

# You'll Hear It

For melody instrument & live electronics

Kier Hall

## How to Perform *You'll Hear It*

### Equipment Required

- Microphone
- MIDI footswitch
- Computer with Pure Data (vanilla) installed.
- *You'll Hear It* electronic resource folder. This includes the patch and supporting files.
- Pure Data will require the following extensions: list-abs, maxlib, cyclone and freeverb.

### Preparation

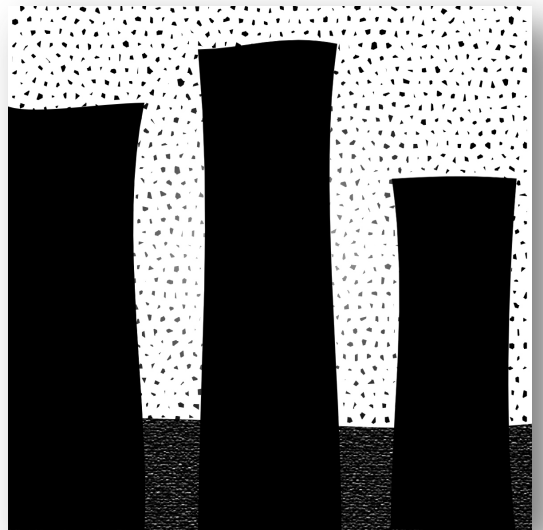
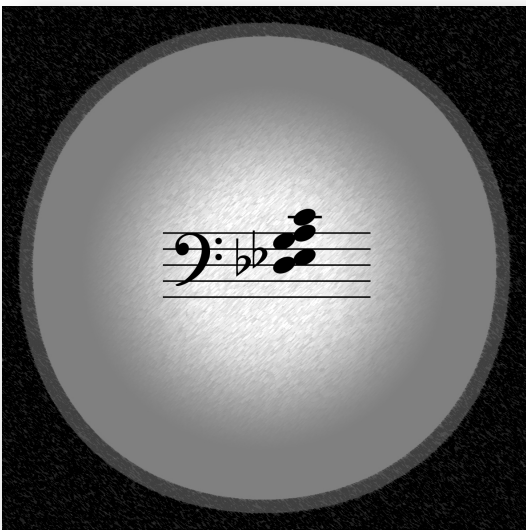
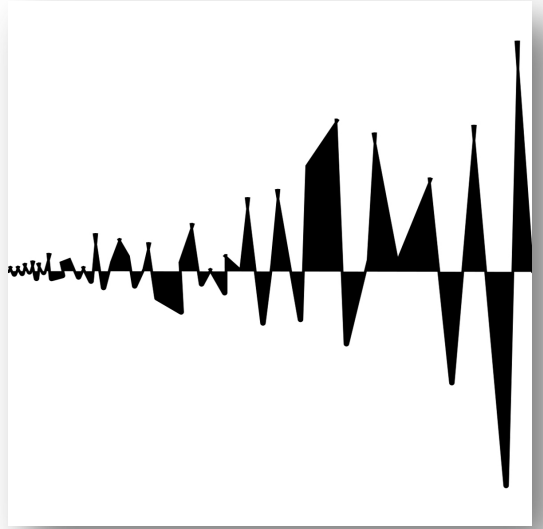
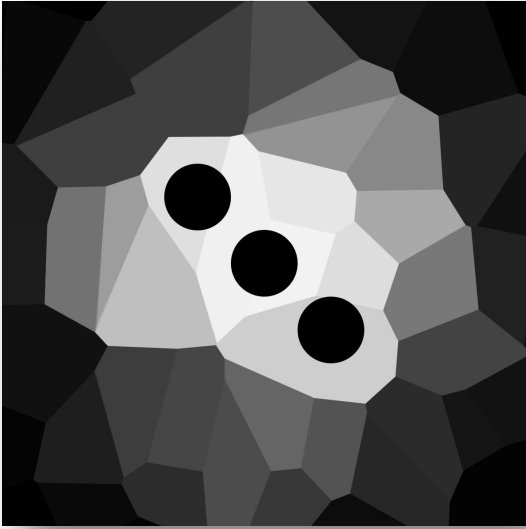
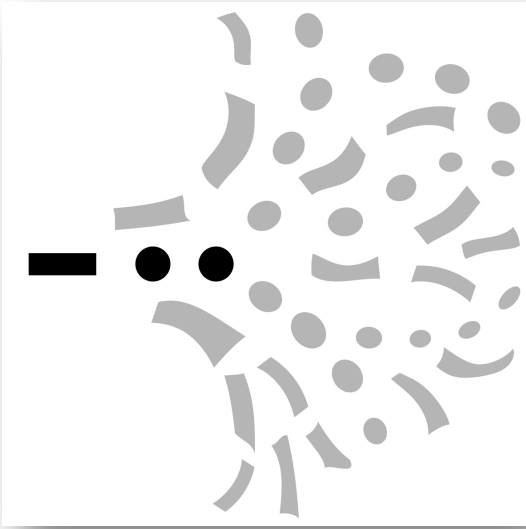
- Print this PDF and cut out the graphic score cards found on the following two pages.
- Connect microphone and footswitch to Pure Data.
- Load *You'll Hear It.pd*.
- Enable audio processing (DSP on) in PD.
- Click 'Sample Room Silence' when the room is quiet so the patch can adjust the microphone gain level automatically for the environment.

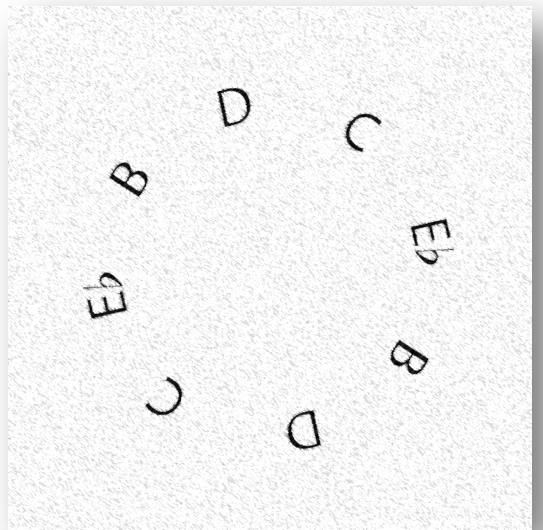
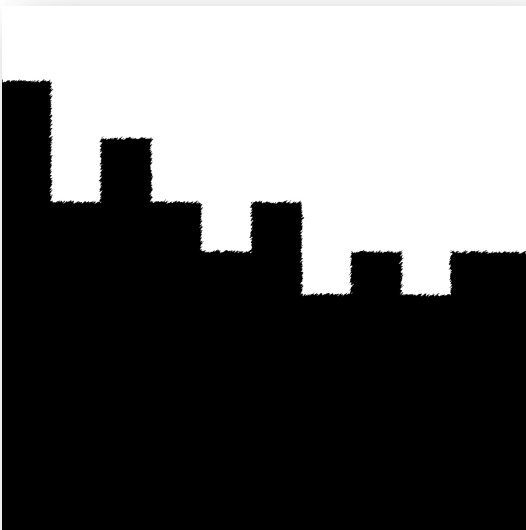
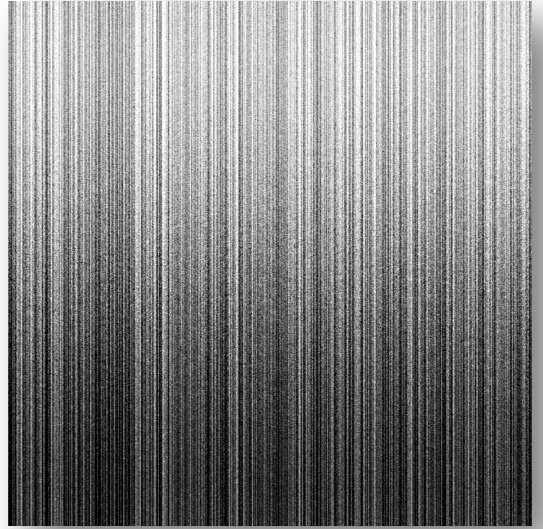
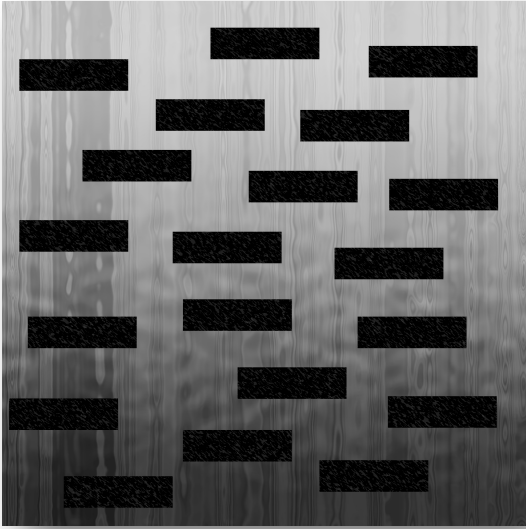
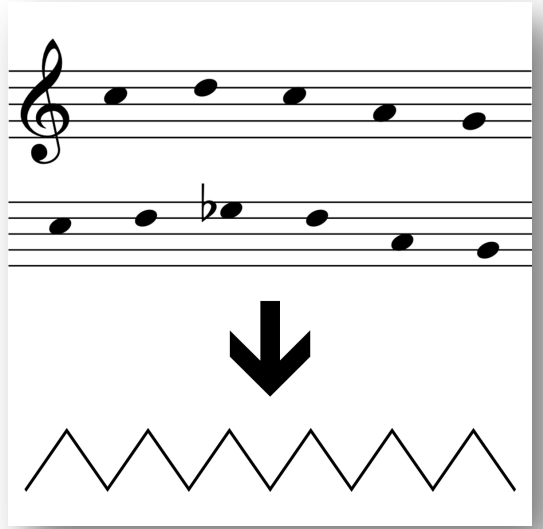
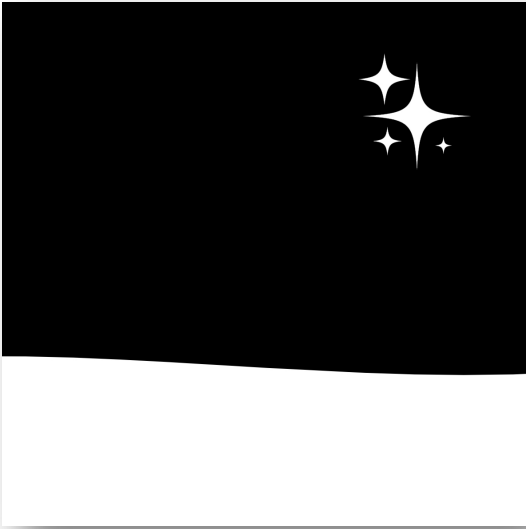
### Performing The Piece

- Without looking at the graphic score cards, the performer should randomly choose three cards immediately before entering the performance. Each card represents a movement of the piece. The performer should look at the cards immediately before performing each movement and should interpret the cards however they wish.
- The piece will have three movements, with movement length being at the performer's discretion.
- The performer should use the footswitch to start and stop the performance.

## How the Patch Works

*You'll Hear It* is designed to respond to any monophonic, pitched improvisation. As you start playing, the patch will record an incredibly short and accurate sample of the timbral properties contained within that note. This forms the basis for the sound of the patch for when it starts to respond to the live instrument. Many other properties such as the note's attack and volume evolution also contribute to the patch's response. After around five notes are played, the patch will begin to improvise with you. It will create a profile of the way in which you are playing that will analyse note choices, melodic shape, phrase characteristics and articulation. After around thirty notes, the patch will have confidently established *how* you are playing and respond more creatively. If you abruptly change characteristics of your playing, the patch will react to this and respond accordingly. The patch will never repeat exactly what you play, but will develop and combine its musical choices with yours appropriately. If the patch responds to you and you respond to the patch, a convincing improvisation will be created that neither two performers could have predicted.





## Appendix A: Patch Notes

### Interface (overleaf)

The interface of this patch is designed to provide intuitive telemetry to the engineer during the setup and performance of this piece. The three buttons to start the performance (1) are accompanied by four feedback and control sections. The 'Sample Room Silence' button takes an ambient volume level that is interpreted as a 'zero point' with regard to detecting when the performer is playing and automatically adjusts the microphone gain within the patch.

The 'Pitch Detection' section features fine-tuning controls to accommodate different acoustics (2). For example, the 'Initial Delay' (3) adjustment delays the 'wait time' before a new pitch is read. Raising this can instruct the patch to ignore the attack of the note, thereby ensure a more accurate timbral profile of the instrument is generated. However, if the 'Initial Delay' variable is too high the pitch may be missed completely when the performer plays shorter notes. A larger 'Recording Envelope' (4) instructs the patch to take more time 'listening' to the pitch and therefore develop a more robust timbral profile. This can be useful for venues with a reverberant acoustic. The 'Recording Envelope' sliders (5) instruct the patch to analyse only the samples within the two thresholds. For example, in the screenshot used here only the samples numbered 1,2,3,4,5 and 6 will be analysed. Samples 0, 7, 8 and 9 will not be analysed in creating the timbral profile of the instrument. In practice, only analysing a selection of the of the ten spectral samples has generated a much more accurate timbral profile, depending on instrument and room acoustic. These controls are designed to be adjusted by the engineer prior to the performance to ensure the patch 'listens' correctly throughout the piece. The 'Pitch Detection' controls are sparse as the vast majority of adjustment for detecting accurate and precise pitches is automatically governed by the patch.

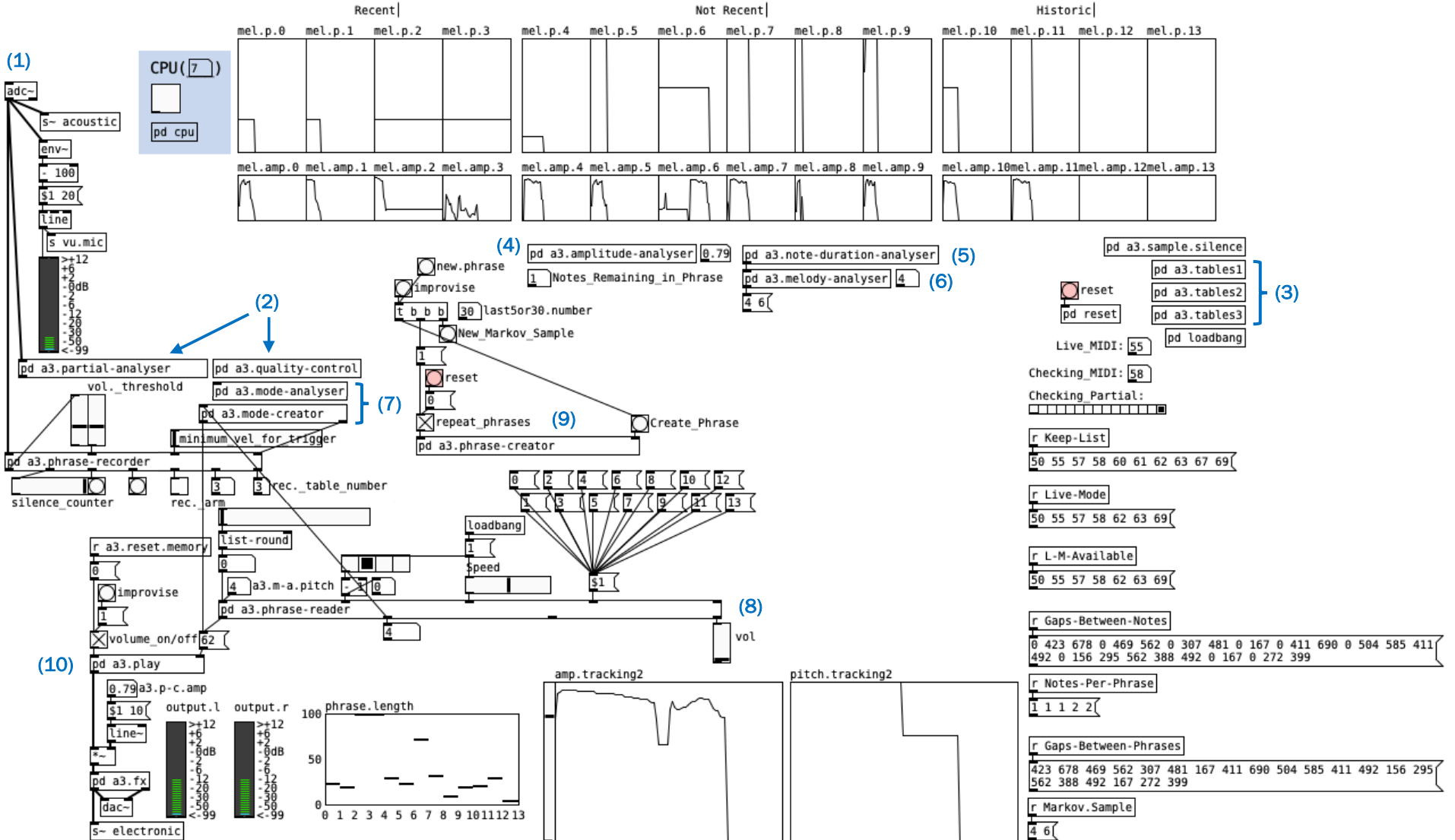
The 'Diagnostic' section allows the engineer to understand whether the patch is 'listening' correctly and enable them to make informed changes within the 'Pitch Detection' controls. The first process (6) highlights a small white box from left to right that indicates which of the sixteen partials is currently being checked by the Quality Control system (**pd a3.quality-control**). The following three numbers (7) display the MIDI value of the pitch that is being played by the performer, the MIDI value of the pitch currently being checked, and the quality of the pitch that is currently being checked. Through extensive testing, it was found that a score of just five out of the sixteen partials passing the quality control stage created a convincing replication of the live instrument's pitch and timbral properties. The 'Keep List' (8) is a list of the MIDI values that have passed quality control and will therefore be added to the library of sounds that constitute the virtual instrument. The 'Live Mode' (9) is a MIDI list of the seven most recent pitches the performer has played. Every note the patch generates will take its pitch from the 'Live Mode' list. The patch may have identified the most recent seven pitches but not have successfully recorded all of them. The 'L-M-Available' (Live Mode Available) list (10) represents the 'Live Mode' list compared with the 'Keep List' and contains the MIDI values that are present in both lists. In the screenshot the 'L-M-Available' list is made of the pitches 55, 57, 60, 62 and 67 (G3, A3, C4, D4, G4) which means the patch will use these pitches for improvisation until it manages to successfully record and therefore add the remaining pitches from the live mode.

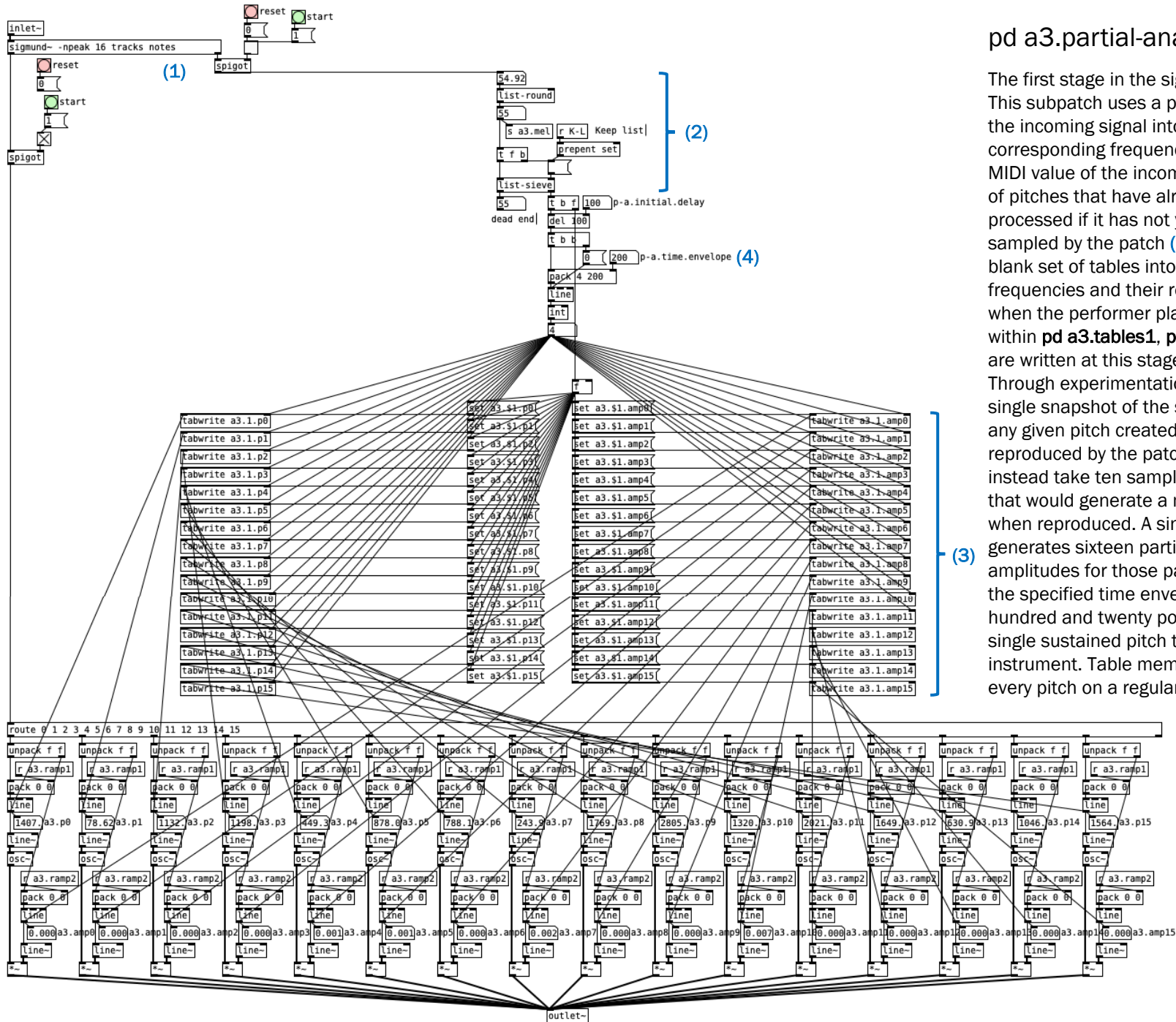
The 'Phrase Generation' section provides graphical representation of the patch's improvisation of volume and pitch over a phrase. Both of these variables are based on what the performer has played but it would be incredibly unlikely to hear a phrase the performer has played repeated back by the patch. The videos of the interface in action that accompany this commentary provide an insight into how the data displayed on this screen is communicated to the engineer.



# pd a3

This subpatch forms a storage area for all of the interconnected systems within *You'll Hear It*. The signal chain starts with the microphone input (1) which is immediately analysed (2) and stored (3). The partial analyser is responsible for creating the sound of the live electronics based on the timbre of the performer's instrument. The amplitude (4), rhythm (5) and pitch (6) of the performer's improvisation are then tracked and analysed. Modal (7) and phrase-focused (8) analysis is carried out, within which the patch looks for trends to inform its own improvisation (9). Finally, the patch's improvisation is rendered using the timbral information generated from the start of the signal chain (10).





## pd a3.partial-analyser

The first stage in the signal chain is the partial analyser. This subpatch uses a pitch tracking system to separate the incoming signal into sixteen partials; each with their corresponding frequencies and amplitudes (1). The MIDI value of the incoming pitch is compared with a list of pitches that have already been analysed and only processed if it has not yet been ‘heard’ and successfully sampled by the patch (2). Every MIDI value has its own blank set of tables into which the lowest sixteen partial frequencies and their respective amplitudes are written when the performer plays a note. These tables, stored within `pd a3.tables1`, `pd a3.tables2` and `pd a3.tables3`, are written at this stage and analysed afterwards (3). Through experimentation, it was discovered that a single snapshot of the spectral characteristics within any given pitch created a flat sounding tone when reproduced by the patch. I decided the patch would instead take ten samples within a time envelope (4) that would generate a more natural sounding tone when reproduced. A single note played by the performer generates sixteen partial frequencies and sixteen amplitudes for those partials all recorded ten times over the specified time envelope. This generates three hundred and twenty points of data used to reproduce a single sustained pitch that mimics the performer’s instrument. Table memory is able to accommodate every pitch on a regular-sized piano.





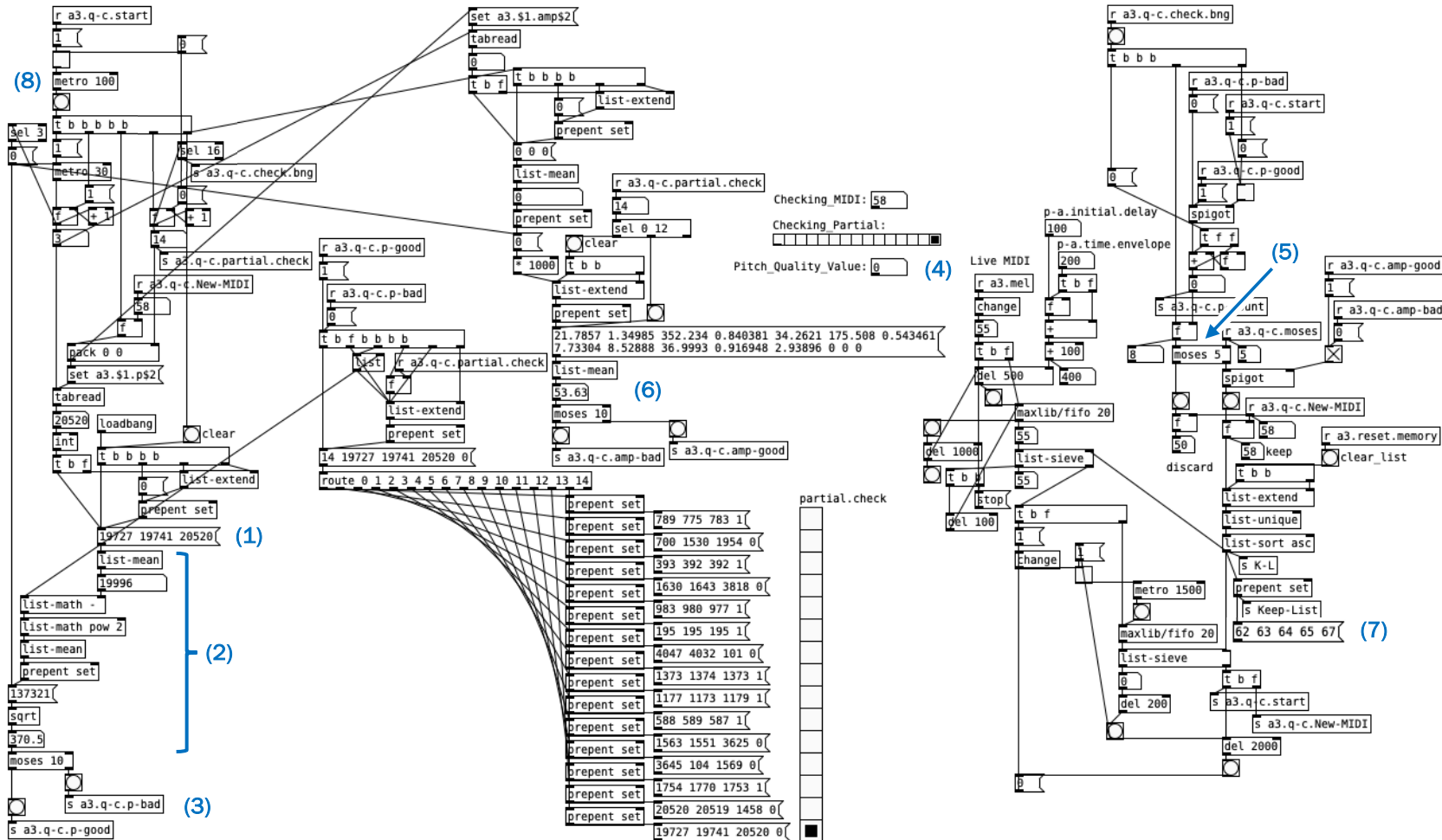






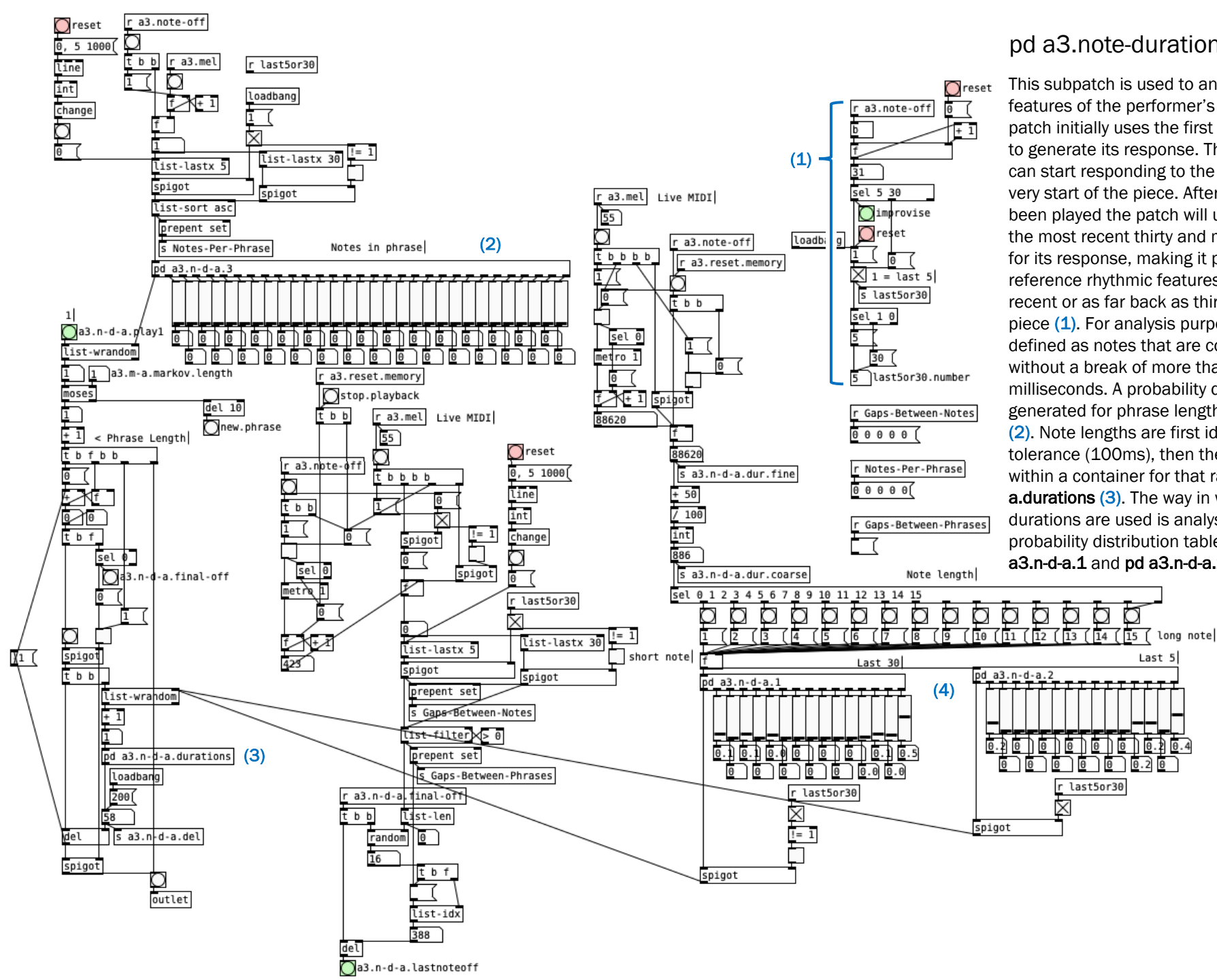
# pd a3.quality-control

After a pitch has been split into sixteen partials and stored in the appropriate tables, a quality check is carried out to ensure the pitch was recorded accurately and will sound like the original note when recreated by the patch. This subpatch checks the quality of the recorded note by checking the consistency of the frequency values for each partial and the strength of their corresponding amplitudes. While ten samples of each partial's frequency and amplitude are taken, it was discovered in testing that if the first three samples of a partial's frequency (1) were 'good' then the remaining seven always followed suit. For a single partial, the standard deviation is calculated for the first three sampled frequencies (2). If the standard deviation is below 10 the partial being checked is assigned a 'good' value of 1. If the standard deviation is 10 or above it is assigned a 'bad' value of 0 (3). The sum of the 'good' and 'bad' values is labelled the 'Pitch Quality Value' (4). This value has to be 5 or higher for the pitch to pass and be used by the patch (5). While the pitch quality is being checked, the mean amplitude of each partial is calculated and filtered in the same fashion as the frequencies. If the mean of the amplitude set is higher than 10 it is labelled 'good' (6). At this stage, should the frequency and amplitude both pass the quality control checks, the MIDI value being checked is added to the 'Keep List' which ensures the pitch is not checked again for the duration of the piece (7). Each partial takes 100ms (8) to check, resulting in a complete check of the sixteen partials within a MIDI value being carried out in 1.6 seconds. Should the pitch fail the quality check the tables containing the frequency and amplitude data are overwritten the next time the performer plays that pitch. The patch will only play pitches that have passed the quality check. A video of this subpatch has been included within the media folder for this piece.



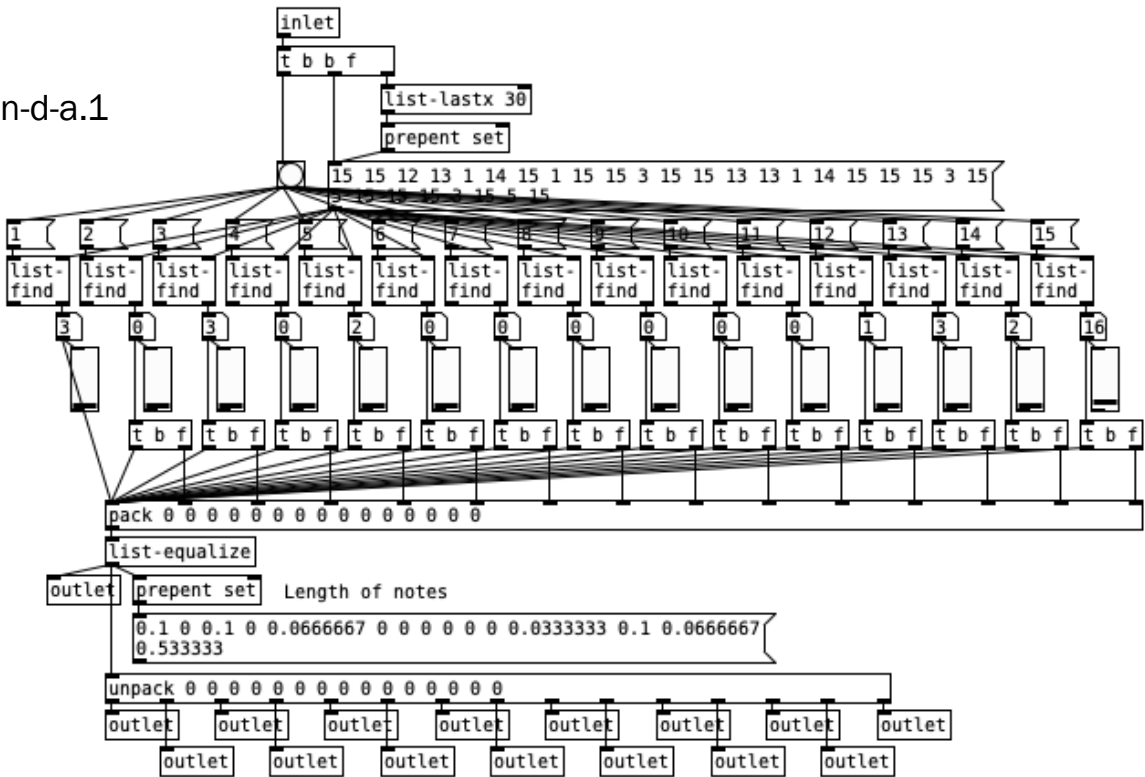
# pd a3.note-duration-analyser

This subpatch is used to analyse rhythmic features of the performer's improvisation. The patch initially uses the first five notes upon which to generate its response. This means the patch can start responding to the performer near the very start of the piece. After thirty notes have been played the patch will use a combination of the most recent thirty and most recent five notes for its response, making it possible for it to reference rhythmic features that are either very recent or as far back as thirty notes ago in the piece (1). For analysis purposes, phrases are defined as notes that are consecutively played without a break of more than a handful of milliseconds. A probability distribution table is generated for phrase length within **pd a3.n-d-a.3** (2). Note lengths are first identified within a wide tolerance (100ms), then the exact value is stored within a container for that range within **pd a3.n-d-a.durations** (3). The way in which different note durations are used is analysed and more probability distribution tables are generated in **pd a3.n-d-a.1** and **pd a3.n-d-a.2** (4).

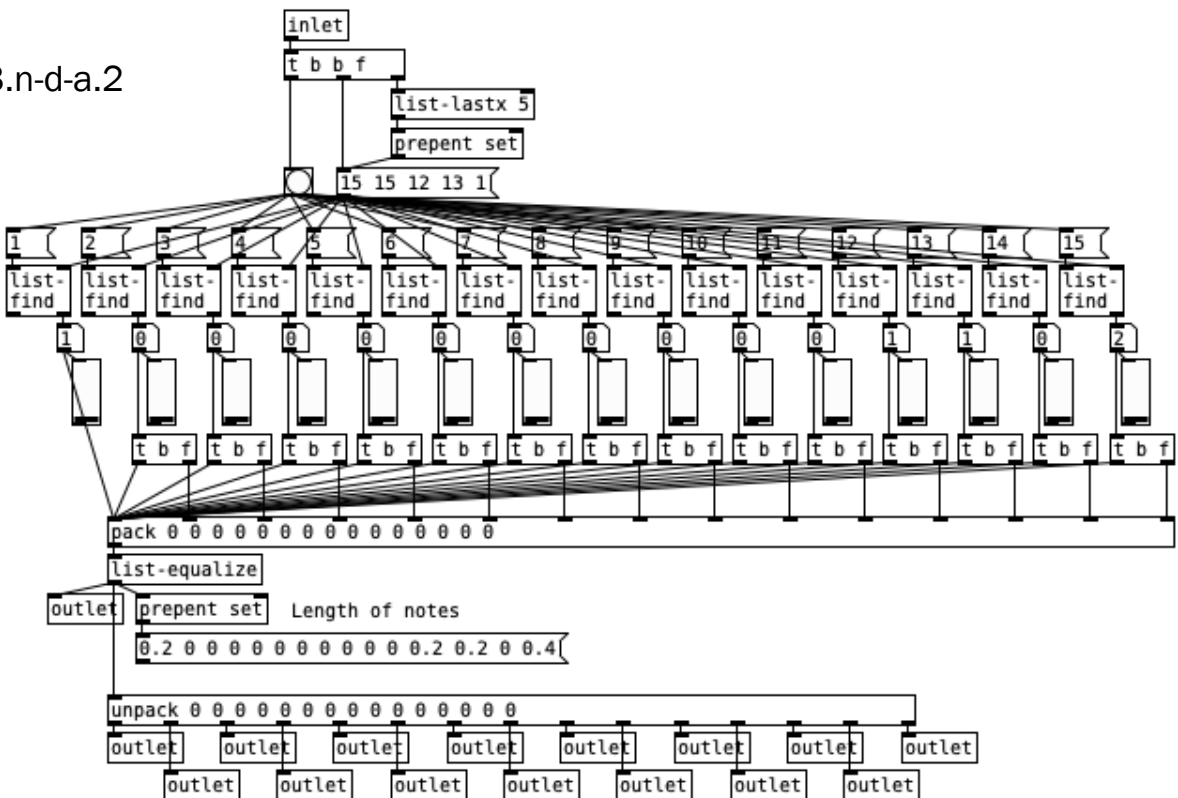


The two subpatches on this page analyse the distribution of note durations from a list of the thirty most recent and five most recent notes played by the performer, respectively.

a3.n-d-a.1

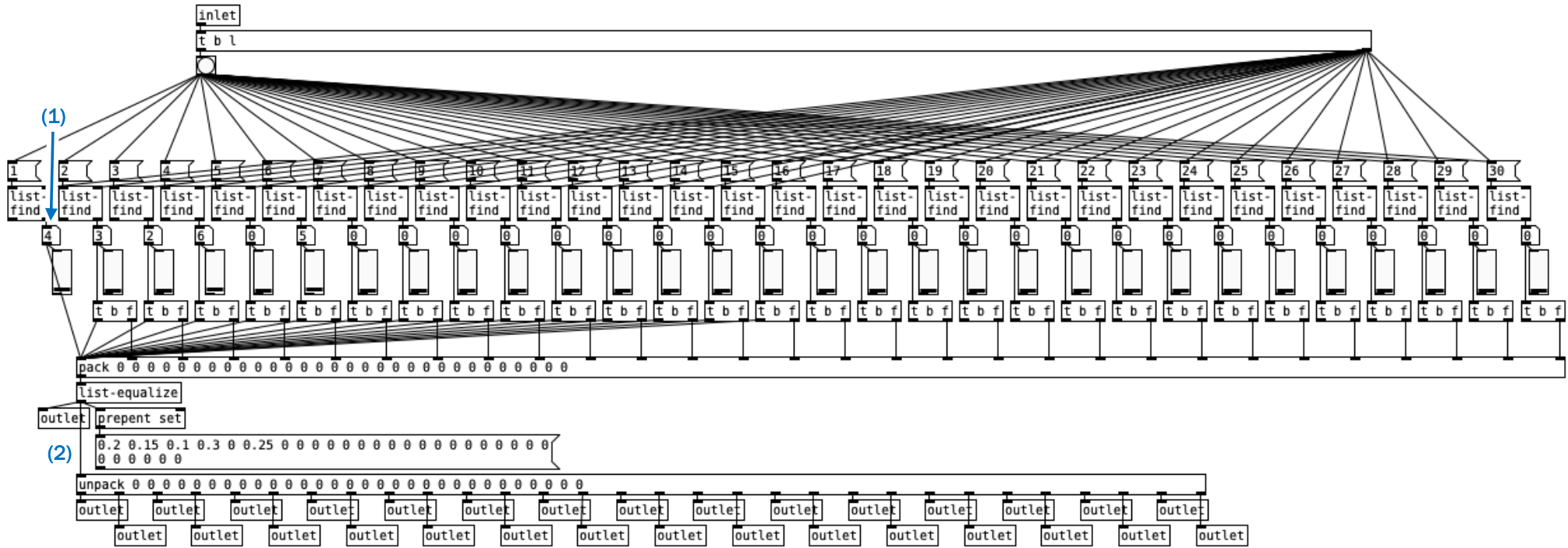


a3.n-d-a.2



### pd a3.n-d-a.3

This subpatch creates a probability distribution table for the number of notes in each phrase. In the screenshot below there have been four phrases with one note in each (1). This value is analysed along with the other notes-per-phrase to create the distribution table beneath (2).

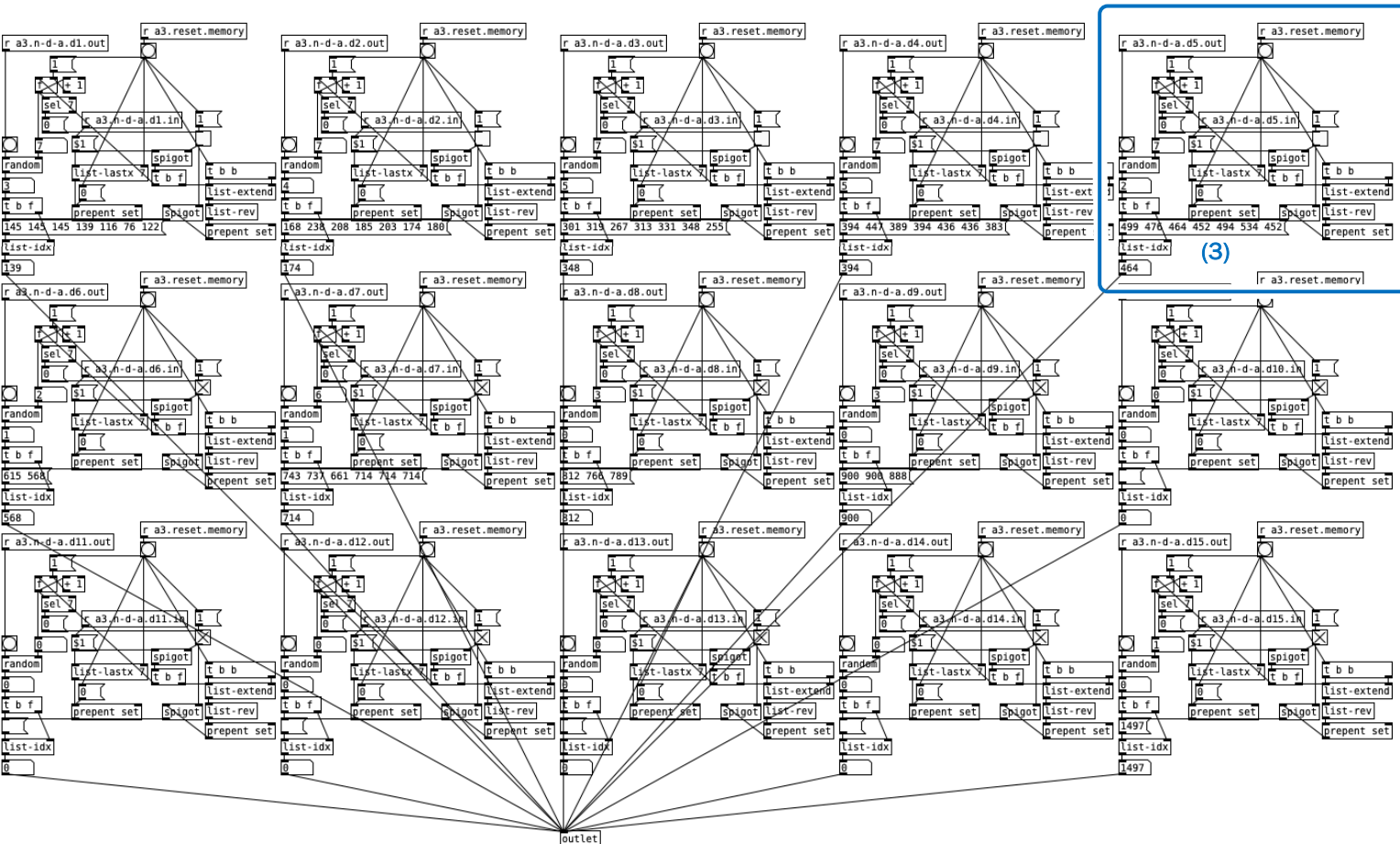
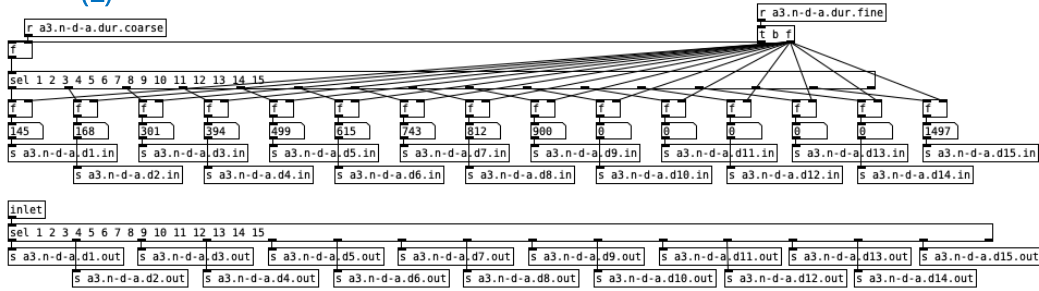


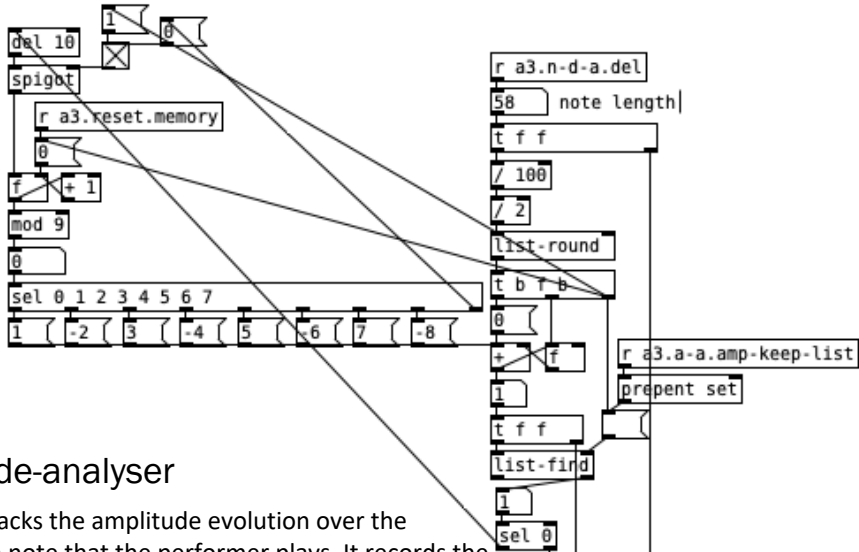


## pd a3.n-d-a.durations

This subpatch is where note lengths are recorded. Note lengths are labelled 1-15 according to their duration in milliseconds (1). This is described as a 'coarse' filter throughout the patch and is useful for calling rhythms that are similar but not exactly the same as the input. For example, if the note lasts between 450-549ms it is stored in the fifth container (2). The most recent seven values are stored in a list within this container, ready to be called by the patch should it require a note of a similar length (3). This system of storing similar note durations and looking for trends in their usage gives the patch more freedom in creating its own rhythms.

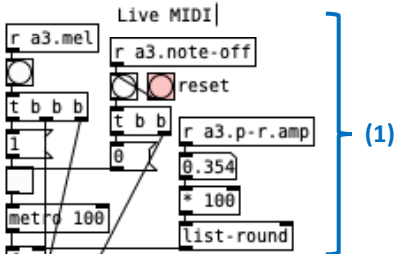
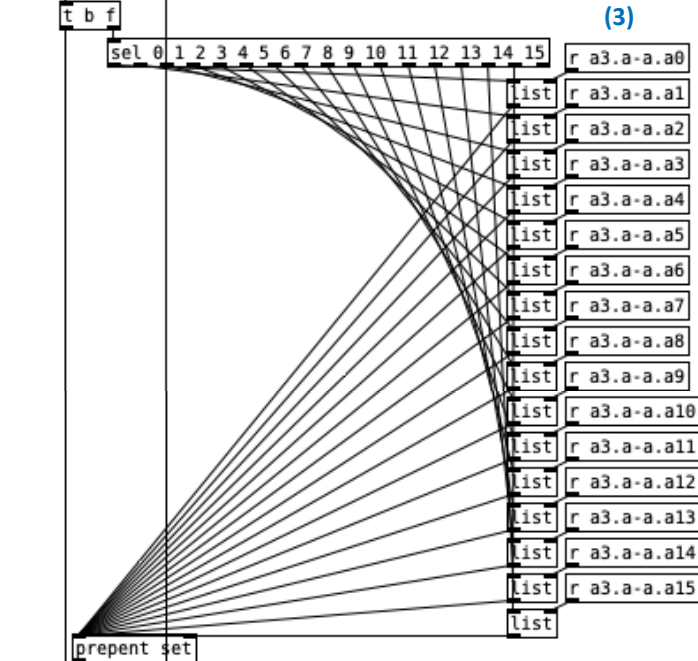
(1)



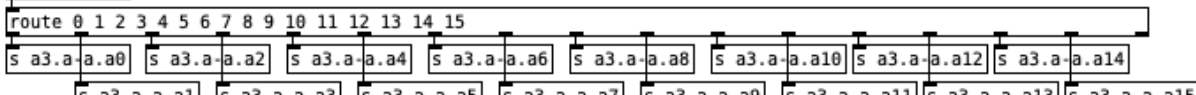
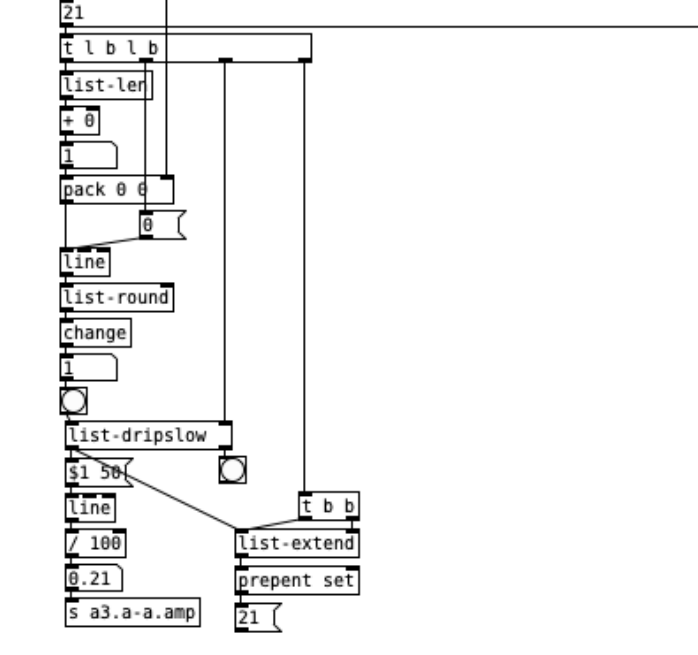
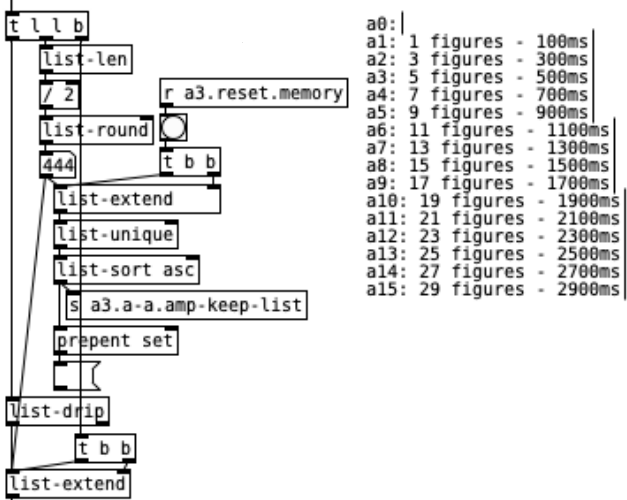


### a3.amplitude-analyser

This subpatch tracks the amplitude evolution over the duration of each note that the performer plays. It records the amplitude every 100ms and converts it into an integer between 0 and 100 (1). This value is then plotted on a list (2). In the example screenshot the values 47,51,52,59,64 and 71 reveal the performer played a crescendo throughout this length note. When the patch plays a note of a similar length, it will interpret the amplitude evolution data analysed in this section of the patch to inform its response (3). This system helps the patch mimic the unique characteristics of the performer and their instrument. For example, a note played on a piano will always get quieter after the initial attack. The patch will model this precisely and, used in combination with the other systems within the patch, the response will start to sound like a piano.

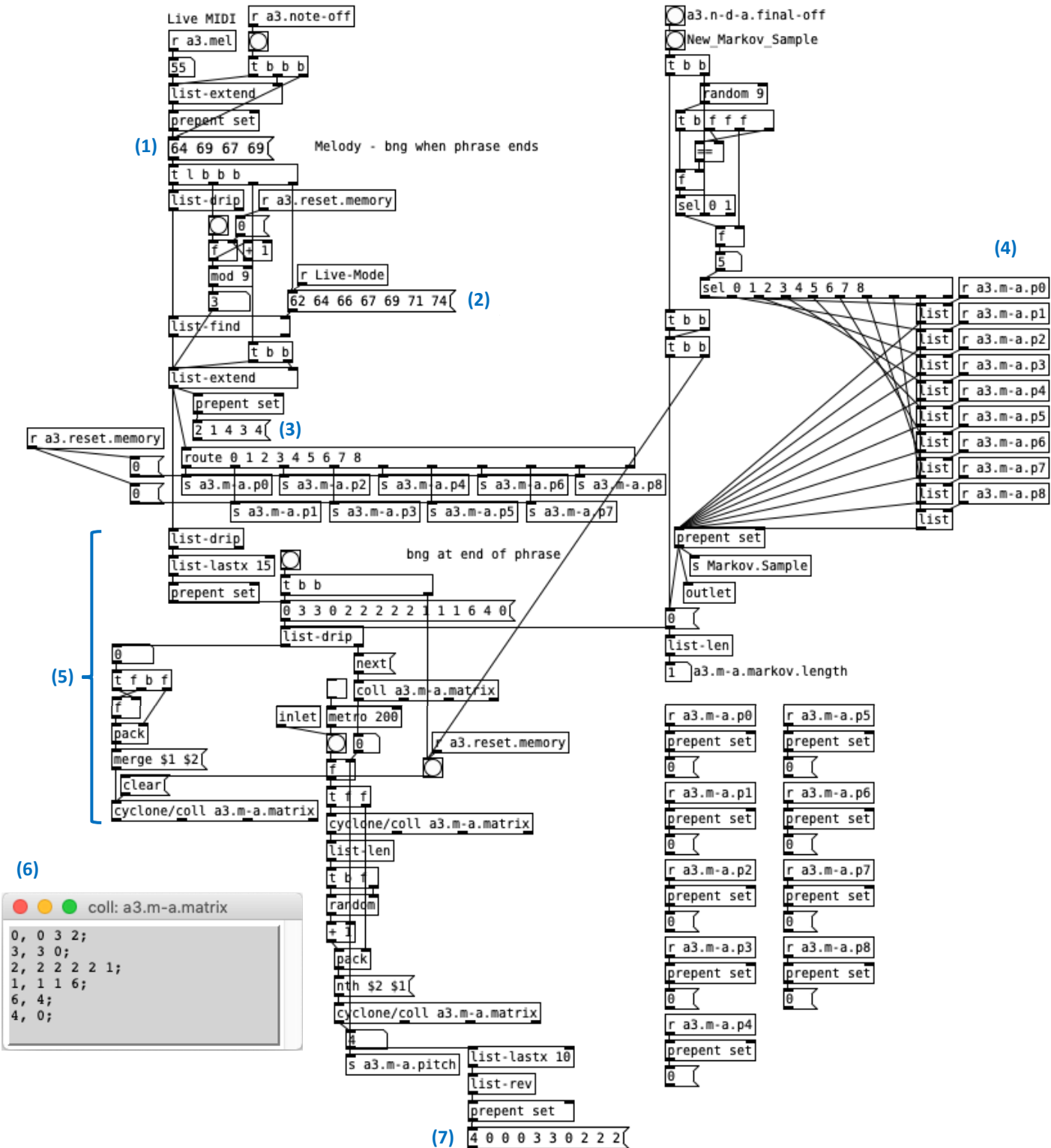


47 51 52 59 64 71 (2)



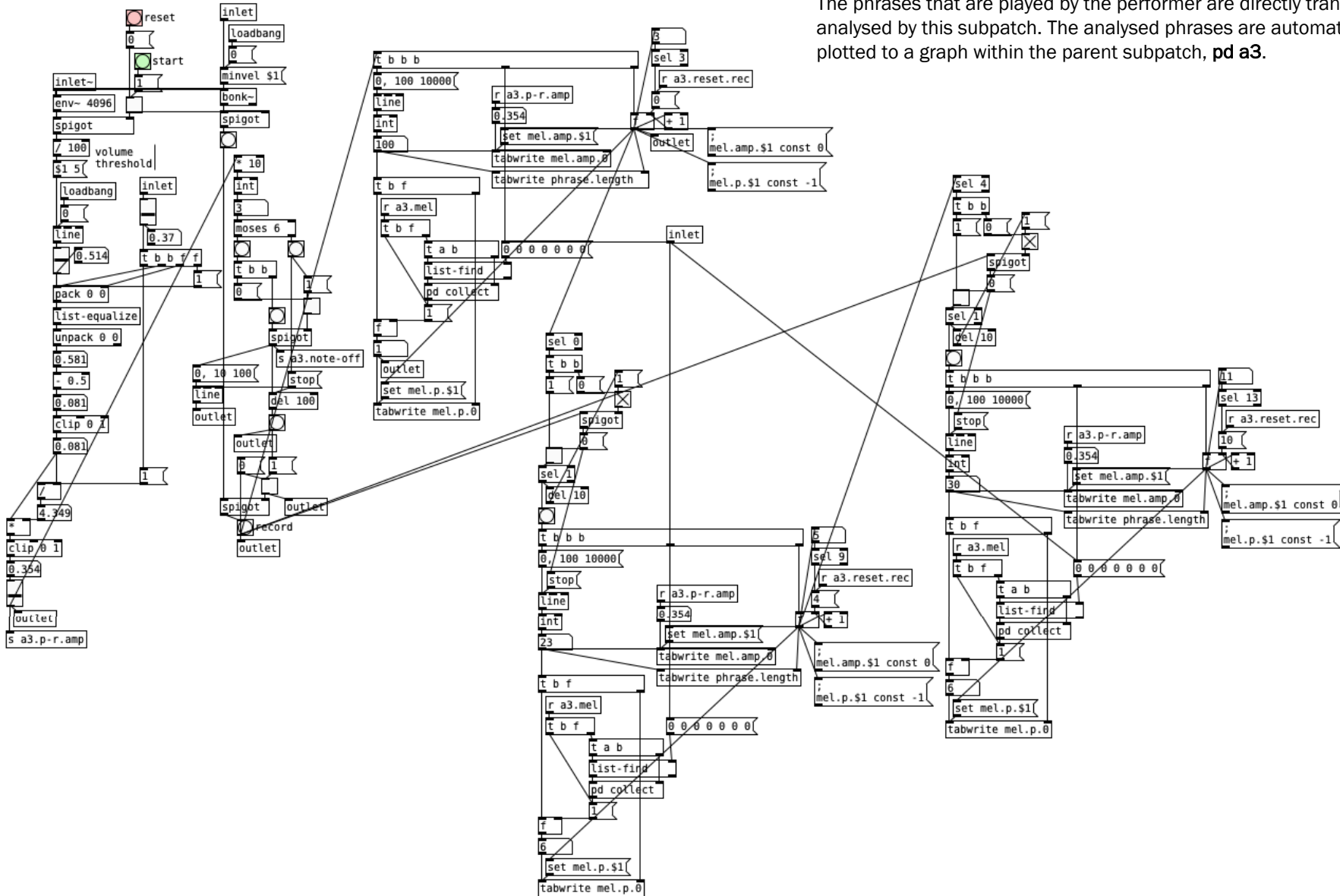
# a3.melody-analyser

The melody analyser compares the most recent phrase (1) against the live mode (the seven most recent pitches) (2) to generate a number sequence that matches the intervals within the most recent phrase (3). The first number in the sequence points the string of values to one of nine containers for analysis (4). A Markov chain analysis is applied to the most recent fifteen pitches (5) and a first order matrix generated (6). This allows the patch to create phrases that are similar to the performer's improvisation but not exactly the same (7). Even if the exact phrase pattern is generated twice, the constantly-evolving live mode means the pitches would be transposed to the new mode.

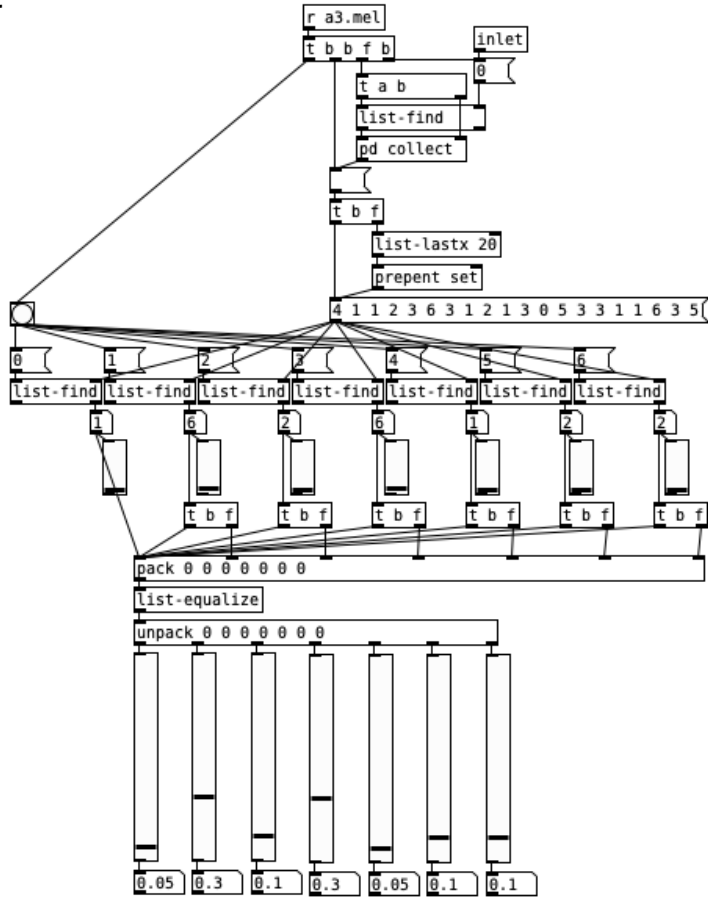


## pd a3.phrase-recorder

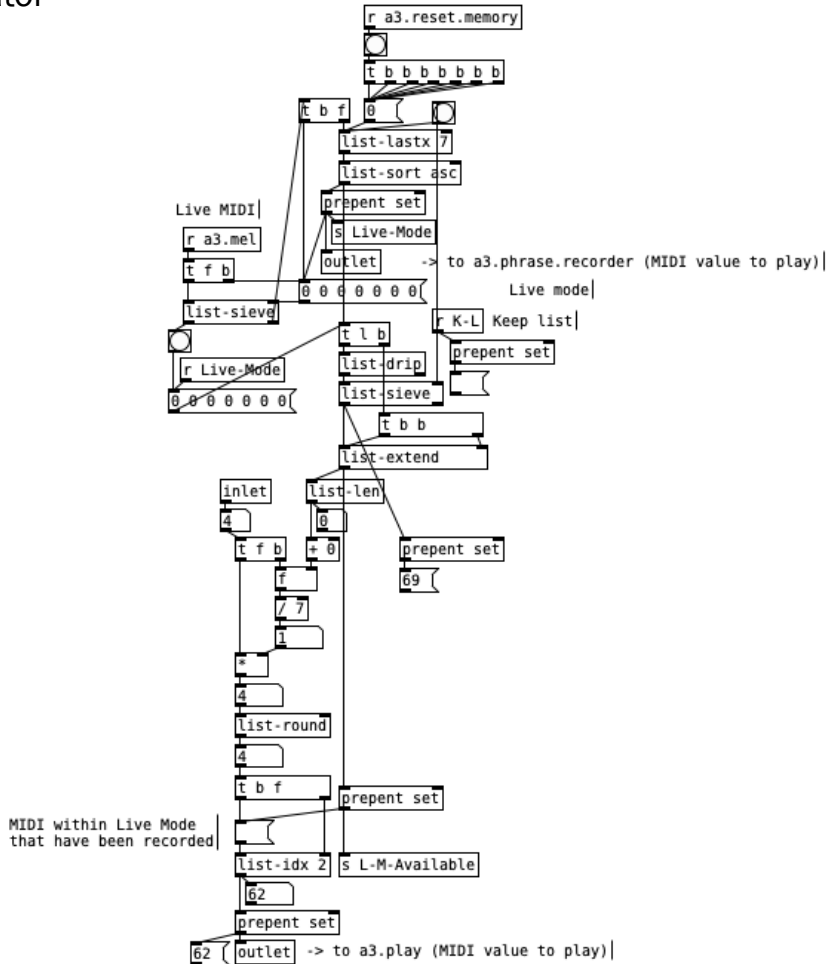
The phrases that are played by the performer are directly transcribed and analysed by this subpatch. The analysed phrases are automatically plotted to a graph within the parent subpatch, **pd a3**.



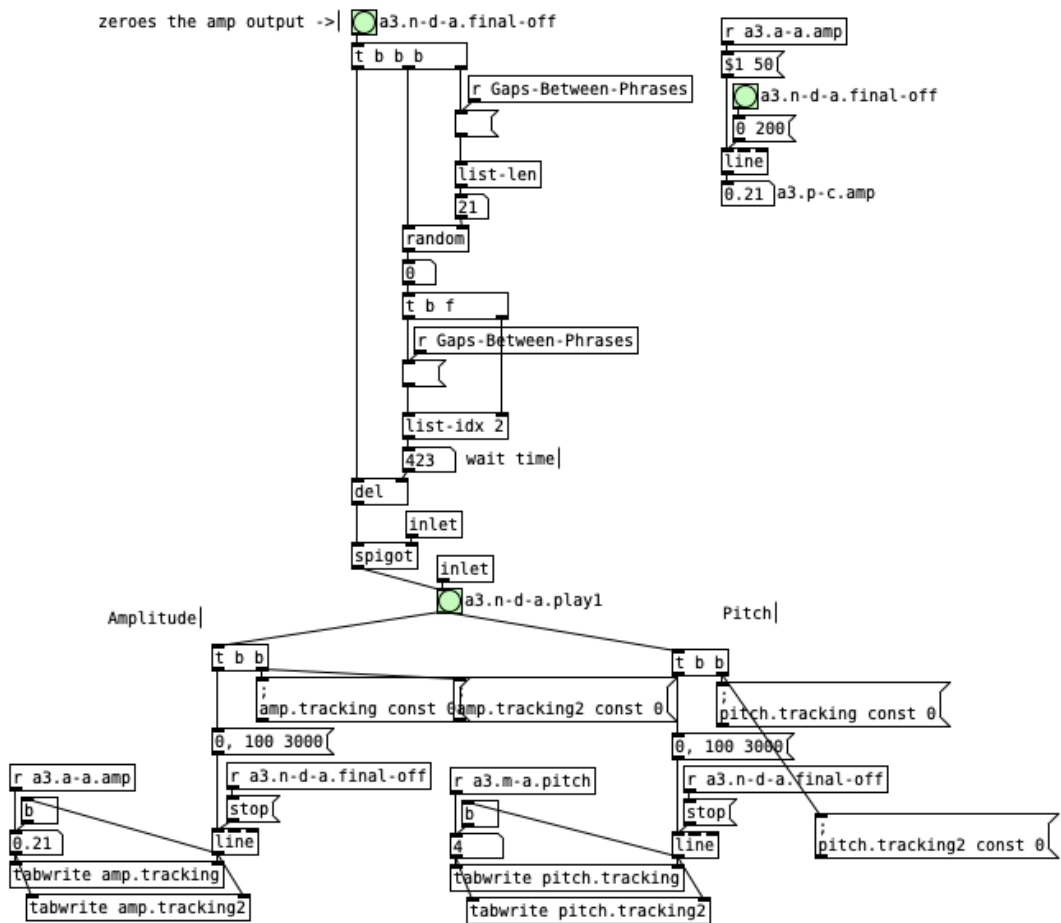
### a3.mode-analyser



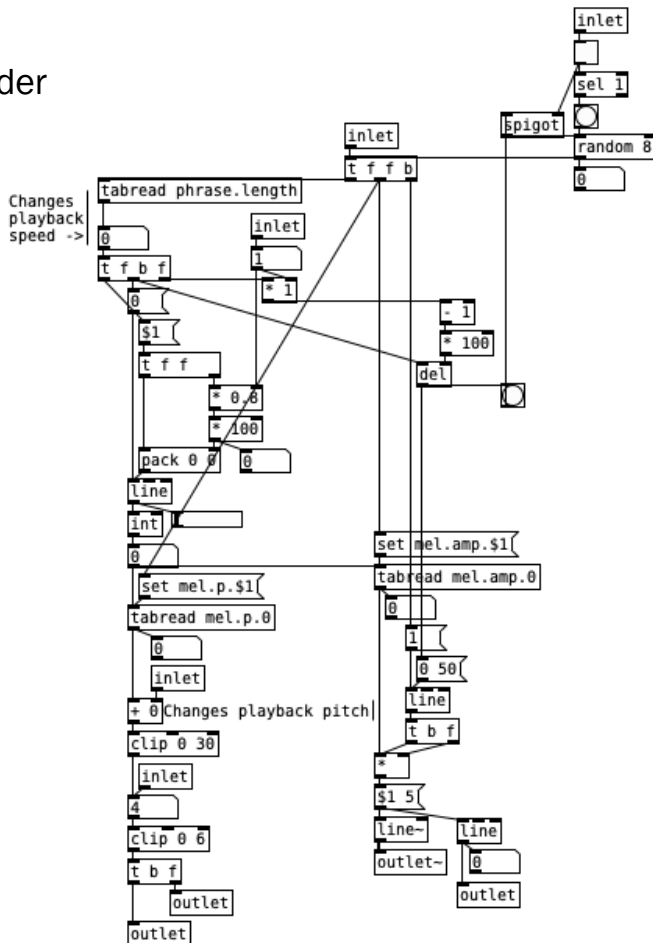
### a3.mode-creator



## a3.phrase-creator

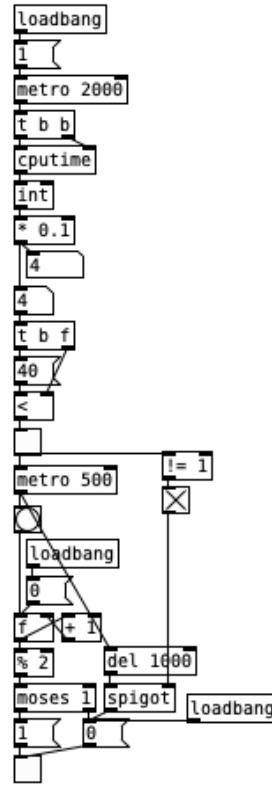
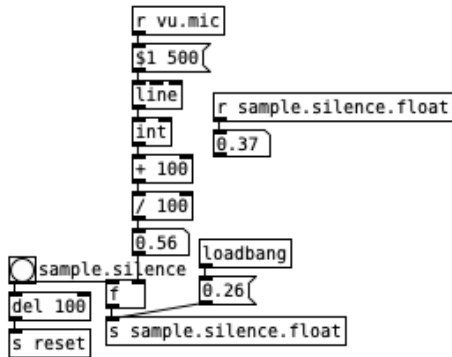


## a3.phrase-reader

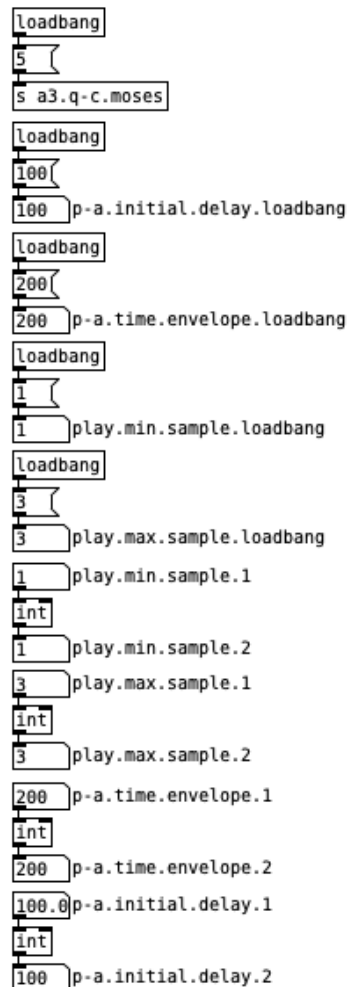
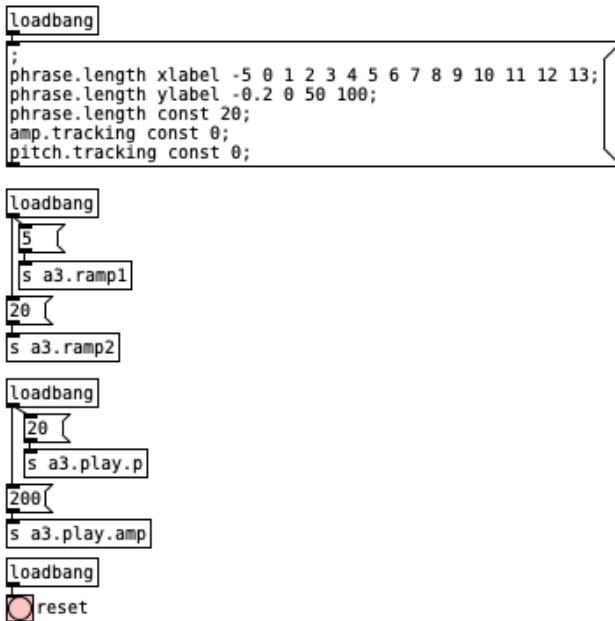


### a3.sample.silence

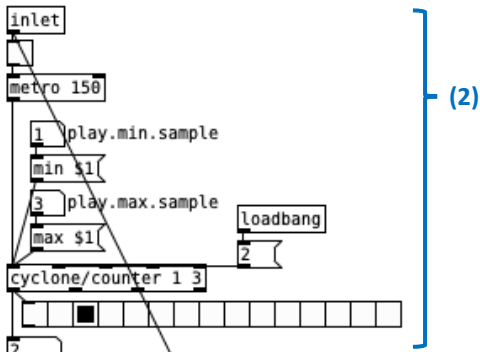
### a3.cpu



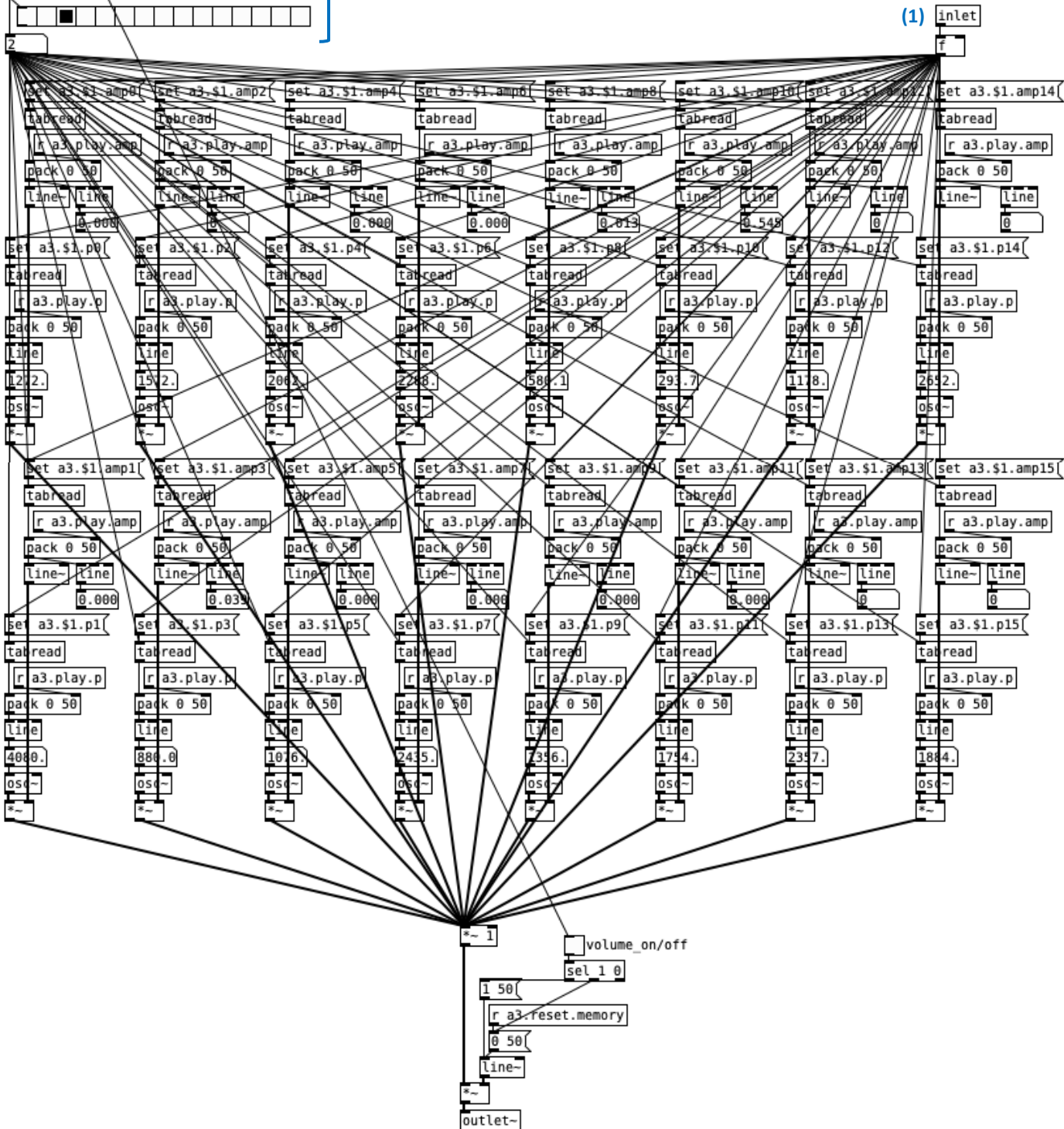
### a3.loadbang



# a3.play



This subpatch is responsible for rendering the timbre of the live electronics. The sixteen partials and their corresponding amplitude strengths sampled in **pd a3.partial-analyser** and stored within **pd a3.tables1-3** are recalled at this stage to recreate the sound of the live instrument. The MIDI value (and therefore the set tables) to be called connects to the right inlet (1). The position within each individual table advances one point every 150ms, generating a natural, fluctuating timbre similar to that of the live instrument (2).





### a3.fx

A small amount of reverb is applied to the output signal. It was discovered in testing that the patch generated a high-pitched 'click' when changing note so a lowpass filter that removes frequencies above 8kHz is also applied to the signal at this stage.

