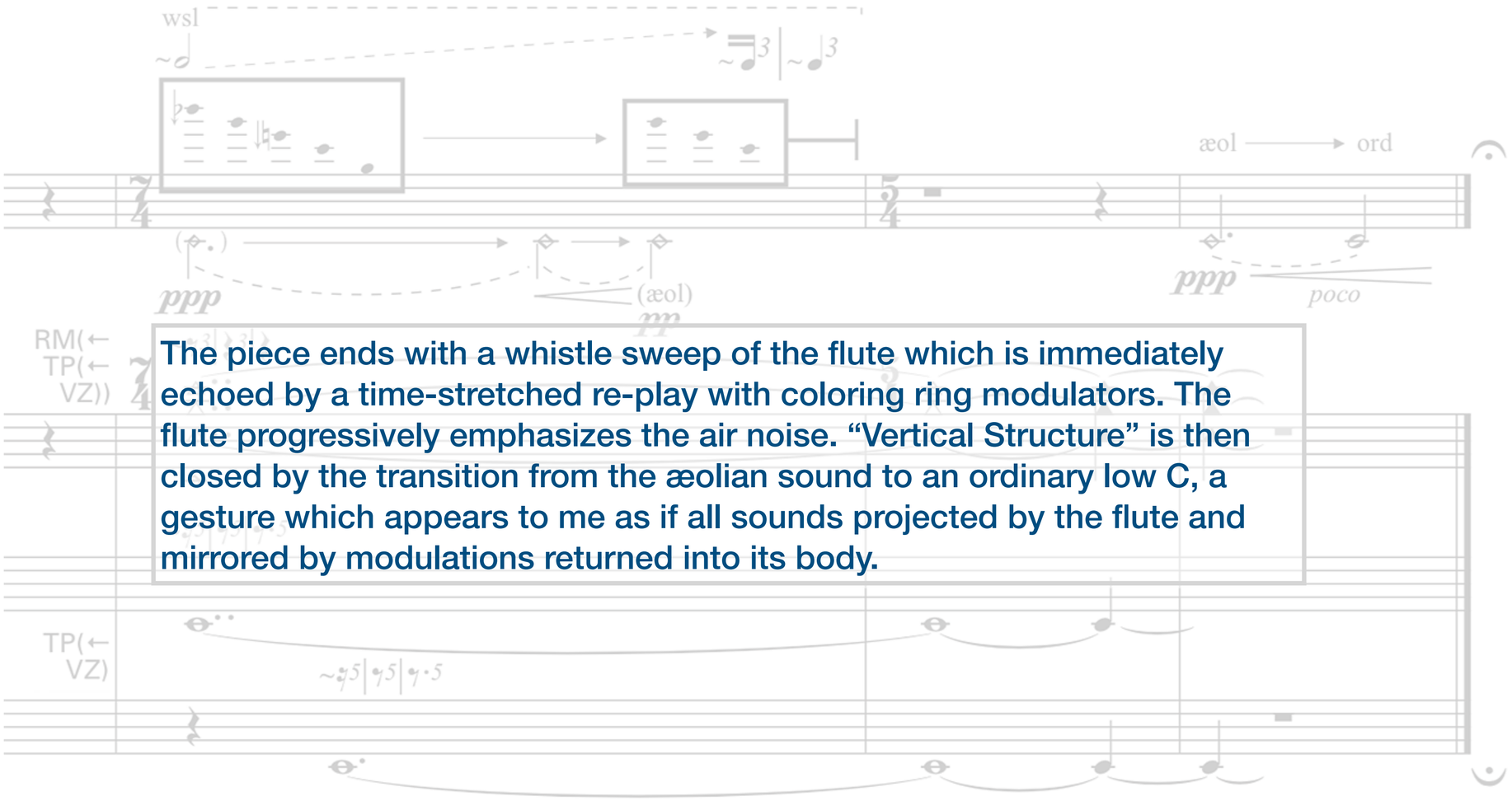


Tracking changes with Git

- List changed files
- Commit (or discard) changes
- Track revisions
- Compare differences
- “Undo” on diverse levels
- Work on a copy of the file tree (branch)
- Merge (integrate) changes
- Save on a remote system (push)
  - Back up, implicitly
- Get from the remote system (clone, pull)
  - Work on different machines
- …
- Clients on all major operating systems
  - **CLI** (command-line interface) or
  - **GUI** (graphical user interface)
- Repository hosting services
- (public or private repositories)
  - GitHub, GitLab, Bitbucket a.o.
- Git official web-site ➽ https://git-scm.com/

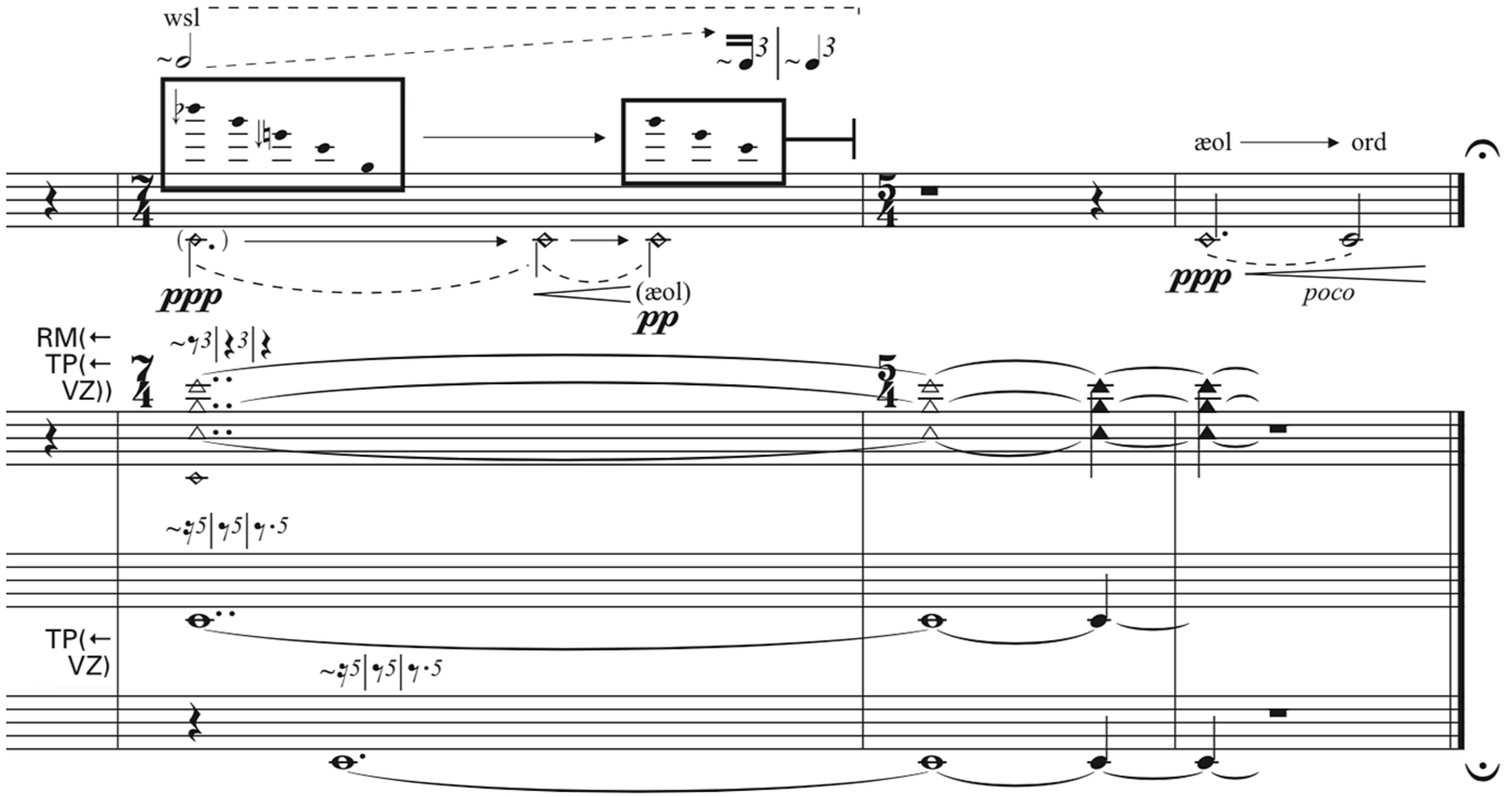


Conclusion



The piece ends with a whistle sweep of the flute which is immediately echoed by a time-stretched re-play with coloring ring modulators. The flute progressively emphasizes the air noise. “Vertical Structure” is then closed by the transition from the æolian sound to an ordinary low C, a gesture which appears to me as if all sounds projected by the flute and mirrored by modulations returned into its body.

Conclusion



▶ ***Excerpt m337-339***

Thank you for listening (or reading).

Supplementary material at
https://kymbala.de/presentation/verticale/

Kai Yves Linden
E-mail: **kylinden@kymbala.de** · Forum IRCAM User ID: **kyl**