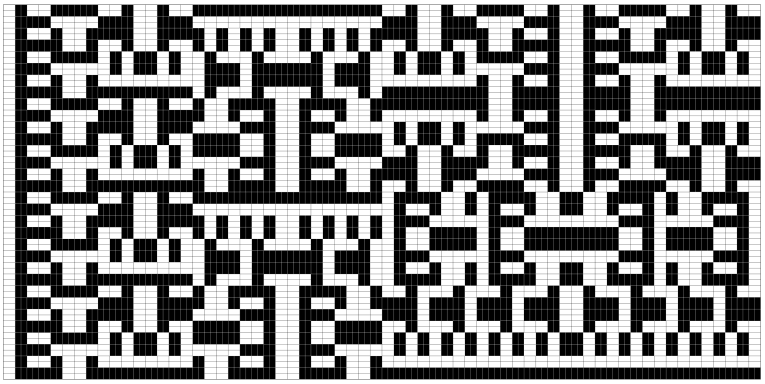
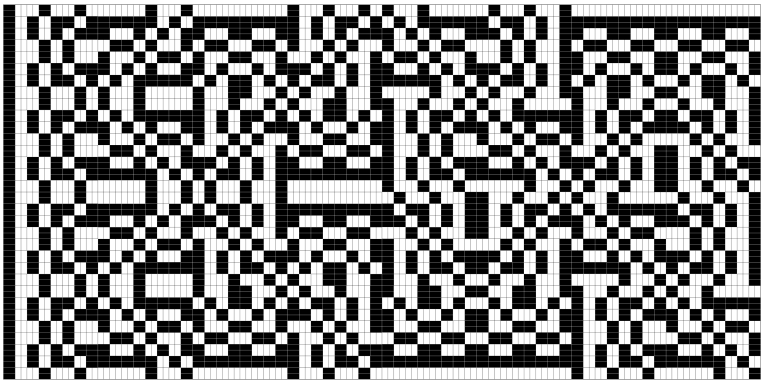
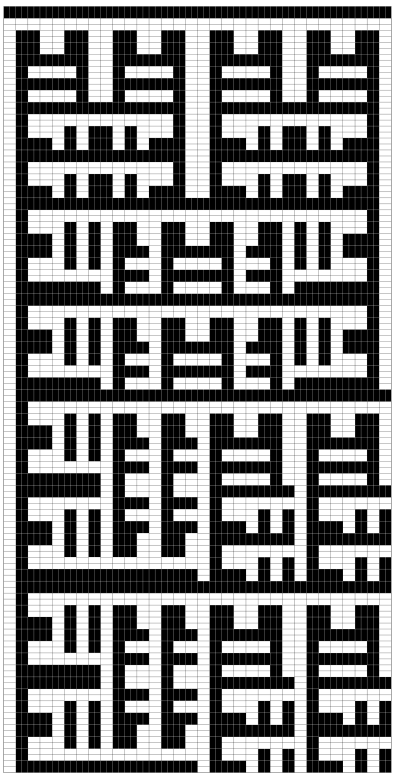
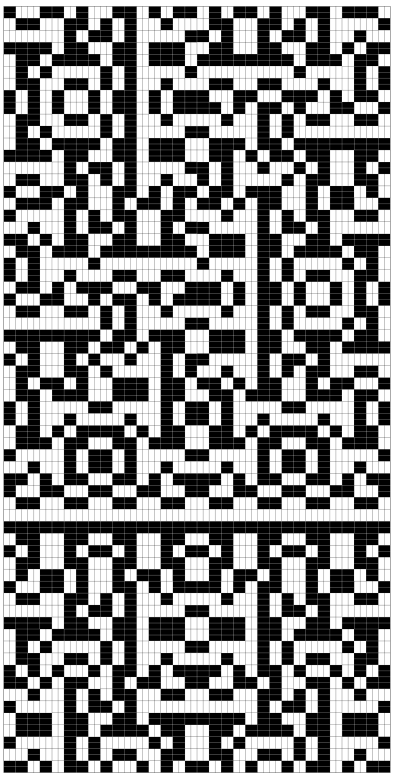
a lot of tiles (trivial scan)

4096 variations on 256 tiles dedicated to Chaim Goodman-Strauss and Omar Lopez
a performance-installation









michael winter (cdmx, mx and nashville, usa; 2018)

a lot of tiles (trivial scan)

4096 variations on 256 tiles dedicated to Chaim Goodman-Strauss and Omar Lopez
a performance-installation

michael winter (cdmx, mx and nashville, usa; 2018)

Program notes

This piece is based on a set of rectangle substitution tilings (commonly referred to as tessellations) explored by Chaim Goodman-Strauss in his seminal paper *Lots of aperiodic sets of tiles*. A rectangle substitution tiling is generated by dissecting a rectangle into 4 smaller rectangles, which are then dissected into 8 even smaller rectangles, and so on. Rectangles can produce both periodic and non-periodic tilings and thus are not strictly aperiodic. A strictly aperiodic set of tiles consists of a group of geometric shapes that submit only non-periodic tilings. However, Goodman-Strauss shows in his paper that the group of rectangle substition tilings used for this piece can be represented by 25380 aperiodic tilings using subsets of 211 non-rectangular tiles. Prior to this discovery, only a handful of strictly aperiodic tilings were known. That is, in one fell swoop, Goodman-Strauss extended the number of known aperiodic tilings by some 500+ fold.

In order to proceed with his proof of the newly discovered 25380 aperiodic tilings, Goodman-Straus outlines how to generate the 256 rectangle substitution tilings which they represent. This piece extends his method of generating these 256 rectangle tilings to create 4096 sonic and visual variations. The parenthetical in the title 'trivial scan' refers to the method of sonification. A given tiling is further dissected into squares (groups of two rectangles side-by-side) which are then scanned / read such that the orientation of the pairs of subtiles which form the squares (whether they are grouped vertically or horizontally) determines sonic parameters of the piece.

Setting

The piece consists of both a visualization and sonification of the tilings generated and projected by a program written in SuperCollider. The program has a 'continuous play' mode for an installation setting that generates random tilings from the 4096 variations in succession. Live performers can also accompany the electronic realization of a tiling as detailed below. This allows the installation to be occasionally augmented by live performances and enables any subset of the variations (preferably at least 3) to be played in a concert setting as well. For the latter, projection of the visualization is optional.

Generally, the setting should be dark and the sonification should be clear and present.

Tile variations (transform code)

Each of the 4046 variations can be represented by a transform code of 7 digits. The first 4 digits generate the 256 archetypal tilings (visualizations of these are appended to this document) by determining the orientation of each of the 4 subtiles relative to its parent tile. This is the method outlined by Goodman Straus in his paper. The 5th and 6th digits determine the orientation and mirroring of the entire tiling. As mentioned in the program note, each tiling is further dissected into squares and scanned. The read-head always scans from left to right and top to bottom. Thus, the 5th and 6th digits of the transform code ultimately result in different read sequences. The 7th digit determines the color inversion of the subtiles: the color (black / white) based on the orientation (vertical / horizontal). These are turned into numeric values for sonic parameters.

Tile sonification (the scan)

The tiling is generated up to a depth of 6 substitutions / hierarchical levels (hls). While the tile visualization is hopefully self-explanatory, the sonification warrants detailed instructions.

hl 6 (fundamental): The maximum depth of 6 substitutions of the tiling is sonified by a highly controlled tremolo on a pitched tone sounding a low g (2 octaves and a perfect fourth below middle c or a frequency of approximately 49 hertz). The scanned white and black values map to amplitudes blurred by an exponential lag (see the SuperCollider UGen 'Lag'). This means that amplitude curves are not linear but logarithmic.

The computer generates this tone with a sine wave oscillator. The sine tone may be doubled or replaced by a low reed woodwind instrument (such as a bass clarinet, baritone saxophone, bassoon, or contrabassoon) sounding the lowest g comfortably playable in the instrument's range. In the score, which is engraved using the Lilypond typesetting language, a gray-scale curved gradient shows the amplitude of the tremolo such that completely black denotes the maximum volume and completely white denotes silence. Below the gradient is a rhythmic notation where the duration of each note corresponds to the amount of time before the curve changes direction (from louder to softer or softer to louder). The tempo for the quarter note strictly equals 120 beats per minute making each variation last just over 4 minutes and 15 seconds. Below each note is a numeric value indicating the target amplitude from the preceding note between: between 0 (silent) and 9 (maximum volume). For longer note durations, the curve will potentially reach the loudest or softest level before the duration indicated by the note (that is, before the volume changes direction again). The very first note of each tiling may indicate a silence if the upper left corner is a white tile which results in a starting amplitude of of 0. If the upper left corner of the tile is black, this part starts at full scale / volume immediately with no fade in. However, each variation ends with a fade out some time before the end of the final measure. Performers should use the electronic tone from the application as a reference for the dynamic profile of the tremolo.

In order to breath, the performer may fade to silence on any note over which a decrease in amplitude occurs and then reenter fading in on any note in which an increase in amplitude occurs.

This should be the most present sonic element of the piece; loud and clear.

hl 6 (harmonics): The fundamental is colored by a set of 3 harmonics (the 5th, 9th, and 13th) that sonify adjacent rows of the tiling. In this sense, the piece contains a sort of a canon. The fundamental sonifies the tiling starting on the 1st row, the 5th harmonic sonifies the tiling starting on the 2nd row, and so on. While the blurring method is the same, the maximum amplitudes of each harmonic decrease directly in correspondence with the harmonic number (1 divided by the harmonic number squared). By default these parts are not included in the score as they are not played by live performers. However, there are lines in the Lilypond file that can be uncommented to view the parts. (Note that the amplitude gradient for these parts is normalized and does not represent the decreasing amplitudes of the harmonics.)

The dynamic of this part should be well below the fundamental (see more in the explanation of the SuperCollider interface). That is, the correspondence between the harmonic number and the decrease in amplitude is maintained between the harmonics, but not in relations to the fundamental, which should sound much louder. Though it is highly preferred to play the piece with the synthesized sounds, a strictly acoustic realization could be played with the harmonics omitted.

hl 5 and 4 (high noise and low noise, respectively): High and low noises sonify the 5th and 4th substitutions / hls, respectively. The SuperCollider application synthesizes these with filtered white noise: using a high pass filter with a cutoff of 5000 hertz for the high noise and a low pass filter with a cutoff of 300 hertz for the low noise. Unlike hl 6, the black / white values are not blurred. A white value results in a soft noise and a black value results in a slightly louder noise. That is, there is no smooth transition, the volume change is abrupt and binary.

Like the fundamental of hl 6, these parts can be doubled / replaced by acoustic instruments such as percussion. Pitched instruments can also be used as long as the resulting noise does not have a clearly defined pitch. In the score, the notes of these parts are accompanied by a numeric value, 0 or 1, denoting the two states (soft or slightly louder, respectively) for the duration of that note. Like the electronics, the change should be abrupt; highlighting the binary nature of the underlying structure. Performers are free to explore different methods of sonifying the two states by different types of sounds (not just considering the dynamic parameter).

These parts should be considered as secondary to the fundamental of hl 6 and any decision moving away from the synthesized version of high and low noises should maintain the subtlety of these parts in relation to the fundamental of hl 6.

SuperCollider program

The application contains a graphical user interface (gui) and a window for the visualization of the tiling. The gui has three tabbed panes shown on the following page: a 'model' pane, a 'transport' pane, and a 'mixer' pane. These are explained in more detail below. To launch the application, execute `supercollider/a_lot_of_tiles_trivial_scan_main.scd` in SuperCollider after booting the server (on Linux, press ctlr+enter with the cursor anywhere within the code block to execute the code).

The source code for the application is appended at the end of this score and can also be downloaded from a git repository at `https://www.github.com/mwinter80/a_lot_of_tiles_trivial_scan`. The generation of this document (using LaTex) contains a version date in order to help track changes and the git repository will also detail commit changes. The piece was written using SuperCollider version 3.9.0 and Lilypond version 2.19.81.

Model pane: This pane allows the user to manipulate the transform code (which automatically updates the tile visualization), control the layout of the tile visualization, and provides display options for the tile visualization window. There are three additional buttons on the bottom right to start a 'continuous play' mode for an installation settings (note that this will disable the transform code buttons), to generate the Lilypond files for the current tiling / transform, and to export an image of the current tiling / transform.

There are four layout options: one for each of the hls and another view that embeds the hls together. The display options allow the user to enter into fullscreen mode (while the escape key will exit fullscreen mode). Deselecting the 'window decorator' button will remove the window border from the tile visualization window. This can be used to launch several instances of the application and project more than one tile visualization using the same projector.

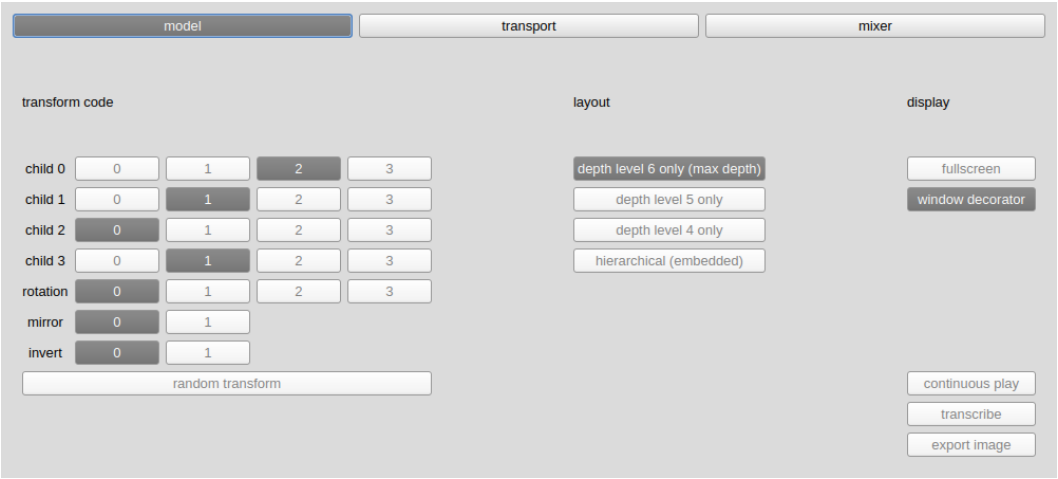
The 'transcribe' button will generate all the Lilypond files for the master score and the parts in the `score/transform_<code>_score/lilypond/` folder (where `<code>` is the digit sequence of the transform code; for example, `score/transform_2101000_score/lilypond/`). It will also attempt to run a Lilypond executable if it is installed in order to automatically create pdfs of the score in the folder `score/transform_<code>_score/pdf/`. Scores of 4 of the variations are included in this document.

Transport pane: This pane has playback controls for the current tiling / transform allowing the user to start from different locations. When the 'play' button is pressed, the application will check if the necessary audio files exist. If not, it will generate them in the `audio/transform_<code>_audio/` folder (this will delay the start a small amount of time). The generated audio files can be used for playback in an alternative environment such as a digital audio workstation. Note that if the application is in 'continuous play' mode, the audio files will be deleted as to not consume too much storage space.

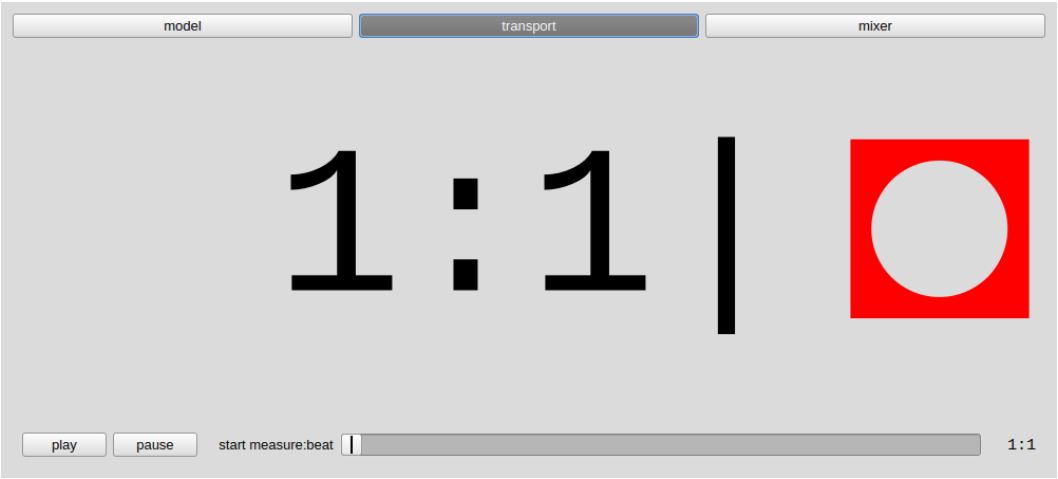
The pane also contains a visual metronome which displays the current measure and beat with respect to the score. When the 'play' button is pressed, the metronome will give two measures of silence before starting.

Mixer pane: This pane provides controls for playback levels of the different sonic elements allowing the user to ensure that the fundamental of hl 6 is the most present sonic element.

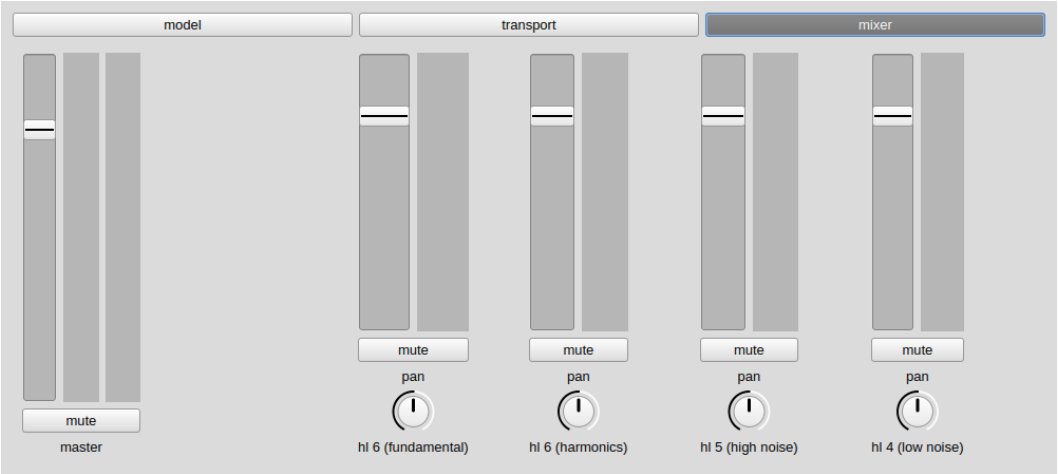
I would like to extend a special thanks to James Harkin and Thomas Morley for their generous help with SuperCollider and Lilypond, respectively.



model pane



transport pane



mixer pane

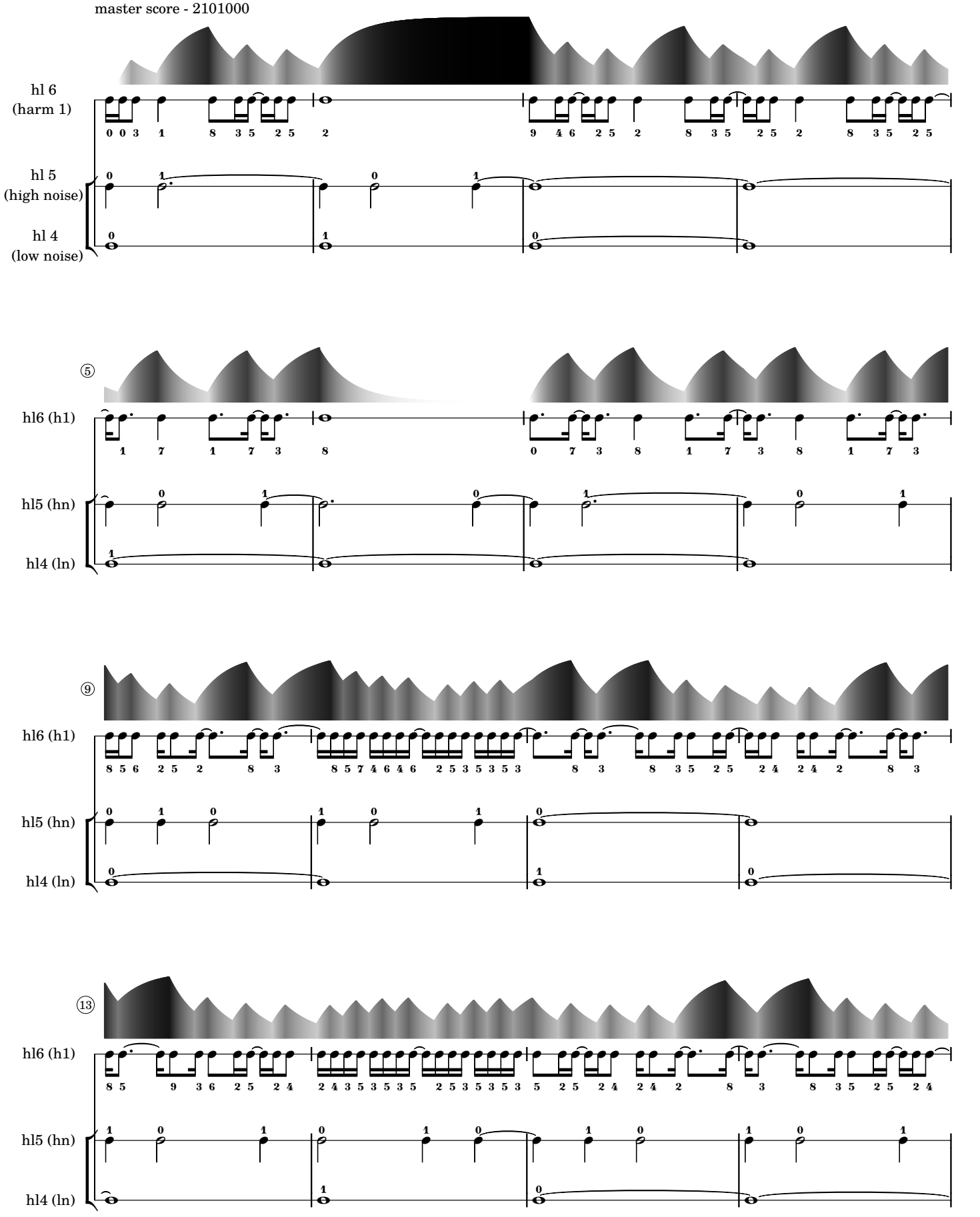
version generated: 2018.03.26

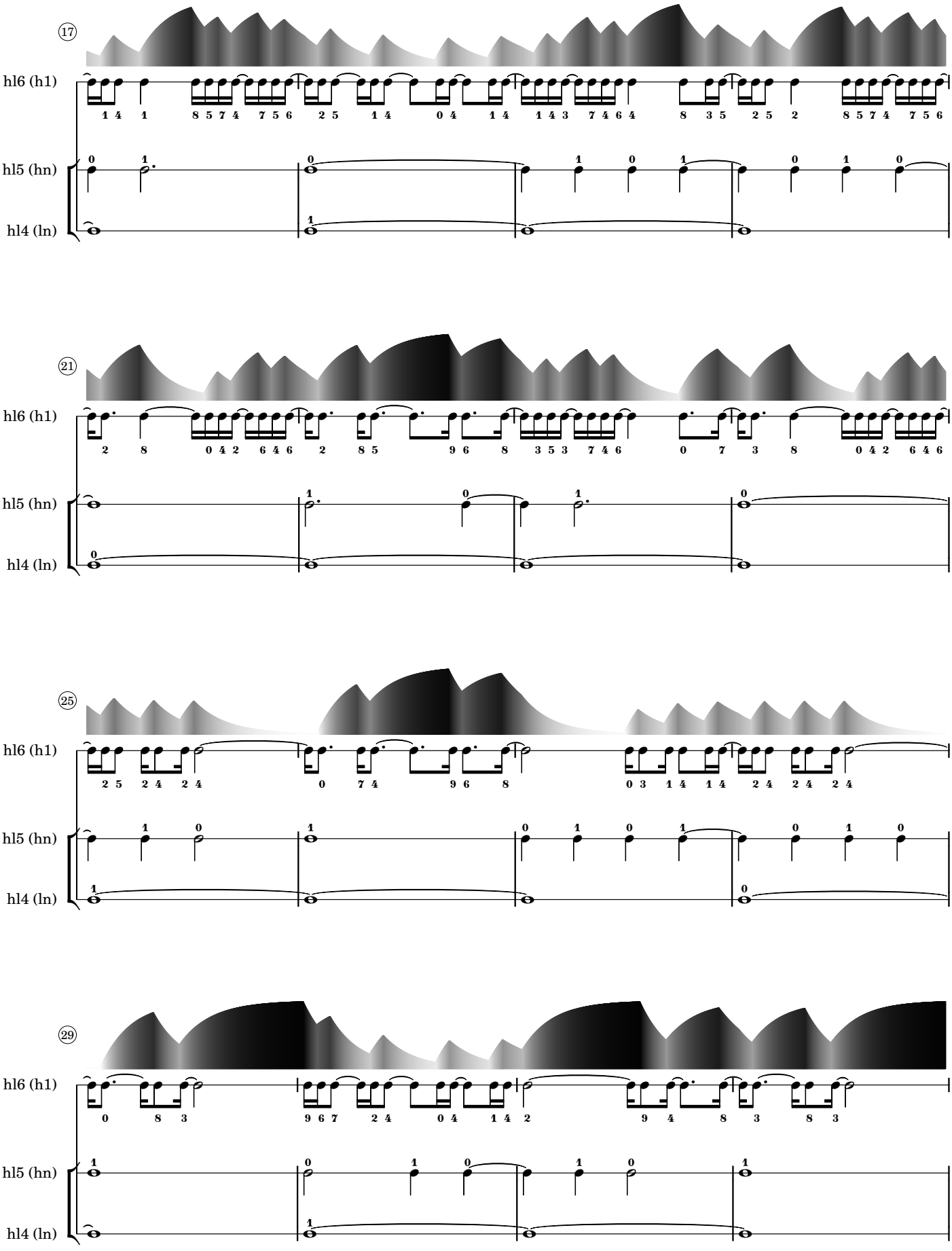
a lot of tiles (trivial scan)

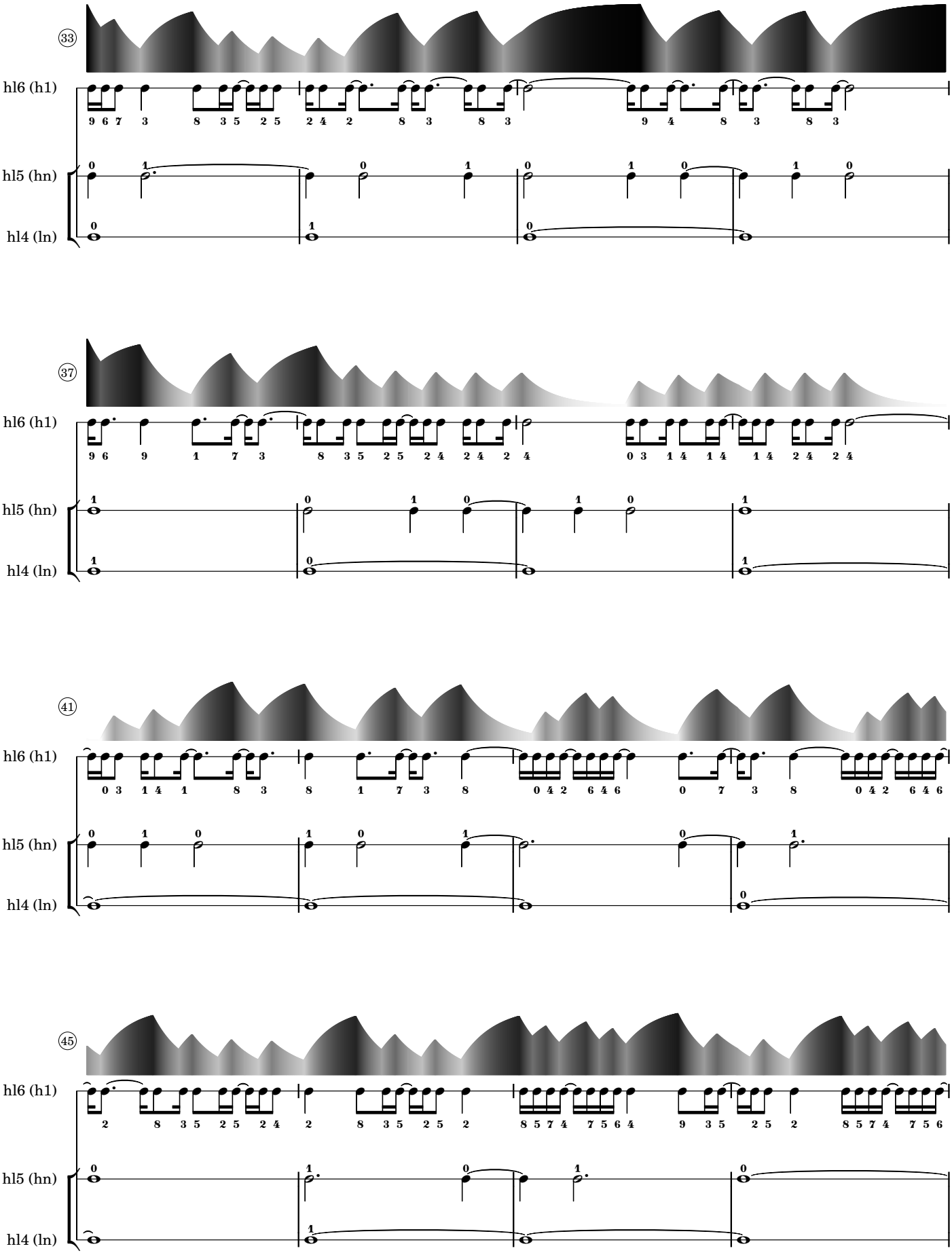
2101000

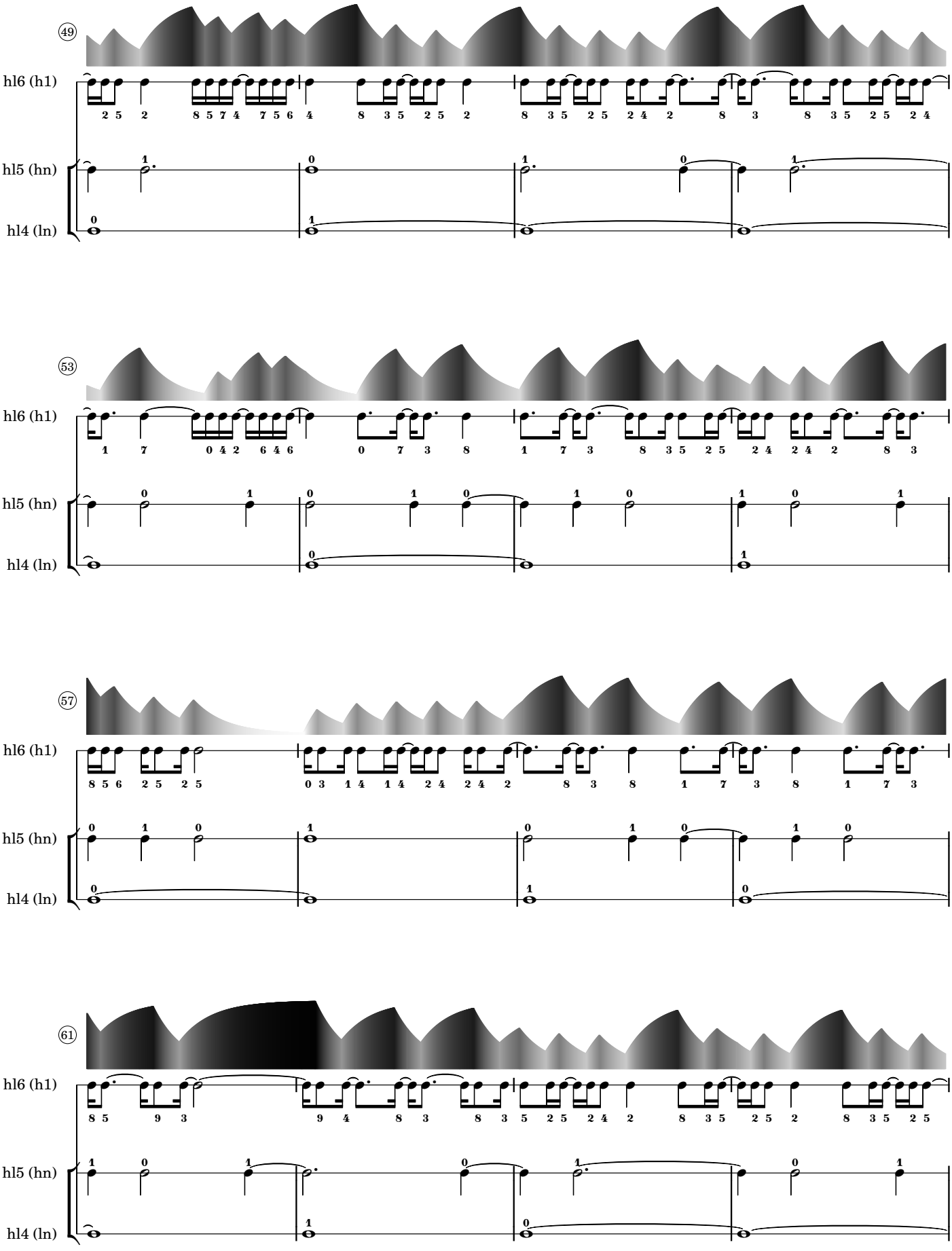
michael winter
(cdmx, mx and nashville, usa; 2018)

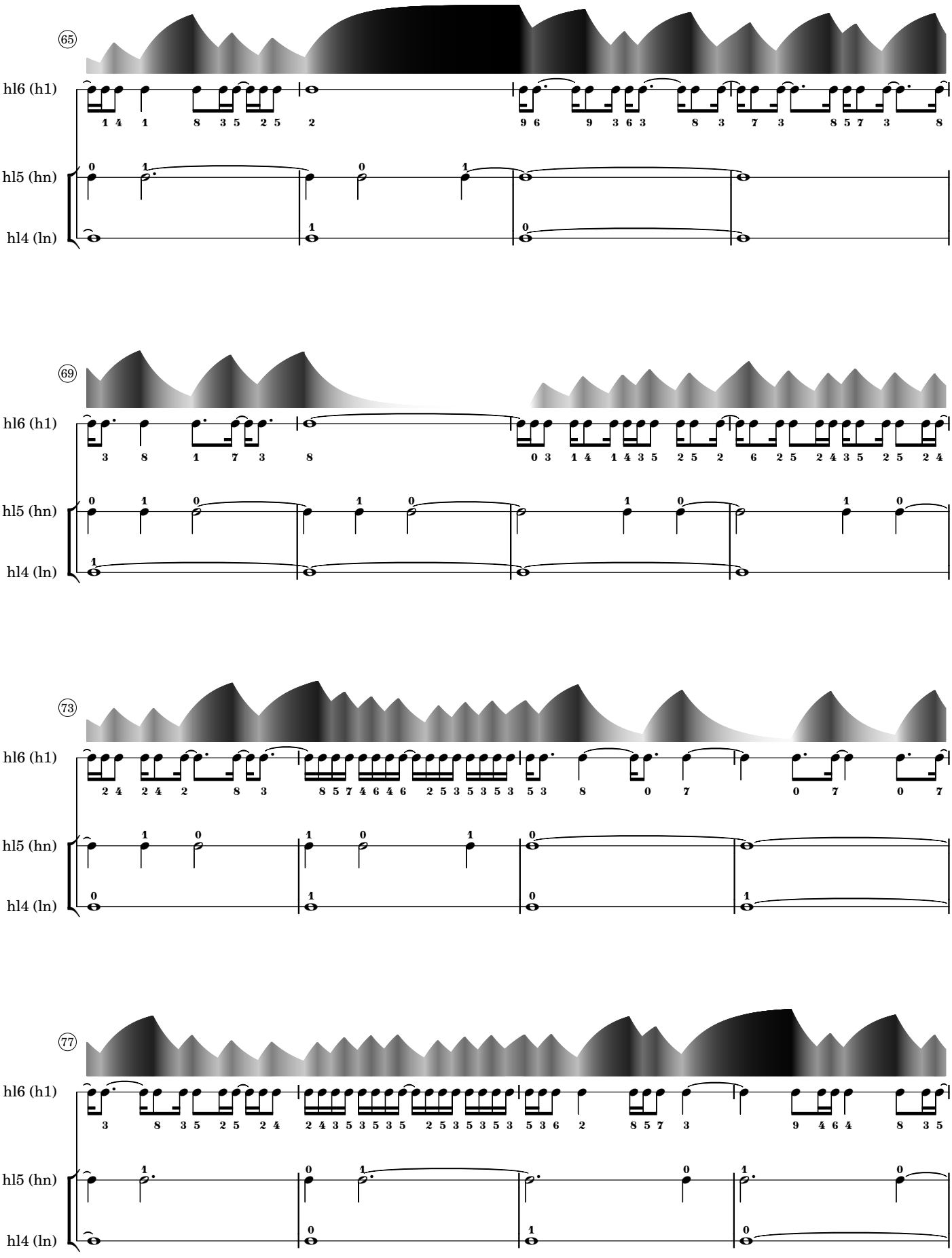
master score - 2101000

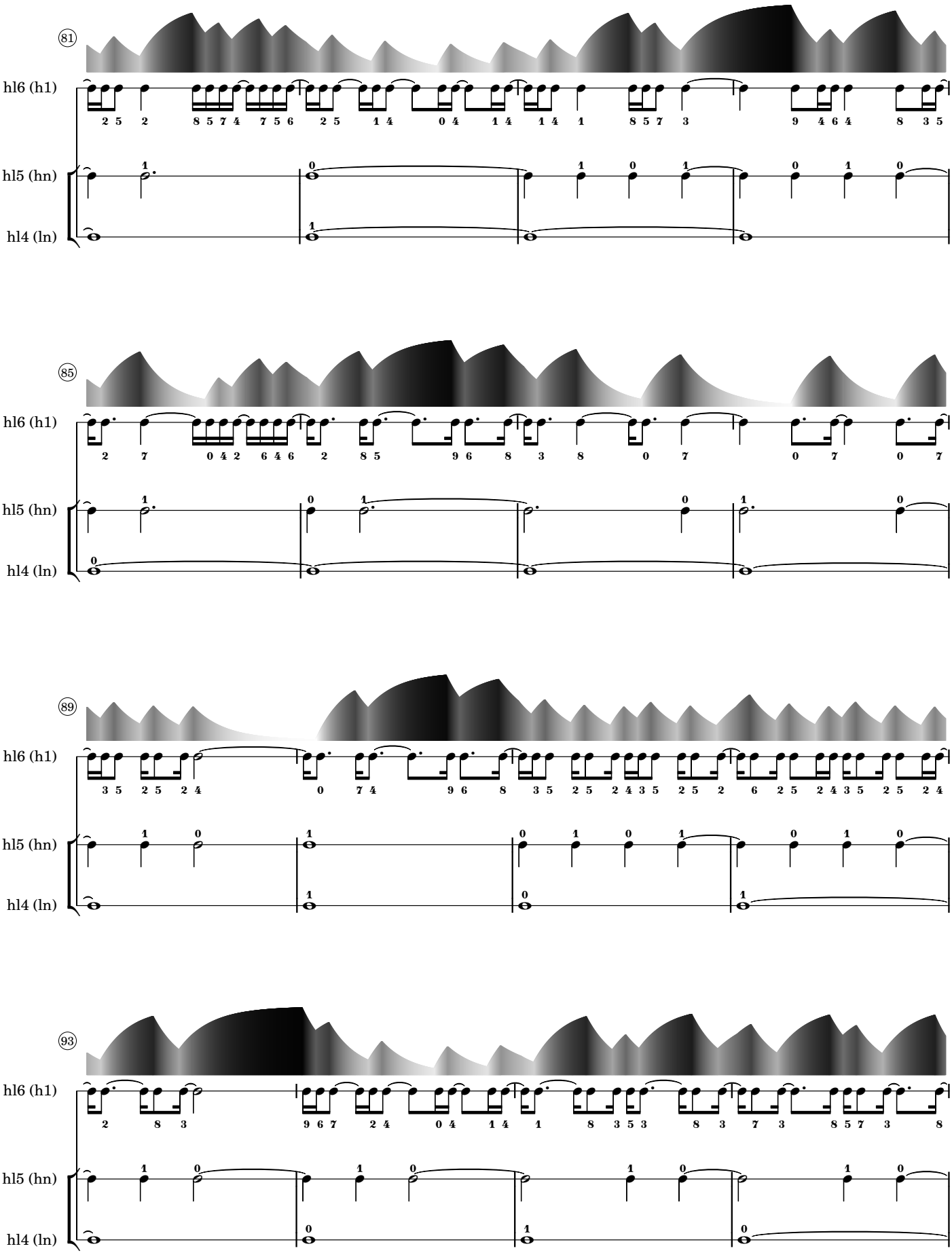


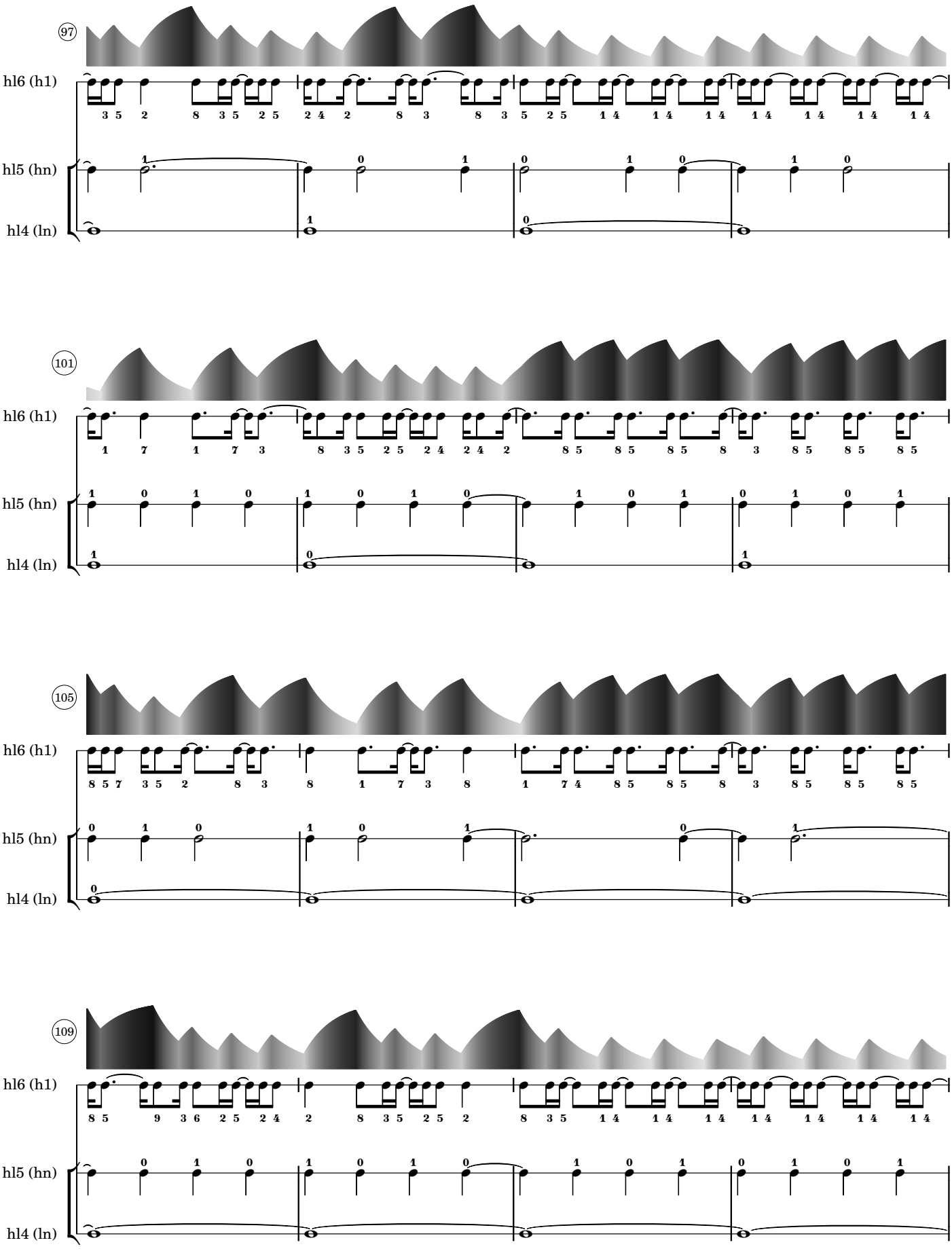

17
hl6 (h1)
hl5 (hn)
hl4 (ln)
21
hl6 (h1)
hl5 (hn)
hl4 (ln)
25
hl6 (h1)
hl5 (hn)
hl4 (ln)
29
hl6 (h1)
hl5 (hn)
hl4 (ln)

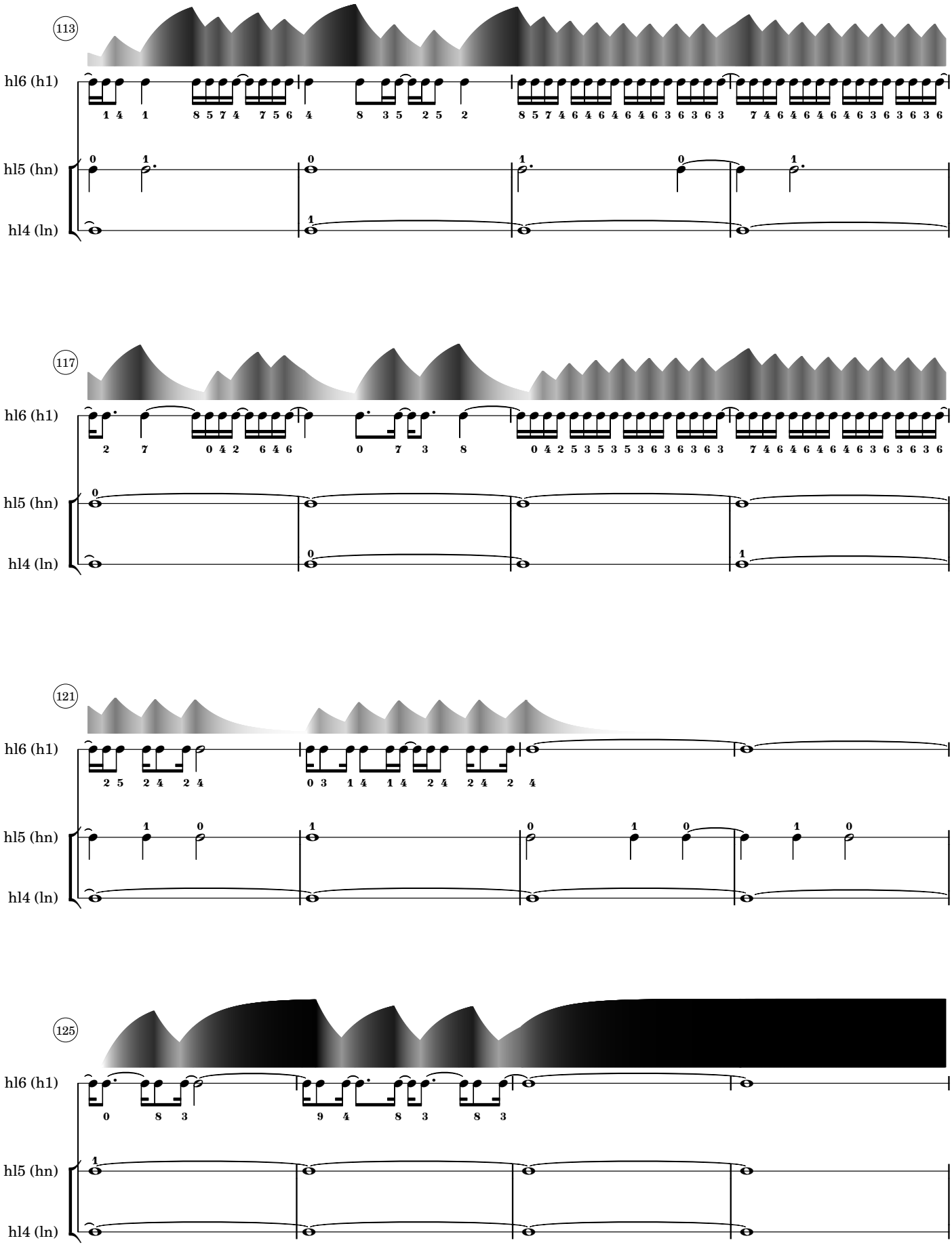

33
hl6 (h1)
hl5 (hn)
hl4 (ln)
37
hl6 (h1)
hl5 (hn)
hl4 (ln)
41
hl6 (h1)
hl5 (hn)
hl4 (ln)
45
hl6 (h1)
hl5 (hn)
hl4 (ln)

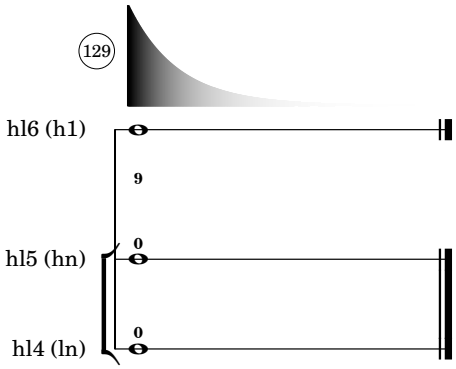

49
hl6 (h1)
hl5 (hn)
hl4 (ln)
53
hl6 (h1)
hl5 (hn)
hl4 (ln)
57
hl6 (h1)
hl5 (hn)
hl4 (ln)
61
hl6 (h1)
hl5 (hn)
hl4 (ln)


65
hl6 (h1)
hl5 (hn)
hl4 (ln)
69
hl6 (h1)
hl5 (hn)
hl4 (ln)
73
hl6 (h1)
hl5 (hn)
hl4 (ln)
77
hl6 (h1)
hl5 (hn)
hl4 (ln)


81
hl6 (h1)
hl5 (hn)
hl4 (ln)
85
hl6 (h1)
hl5 (hn)
hl4 (ln)
89
hl6 (h1)
hl5 (hn)
hl4 (ln)
93
hl6 (h1)
hl5 (hn)
hl4 (ln)


97
hl6 (h1)
hl5 (hn)
hl4 (ln)
101
hl6 (h1)
hl5 (hn)
hl4 (ln)
105
hl6 (h1)
hl5 (hn)
hl4 (ln)
109
hl6 (h1)
hl5 (hn)
hl4 (ln)


113
hl6 (h1)
hl5 (hn)
hl4 (ln)
117
hl6 (h1)
hl5 (hn)
hl4 (ln)
121
hl6 (h1)
hl5 (hn)
hl4 (ln)
125
hl6 (h1)
hl5 (hn)
hl4 (ln)

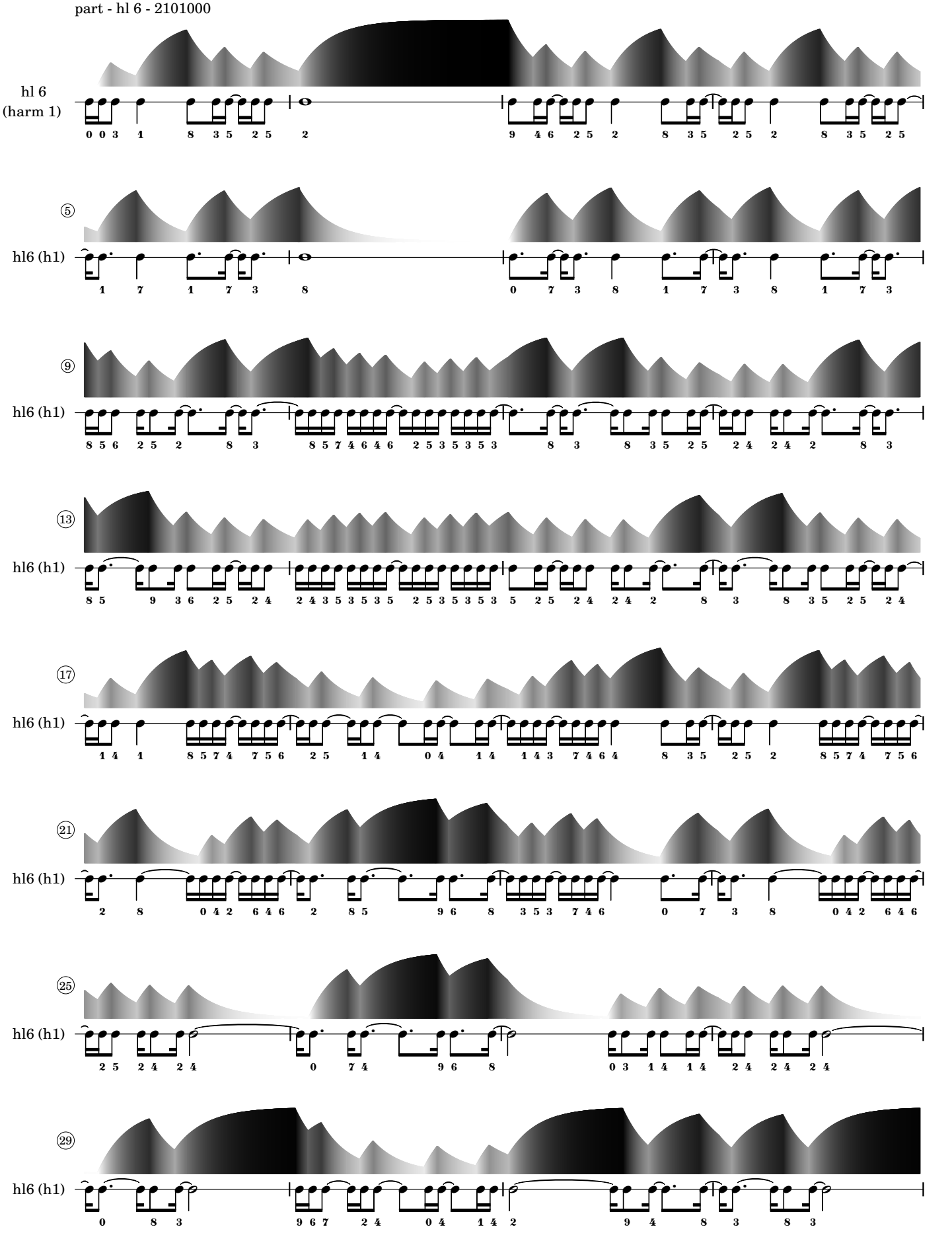

129
hl6 (h1)
9
hl5 (hn)
0
hl4 (ln)
0

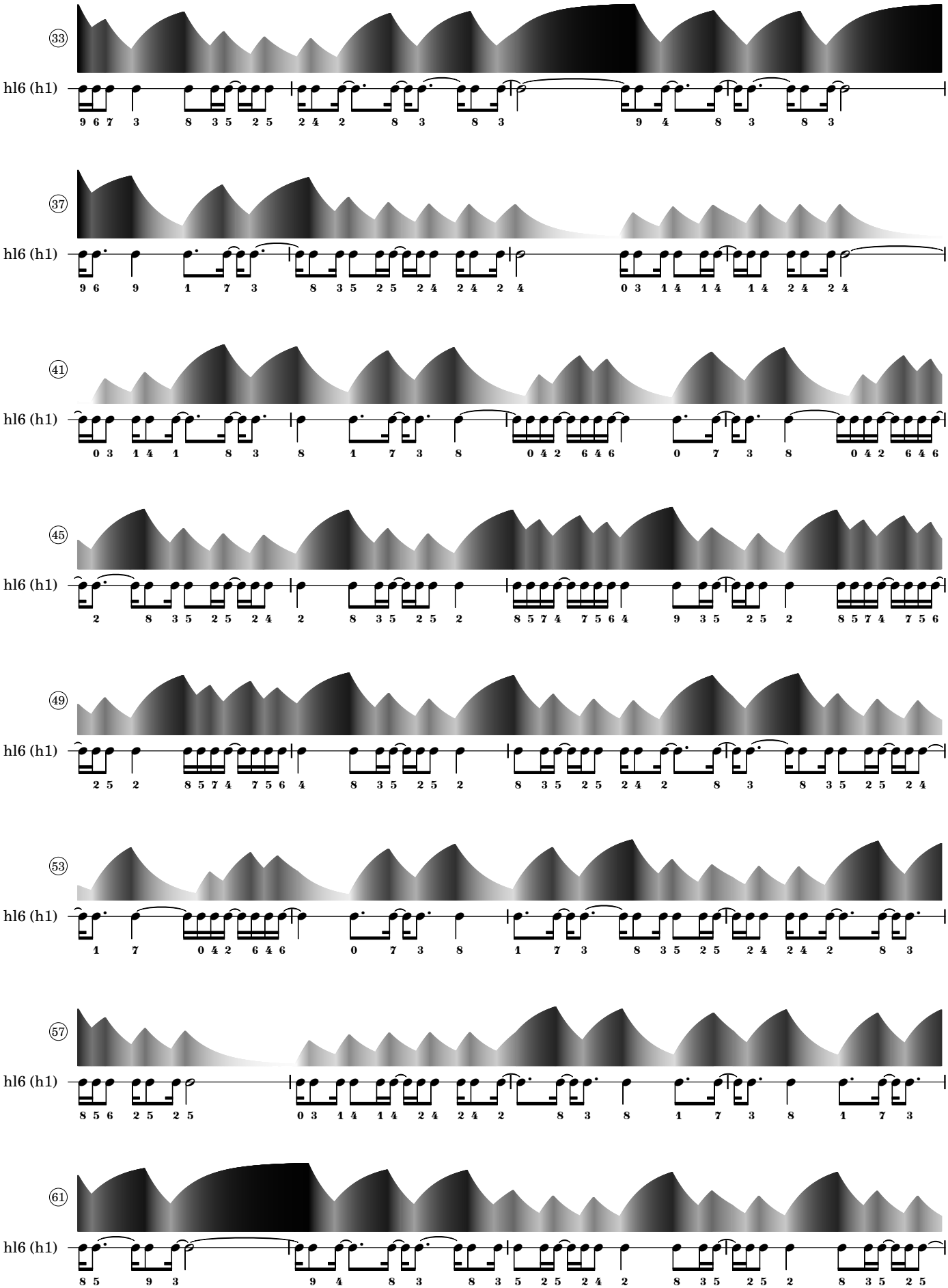
a lot of tiles (trivial scan)

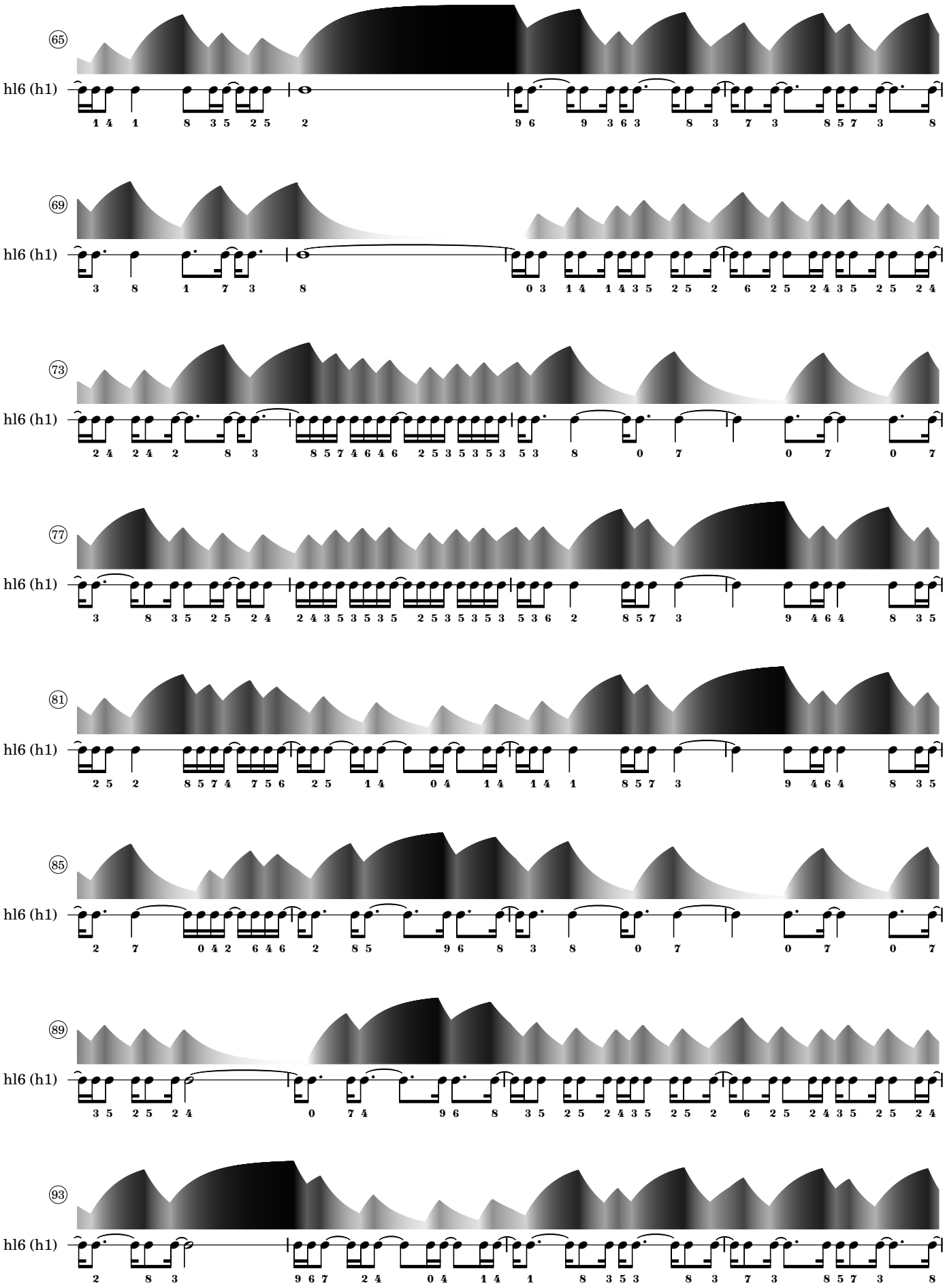
2101000

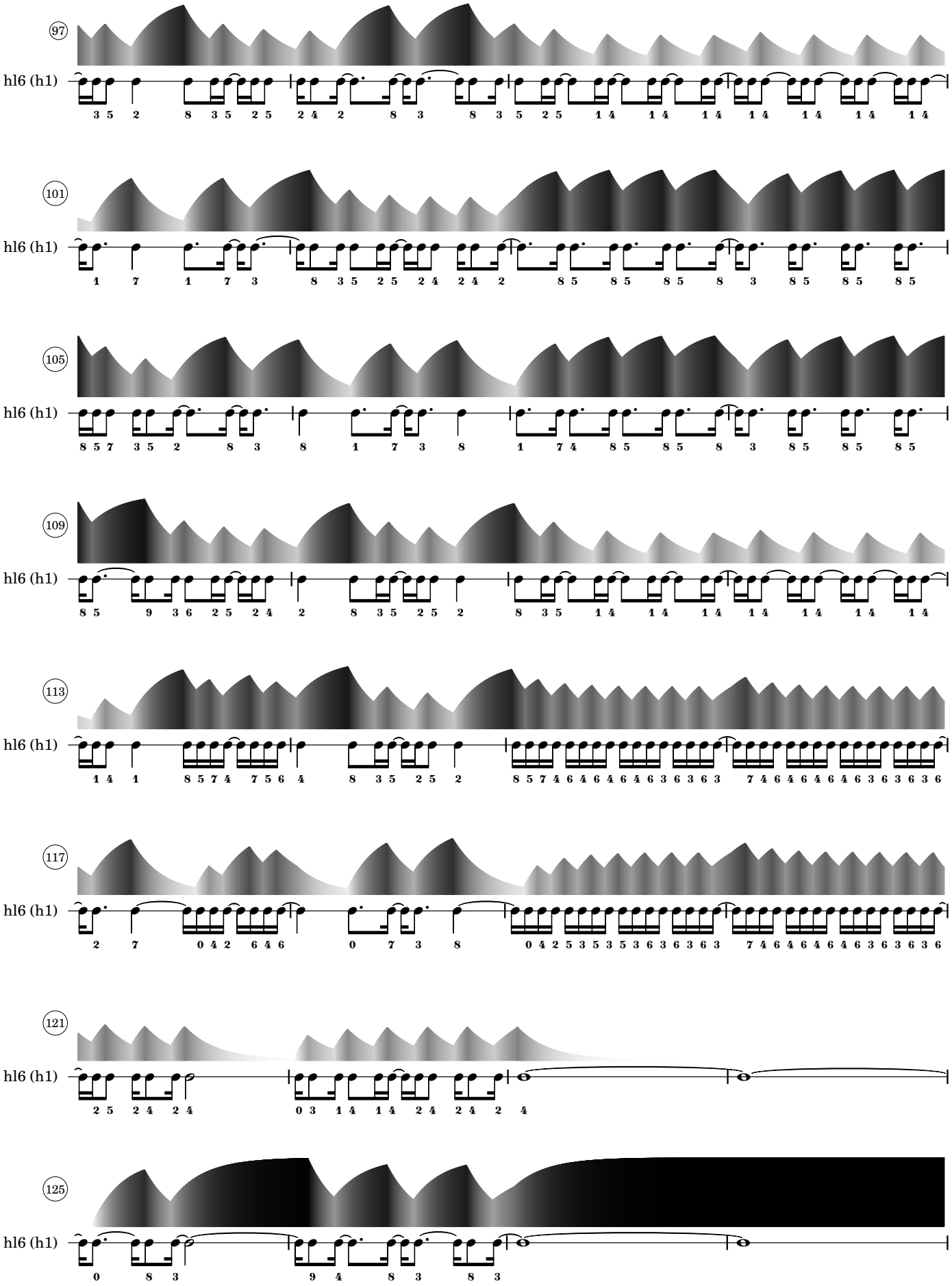
michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 6 - 2101000

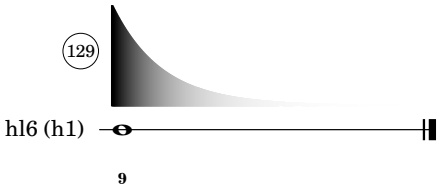



33
hl6 (h1)
9 6 7 3 8 3 5 2 5 2 4 2 8 3 8 3 9 4 8 3 8 3
37
hl6 (h1)
9 6 9 1 7 3 8 3 5 2 5 2 4 2 4 2 4 0 3 1 4 1 4 1 4 2 4 2 4
41
hl6 (h1)
0 3 1 4 1 8 3 8 1 7 3 8 0 4 2 6 4 6 0 7 3 8 0 4 2 6 4 6
45
hl6 (h1)
2 8 3 5 2 5 2 4 2 8 3 5 2 5 2 8 5 7 4 7 5 6 4 9 3 5 2 5 2 8 5 7 4 7 5 6
49
hl6 (h1)
2 5 2 8 5 7 4 7 5 6 4 8 3 5 2 5 2 8 3 5 2 5 2 4 2 8 3 8 3 5 2 5 2 4
53
hl6 (h1)
1 7 0 4 2 6 4 6 0 7 3 8 1 7 3 8 3 5 2 5 2 4 2 4 2 8 3
57
hl6 (h1)
8 5 6 2 5 2 5 0 3 1 4 1 4 2 4 2 4 2 8 3 8 1 7 3 8 1 7 3
61
hl6 (h1)
8 5 9 3 9 4 8 3 8 3 5 2 5 2 4 2 8 3 5 2 5 2 8 3 5 2 5


65
hl6 (h1)
1 4 1 8 3 5 2 5 2 9 6 9 3 6 3 8 3 7 3 8 5 7 3 8
69
hl6 (h1)
3 8 1 7 3 8 0 3 1 4 1 4 3 5 2 5 2 6 2 5 2 4 3 5 2 5 2 4
73
hl6 (h1)
2 4 2 4 2 8 3 8 5 7 4 6 4 6 2 5 3 5 3 5 3 5 3 8 0 7 0 7 0 7
77
hl6 (h1)
3 8 3 5 2 5 2 4 2 4 3 5 3 5 3 5 2 5 3 5 3 5 3 5 3 6 2 8 5 7 3 9 4 6 4 8 3 5
81
hl6 (h1)
2 5 2 8 5 7 4 7 5 6 2 5 1 4 0 4 1 4 1 4 1 8 5 7 3 9 4 6 4 8 3 5
85
hl6 (h1)
2 7 0 4 2 6 4 6 2 8 5 9 6 8 3 8 0 7 0 7 0 7
89
hl6 (h1)
3 5 2 5 2 4 0 7 4 9 6 8 3 5 2 5 2 4 3 5 2 5 2 6 2 5 2 4 3 5 2 5 2 4
93
hl6 (h1)
2 8 3 9 6 7 2 4 0 4 1 4 1 8 3 5 3 8 3 7 3 8 5 7 3 8


97
hl6 (h1)
3 5 2 8 3 5 2 5 2 4 2 8 3 8 3 5 2 5 1 4 1 4 1 4 1 4 1 4 1 4 1 4
101
hl6 (h1)
1 7 1 7 3 8 3 5 2 5 2 4 2 4 2 8 5 8 5 8 5 8 3 8 5 8 5 8 5
105
hl6 (h1)
8 5 7 3 5 2 8 3 8 1 7 3 8 1 7 4 8 5 8 5 8 3 8 5 8 5 8 5
109
hl6 (h1)
8 5 9 3 6 2 5 2 4 2 8 3 5 2 5 2 8 3 5 1 4 1 4 1 4 1 4 1 4 1 4 1 4
113
hl6 (h1)
1 4 1 8 5 7 4 7 5 6 4 8 3 5 2 5 2 8 5 7 4 6 4 6 4 6 4 6 3 6 3 6 3 7 4 6 4 6 4 6 4 6 3 6 3 6 3 6
117
hl6 (h1)
2 7 0 4 2 6 4 6 0 7 3 8 0 4 2 5 3 5 3 5 3 6 3 6 3 6 3 7 4 6 4 6 4 6 4 6 3 6 3 6 3 6
121
hl6 (h1)
2 5 2 4 2 4 0 3 1 4 1 4 2 4 2 4 2 4
125
hl6 (h1)
0 8 3 9 4 8 3 8 3

(part - hl 6 - 2101000)

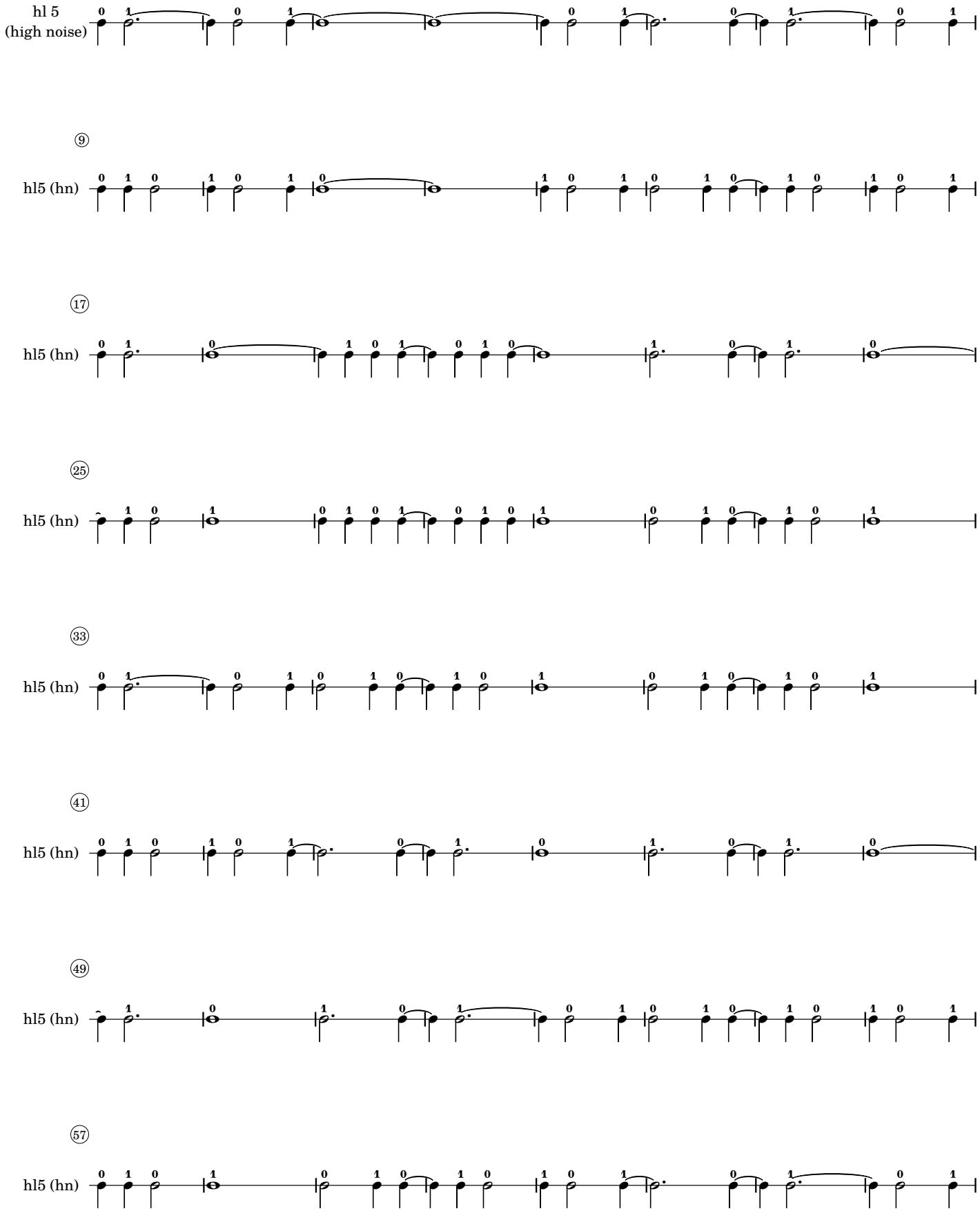


a lot of tiles (trivial scan)

2101000

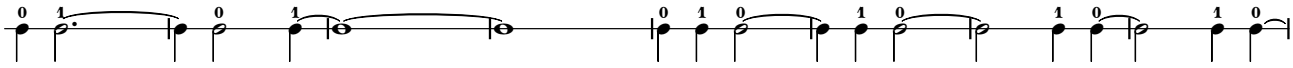
michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 5 - 2101000



65

hl5 (hn)

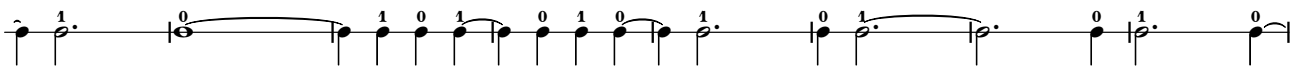


73

hl5 (hn)

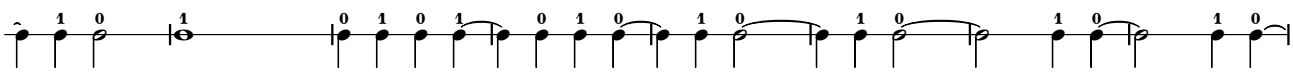
81

hl5 (hn)



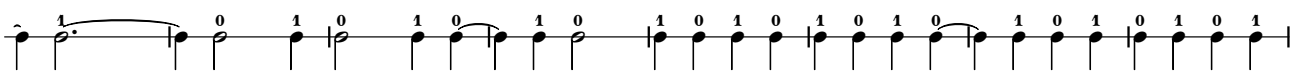
89

hl5 (hn)



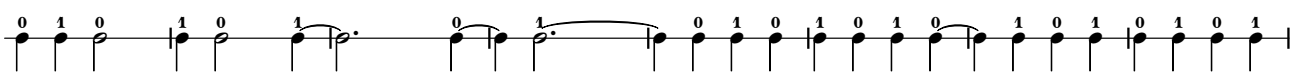
97

hl5 (hn)



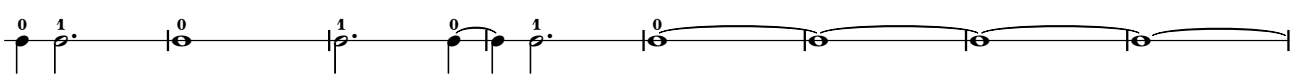
105

hl5 (hn)



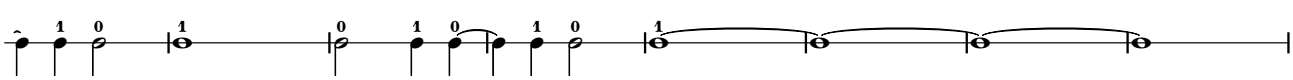
113

hl5 (hn)



121

hl5 (hn)



(part - hl 5 - 2101000)

129

hl5 (hn) 

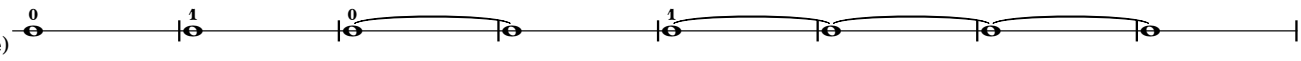
a lot of tiles (trivial scan)

2101000

michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 4 - 2101000

hl 4
(low noise)

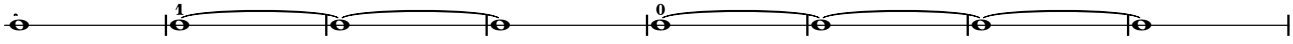


⑨

hl4 (ln)

⑰

hl4 (ln)

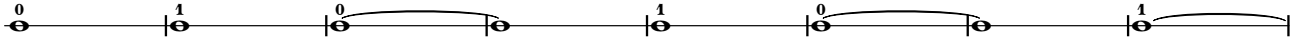


㉕

hl4 (ln)

㉝

hl4 (ln)



㊶

hl4 (ln)

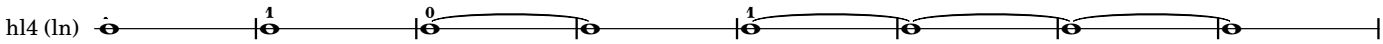
㊾

hl4 (ln)

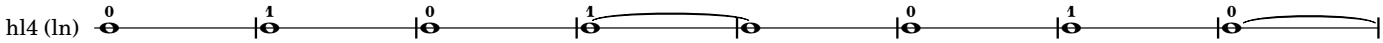
㊼

hl4 (ln)

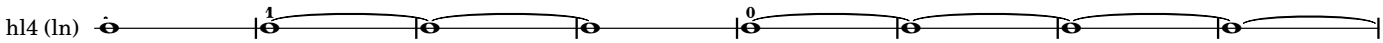
65



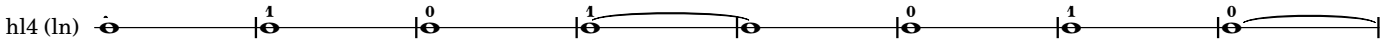
73



81

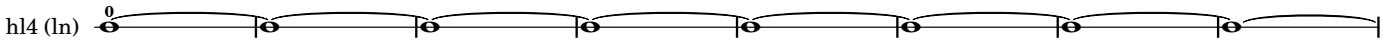


89

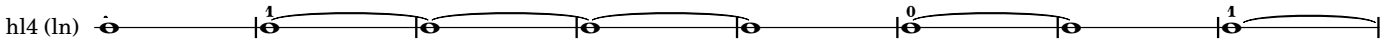


97

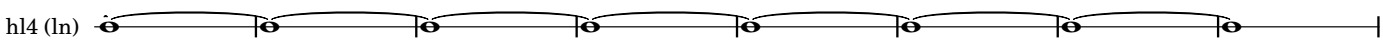
105



113



121



(part - hl 4 - 2101000)

129

hl4 (ln) 

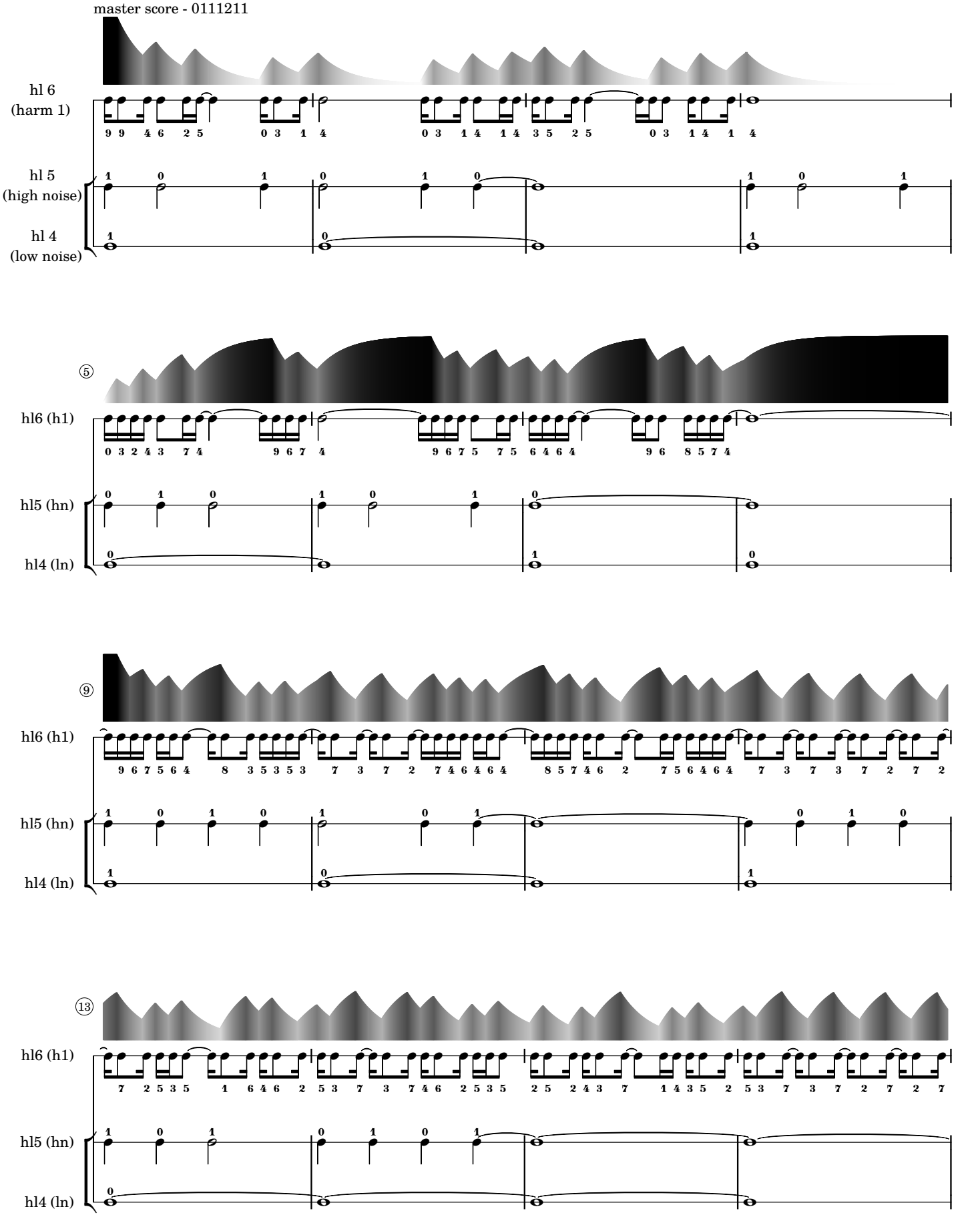
a lot of tiles (trivial scan)

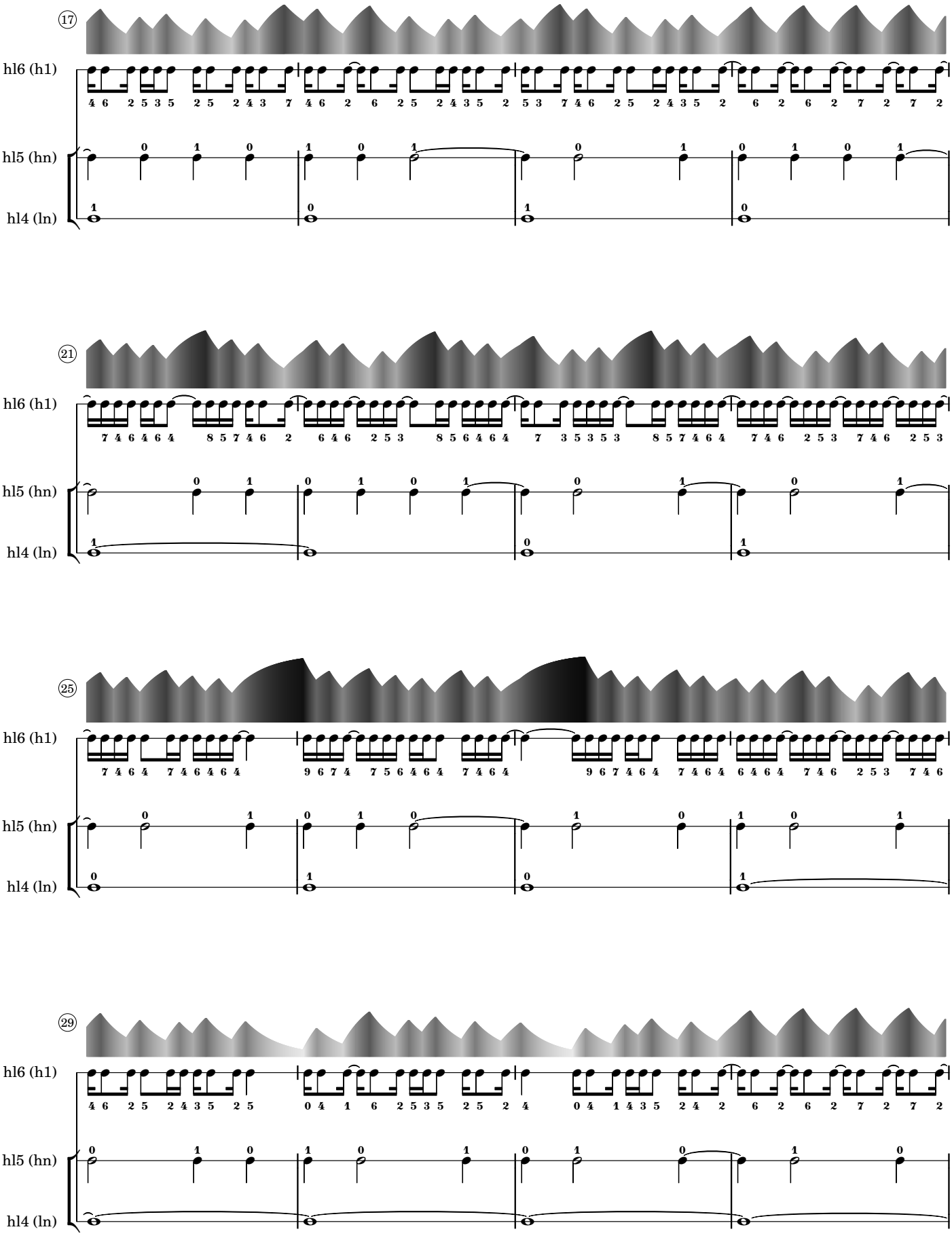
0111211

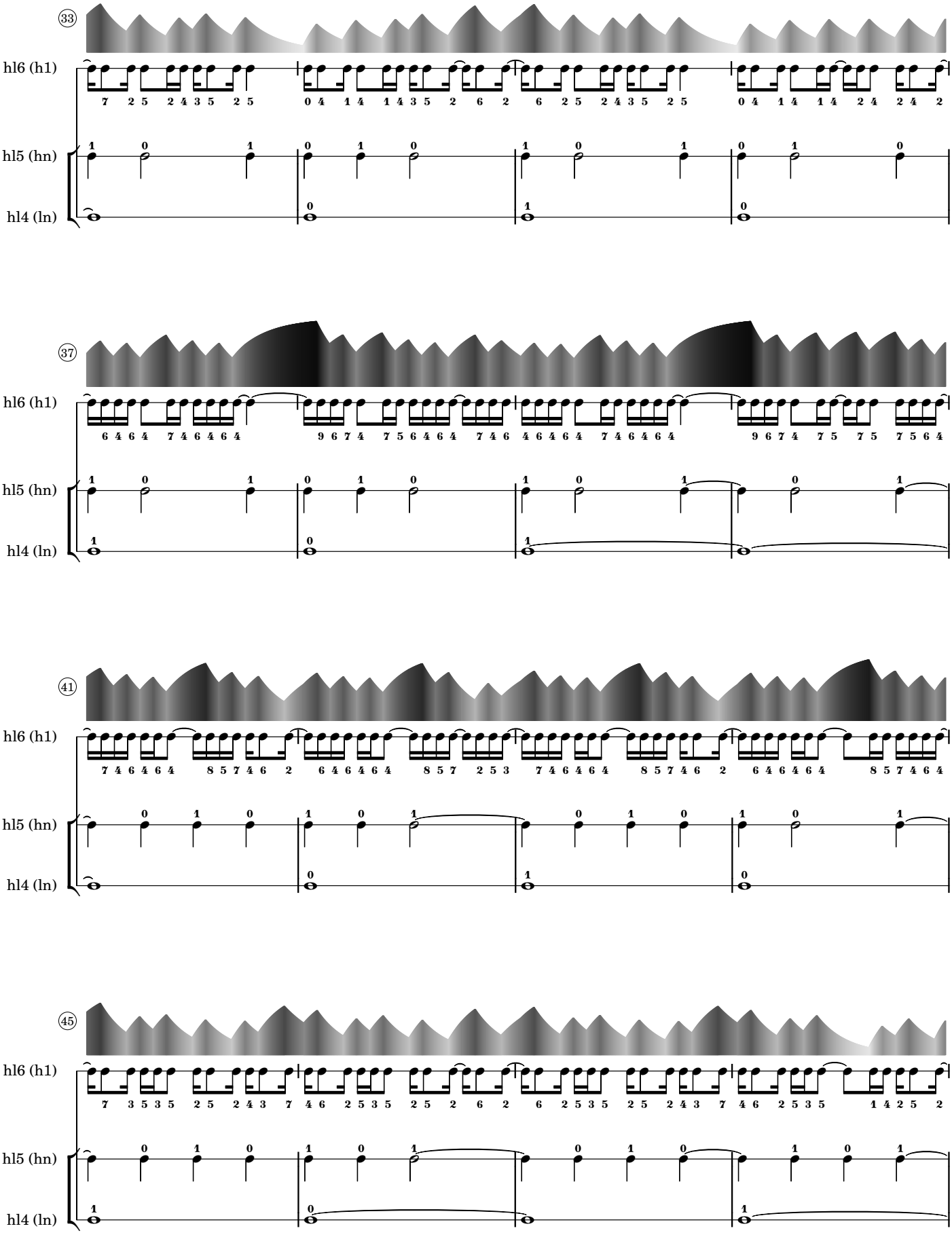
michael winter

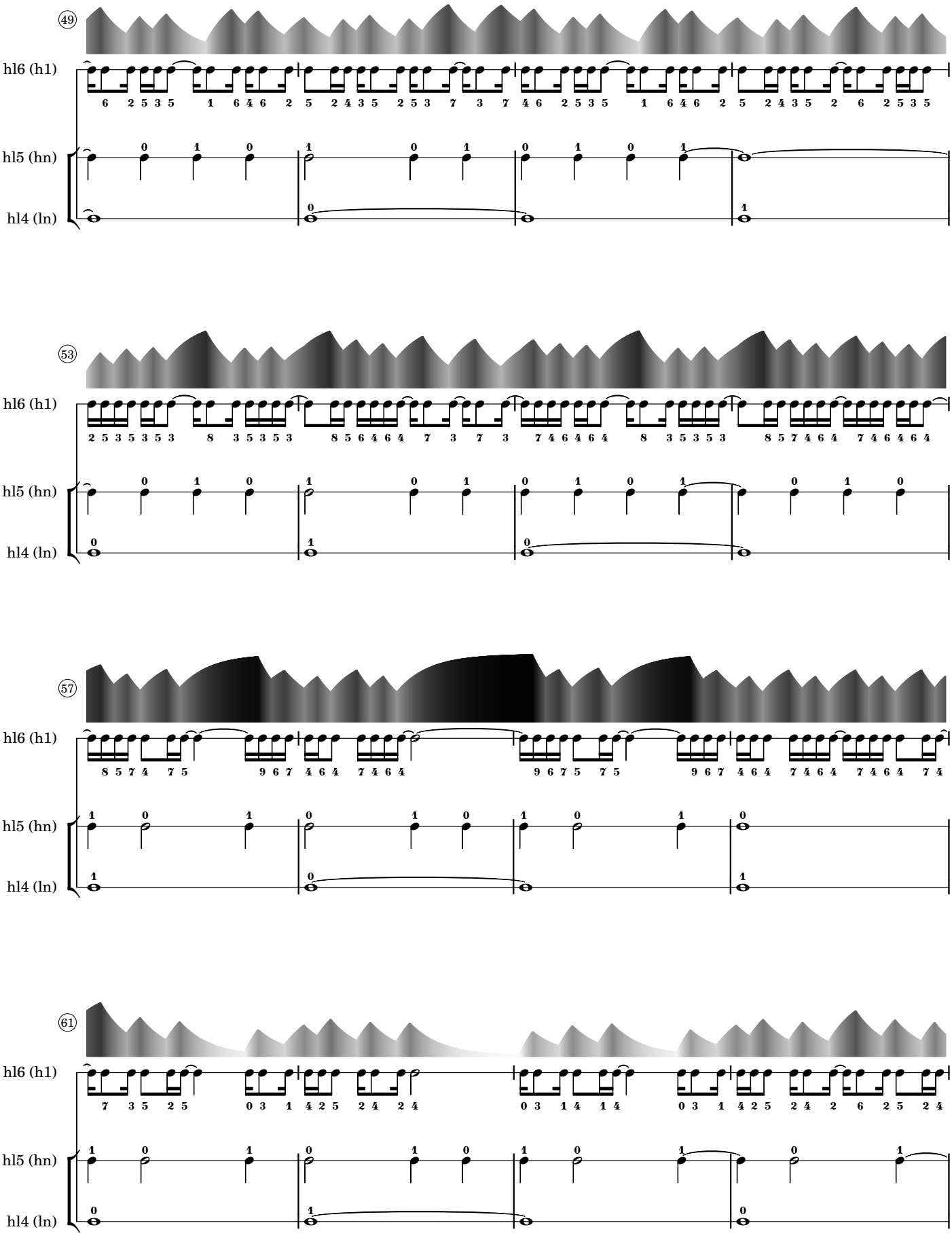
(cdmx, mx and nashville, usa; 2018)

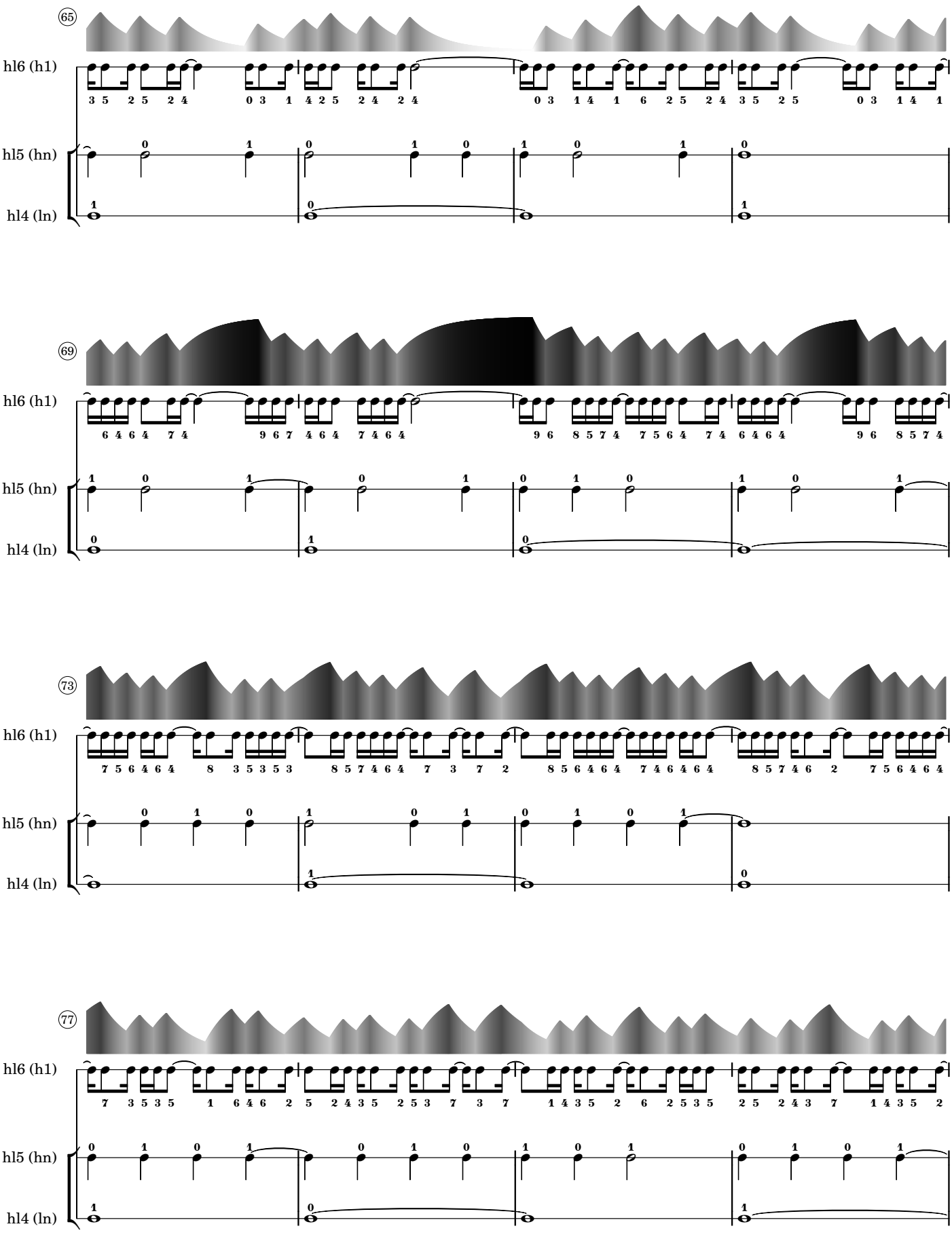
master score - 0111211

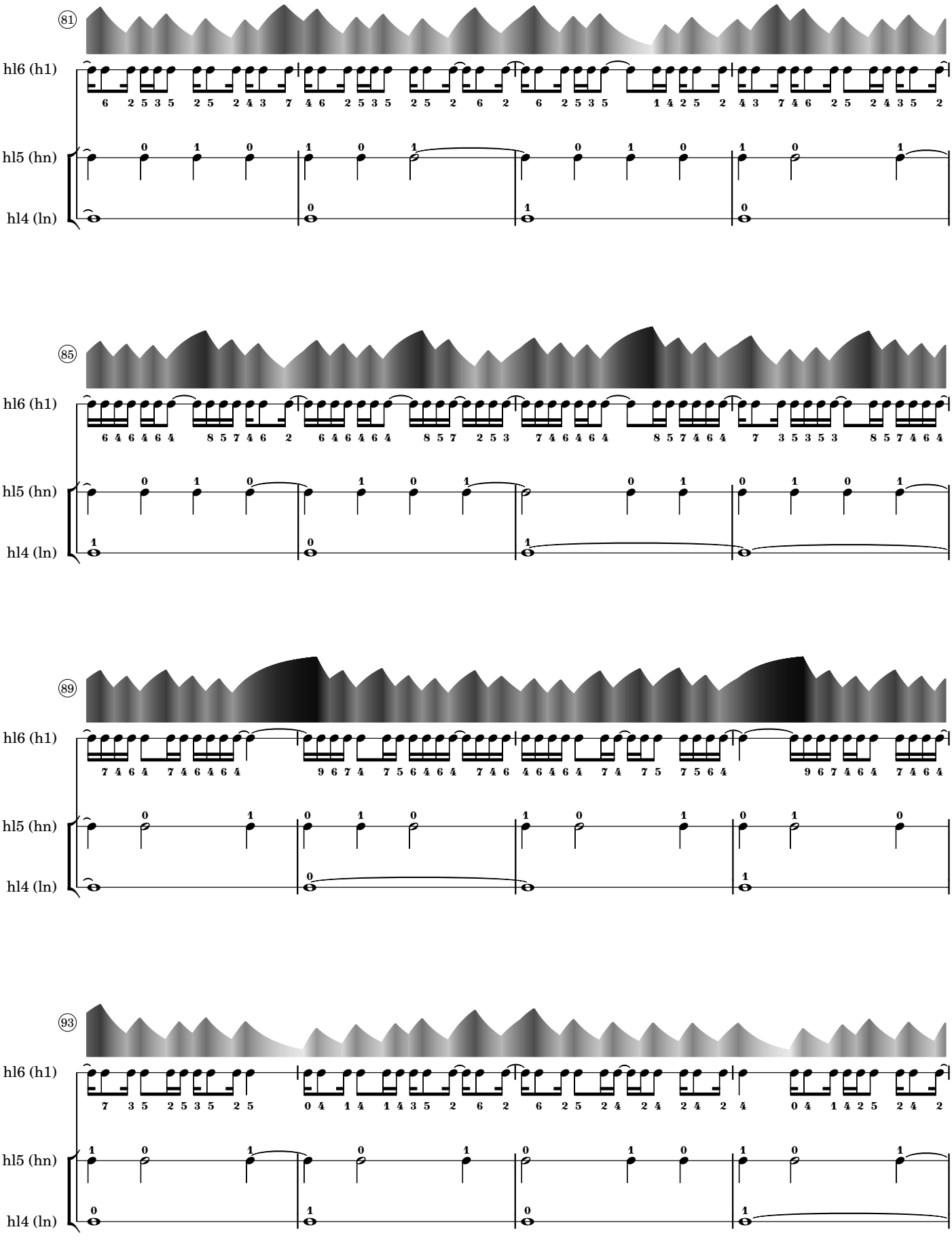


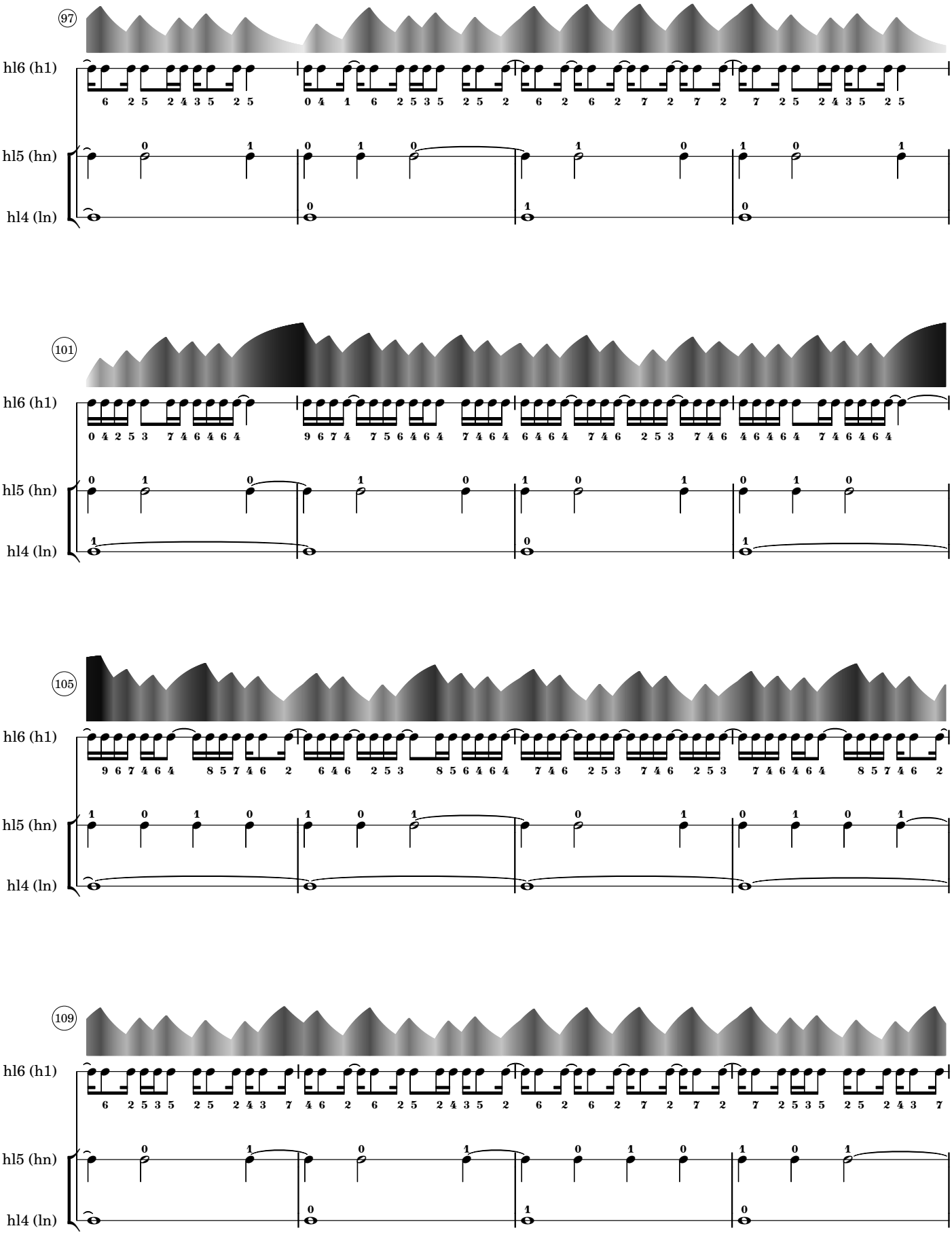

17
hl6 (h1)
4 6 2 5 3 5 2 5 2 4 3 7 4 6 2 6 2 5 2 4 3 5 2 5 3 7 4 6 2 5 2 4 3 5 2 6 2 6 2 7 2 7 2
hl5 (hn)
hl4 (ln)
21
hl6 (h1)
7 4 6 4 6 4 8 5 7 4 6 2 6 4 6 2 5 3 8 5 6 4 6 4 7 3 5 3 5 3 8 5 7 4 6 4 7 4 6 2 5 3 7 4 6 2 5 3
hl5 (hn)
hl4 (ln)
25
hl6 (h1)
7 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 6 4 6 4 7 4 6 4 9 6 7 4 6 4 7 4 6 4 6 4 6 4 7 4 6 2 5 3 7 4 6
hl5 (hn)
hl4 (ln)
29
hl6 (h1)
4 6 2 5 2 4 3 5 2 5 0 4 1 6 2 5 3 5 2 5 2 4 0 4 1 4 3 5 2 4 2 6 2 6 2 7 2 7 2
hl5 (hn)
hl4 (ln)

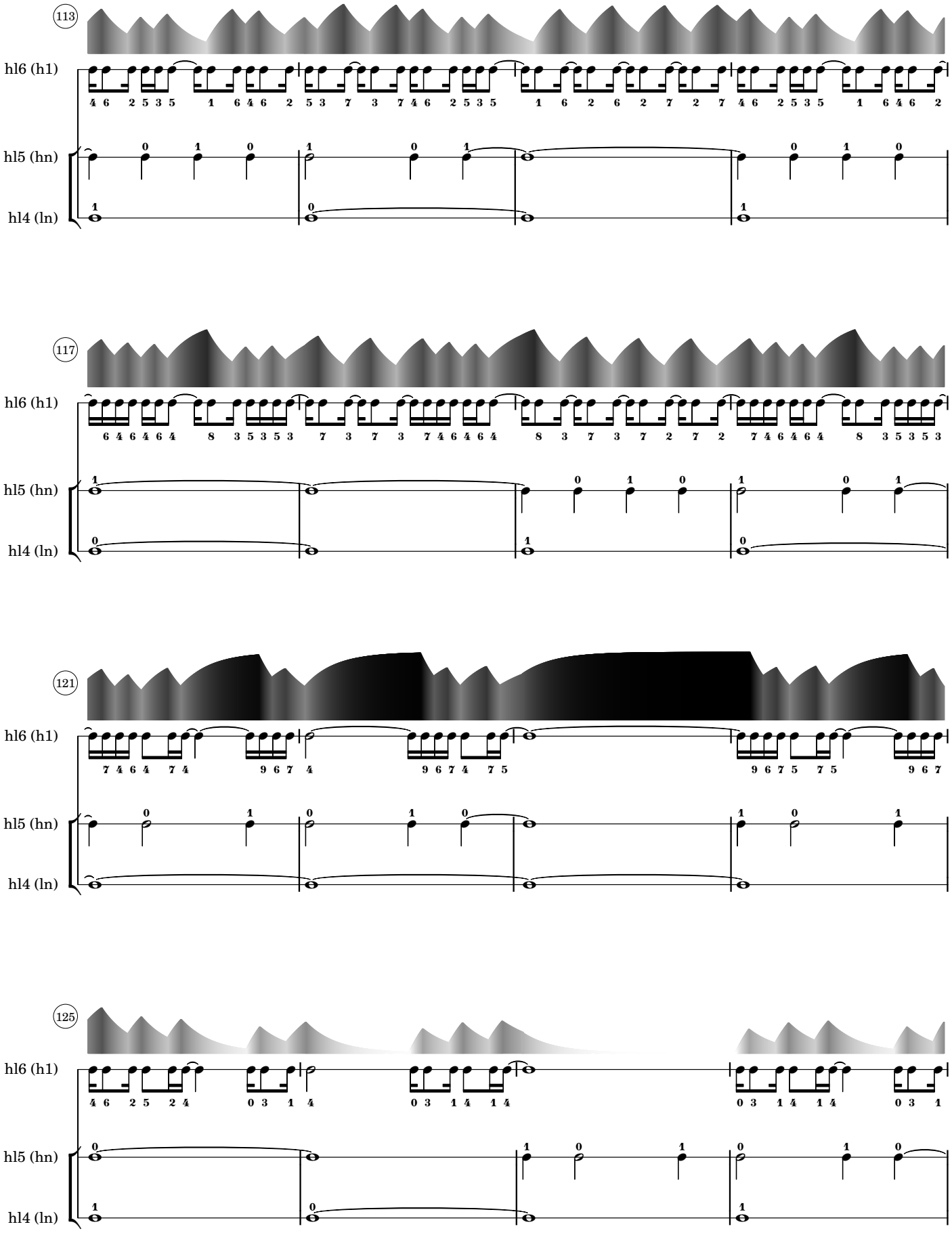

33
hl6 (h1)
7 2 5 2 4 3 5 2 5 0 4 1 4 1 4 3 5 2 6 2 6 2 5 2 4 3 5 2 5 0 4 1 4 1 4 2 4 2 4 2
hl5 (hn)
hl4 (ln)
37
hl6 (h1)
6 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 6 4 6 4 7 4 6 4 6 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 7 5 7 5 6 4
hl5 (hn)
hl4 (ln)
41
hl6 (h1)
7 4 6 4 6 4 8 5 7 4 6 2 6 4 6 4 6 4 8 5 7 2 5 3 7 4 6 4 6 4 8 5 7 4 6 2 6 4 6 4 6 4 8 5 7 4 6 4
hl5 (hn)
hl4 (ln)
45
hl6 (h1)
7 3 5 3 5 2 5 2 4 3 7 4 6 2 5 3 5 2 5 2 6 2 6 2 5 3 5 2 5 2 4 3 7 4 6 2 5 3 5 1 4 2 5 2
hl5 (hn)
hl4 (ln)

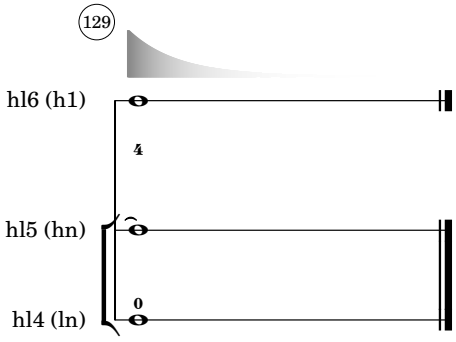

49
hl6 (h1)
6 2 5 3 5 1 6 4 6 2 5 2 4 3 5 2 5 3 7 3 7 4 6 2 5 3 5 1 6 4 6 2 5 2 4 3 5 2 6 2 5 3 5
hl5 (hn)
hl4 (ln)
53
hl6 (h1)
2 5 3 5 3 5 3 8 3 5 3 5 3 8 5 6 4 6 4 7 3 7 3 7 4 6 4 6 4 8 3 5 3 5 3 8 5 7 4 6 4 7 4 6 4 6 4
hl5 (hn)
hl4 (ln)
57
hl6 (h1)
8 5 7 4 7 5 9 6 7 4 6 4 7 4 6 4 9 6 7 5 7 5 9 6 7 4 6 4 7 4 6 4 7 4 6 4 7 4
hl5 (hn)
hl4 (ln)
61
hl6 (h1)
7 3 5 2 5 0 3 1 4 2 5 2 4 2 4 0 3 1 4 1 4 0 3 1 4 2 5 2 4 2 6 2 5 2 4
hl5 (hn)
hl4 (ln)


65
hl6 (h1)
3 5 2 5 2 4 0 3 1 4 2 5 2 4 2 4 0 3 1 4 1 6 2 5 2 4 3 5 2 5 0 3 1 4 1
hl5 (hn)
hl4 (ln)
69
hl6 (h1)
6 4 6 4 7 4 9 6 7 4 6 4 7 4 6 4 9 6 8 5 7 4 7 5 6 4 7 4 6 4 6 4 9 6 8 5 7 4
hl5 (hn)
hl4 (ln)
73
hl6 (h1)
7 5 6 4 6 4 8 3 5 3 5 3 8 5 7 4 6 4 7 3 7 2 8 5 6 4 6 4 7 4 6 4 6 4 8 5 7 4 6 2 7 5 6 4 6 4
hl5 (hn)
hl4 (ln)
77
hl6 (h1)
7 3 5 3 5 1 6 4 6 2 5 2 4 3 5 2 5 3 7 3 7 1 4 3 5 2 6 2 5 3 5 2 5 2 4 3 7 1 4 3 5 2
hl5 (hn)
hl4 (ln)


81
hl6 (h1)
6 2 5 3 5 2 5 2 4 3 7 4 6 2 5 3 5 2 5 2 6 2 6 2 5 3 5 1 4 2 5 2 4 3 7 4 6 2 5 2 4 3 5 2
hl5 (hn)
hl4 (ln)
85
hl6 (h1)
6 4 6 4 6 4 8 5 7 4 6 2 6 4 6 4 6 4 8 5 7 2 5 3 7 4 6 4 6 4 8 5 7 4 6 4 7 3 5 3 5 3 8 5 7 4 6 4
hl5 (hn)
hl4 (ln)
89
hl6 (h1)
7 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 6 4 6 4 7 4 6 4 6 4 6 4 7 4 7 5 7 5 6 4 9 6 7 4 6 4 7 4 6 4
hl5 (hn)
hl4 (ln)
93
hl6 (h1)
7 3 5 2 5 3 5 2 5 0 4 1 4 1 4 3 5 2 6 2 6 2 5 2 4 2 4 2 4 2 4 2 4 0 4 1 4 2 5 2 4 2
hl5 (hn)
hl4 (ln)


97
hl6 (h1)
hl5 (hn)
hl4 (ln)
101
hl6 (h1)
hl5 (hn)
hl4 (ln)
105
hl6 (h1)
hl5 (hn)
hl4 (ln)
109
hl6 (h1)
hl5 (hn)
hl4 (ln)


113
hl6 (h1)
4 6 2 5 3 5 1 6 4 6 2 5 3 7 3 7 4 6 2 5 3 5 1 6 2 6 2 7 2 7 4 6 2 5 3 5 1 6 4 6 2
hl5 (hn)
0 1 0 1 0 1 0 1 0
hl4 (ln)
1 0 1
117
hl6 (h1)
6 4 6 4 6 4 8 3 5 3 5 3 7 3 7 3 7 4 6 4 6 4 8 3 7 3 7 2 7 2 7 4 6 4 6 4 8 3 5 3 5 3
hl5 (hn)
1 0 1 0 1 0 1
hl4 (ln)
0 1 0
121
hl6 (h1)
7 4 6 4 7 4 9 6 7 4 9 6 7 4 7 5 9 6 7 5 7 5 9 6 7
hl5 (hn)
0 1 0 1 0 1 0 1
hl4 (ln)
125
hl6 (h1)
4 6 2 5 2 4 0 3 1 4 0 3 1 4 1 4 0 3 1 4 1 4 0 3 1
hl5 (hn)
0 1 0 1 0 1 0
hl4 (ln)
1 0 1

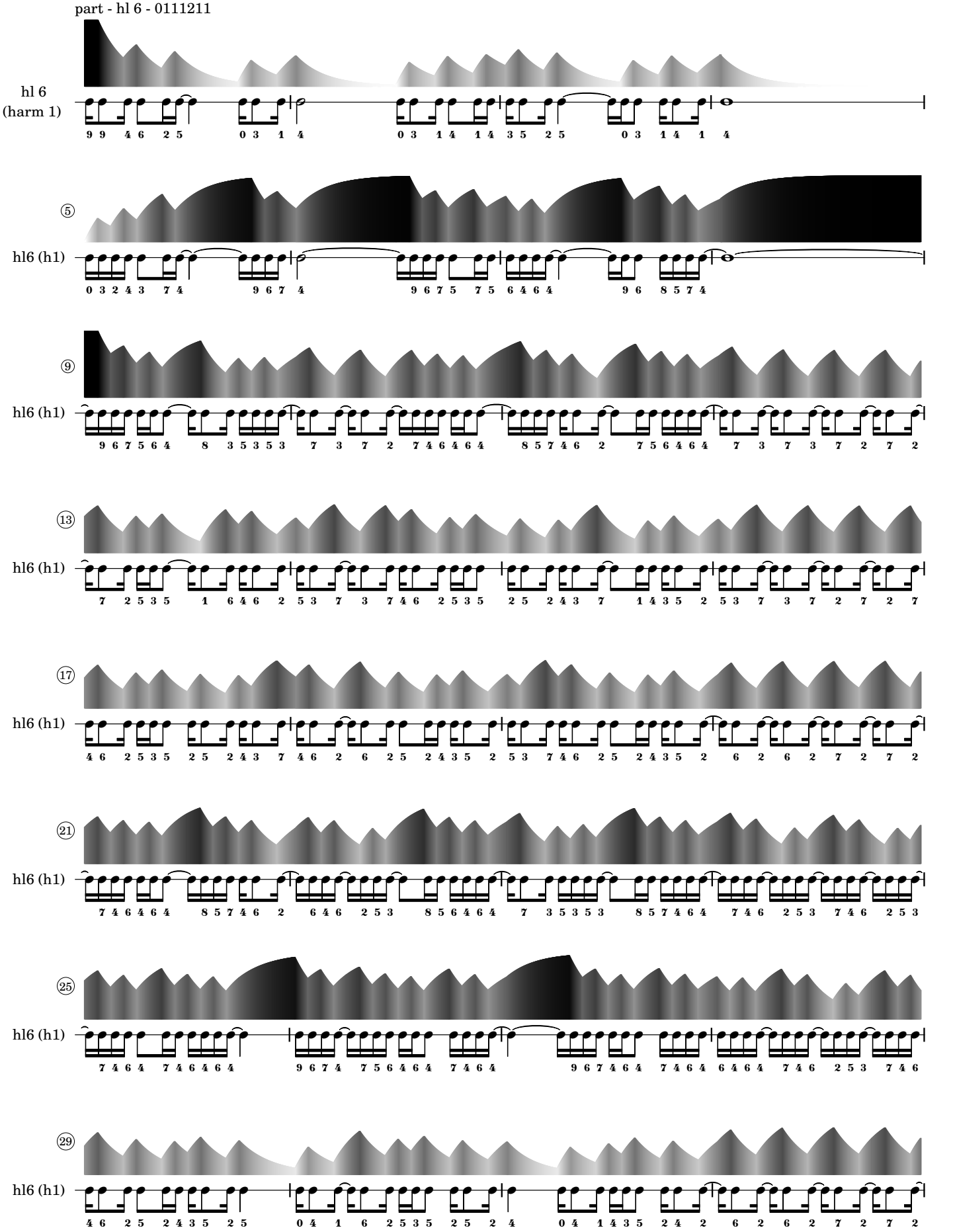

129
hl6 (h1)
4
hl5 (hn)
0
hl4 (ln)

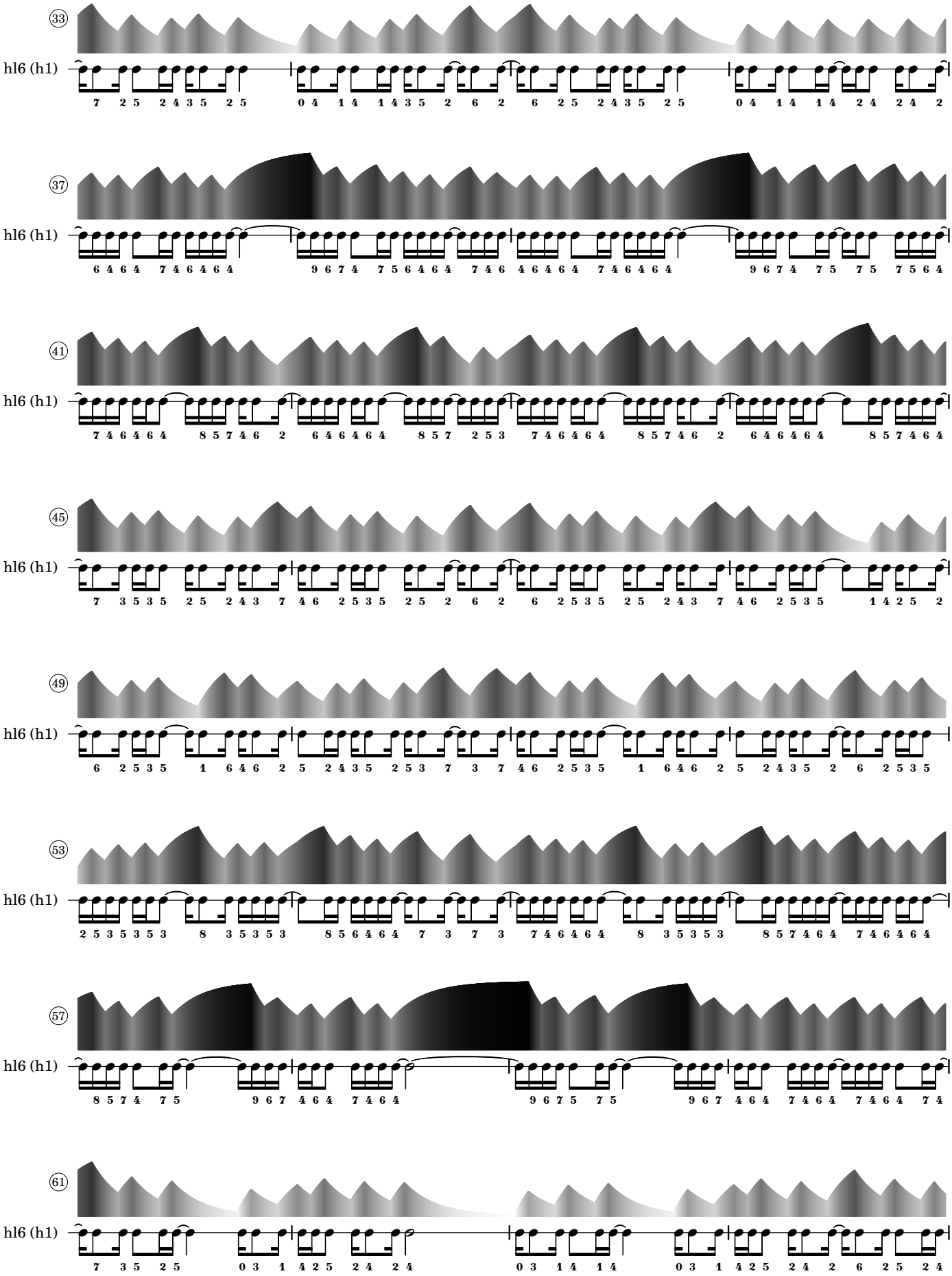
a lot of tiles (trivial scan)

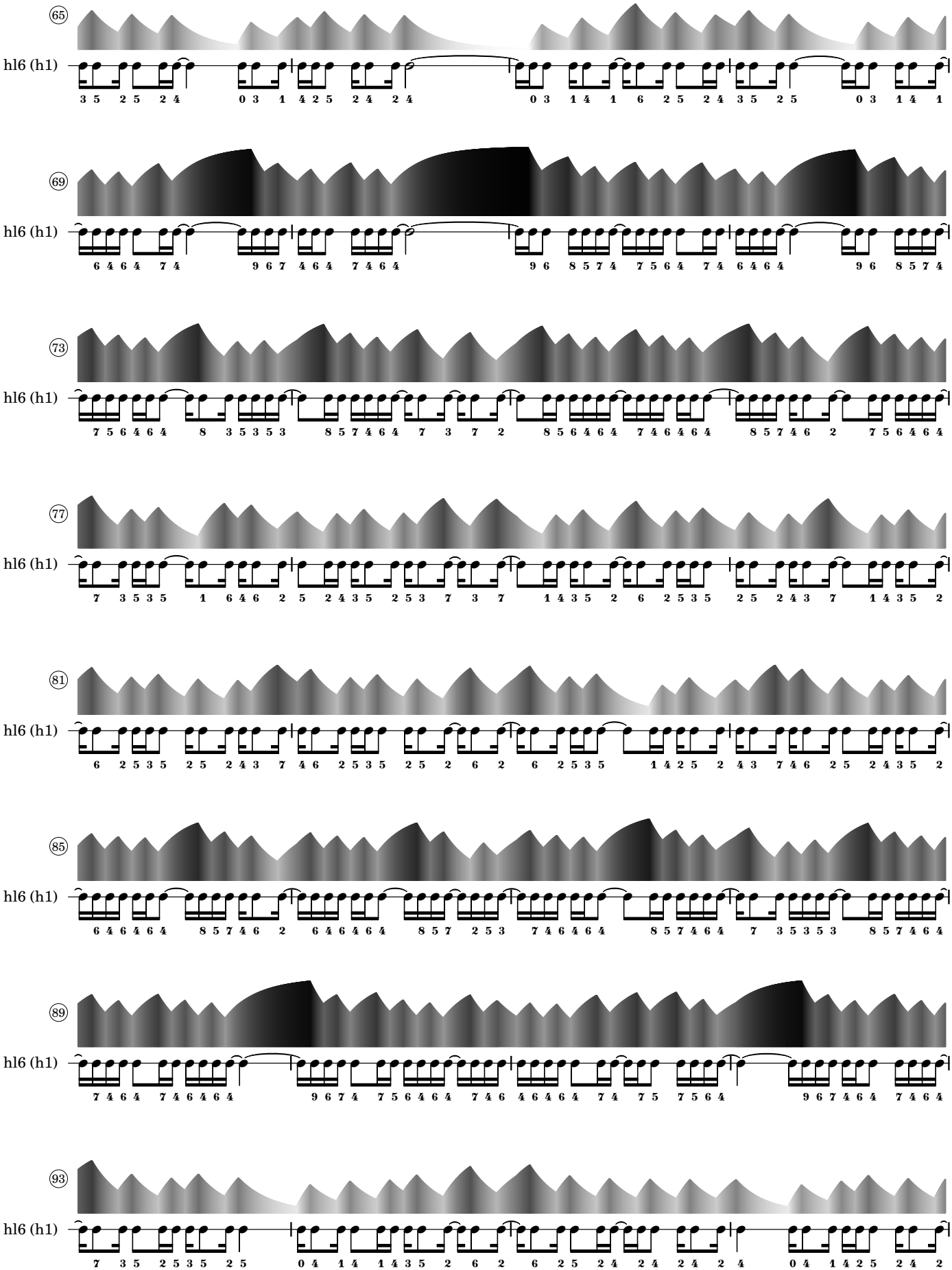
0111211

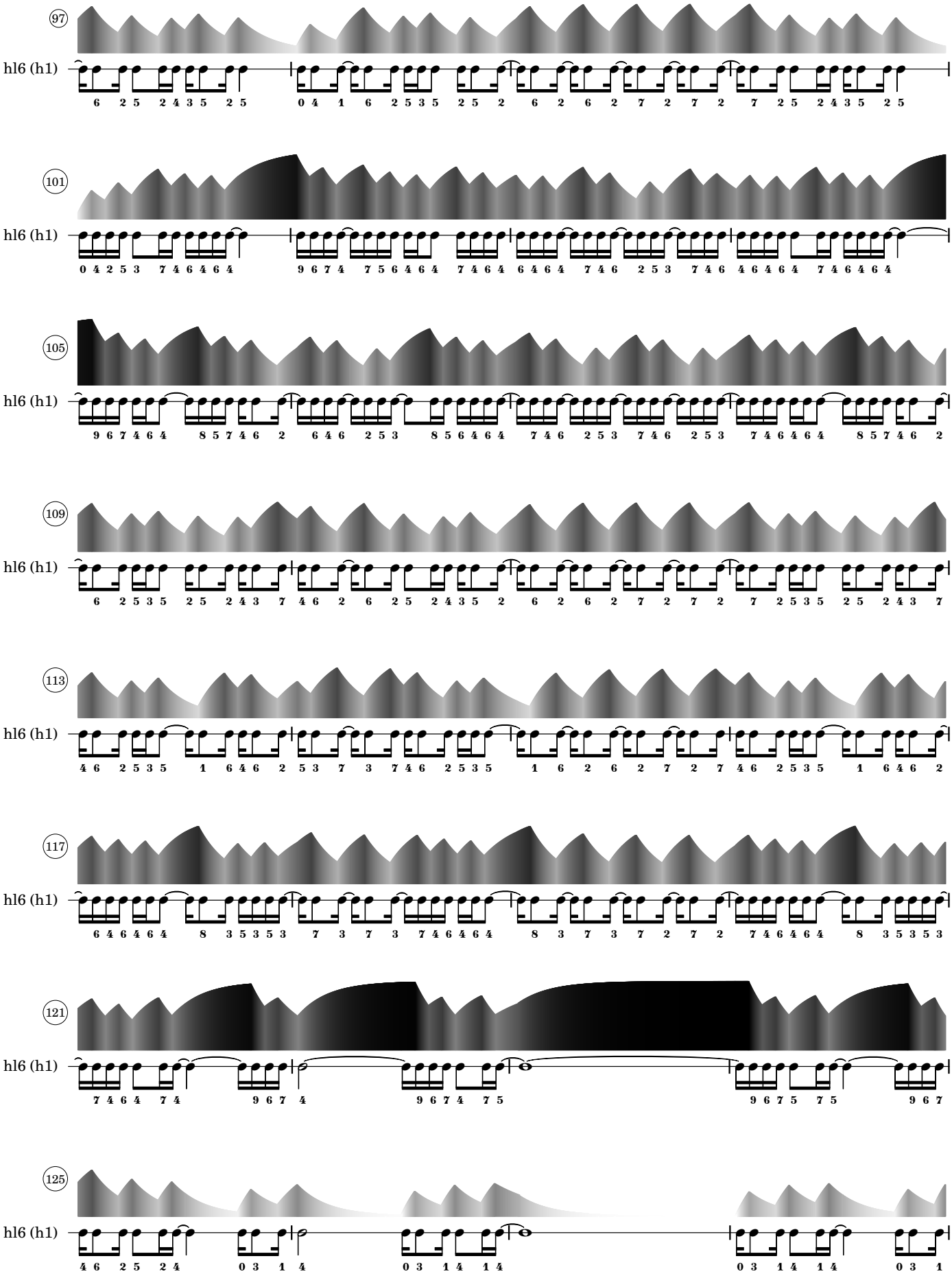
michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 6 - 0111211

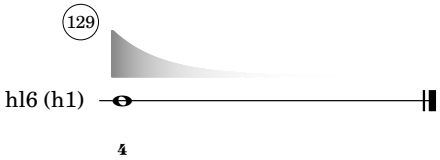



33
hl6 (h1)
7 2 5 2 4 3 5 2 5 0 4 1 4 1 4 3 5 2 6 2 6 2 5 2 4 3 5 2 5 0 4 1 4 1 4 2 4 2 4 2
37
hl6 (h1)
6 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 6 4 6 4 7 4 6 4 6 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 7 5 7 5 6 4
41
hl6 (h1)
7 4 6 4 6 4 8 5 7 4 6 2 6 4 6 4 6 4 8 5 7 2 5 3 7 4 6 4 6 4 8 5 7 4 6 2 6 4 6 4 6 4 8 5 7 4 6 4
45
hl6 (h1)
7 3 5 3 5 2 5 2 4 3 7 4 6 2 5 3 5 2 5 2 6 2 6 2 5 3 5 2 5 2 4 3 7 4 6 2 5 3 5 1 4 2 5 2
49
hl6 (h1)
6 2 5 3 5 1 6 4 6 2 5 2 4 3 5 2 5 3 7 3 7 4 6 2 5 3 5 1 6 4 6 2 5 2 4 3 5 2 6 2 5 3 5
53
hl6 (h1)
2 5 3 5 3 5 3 8 3 5 3 5 3 8 5 6 4 6 4 7 3 7 3 7 4 6 4 6 4 8 3 5 3 5 3 8 5 7 4 6 4 7 4 6 4 6 4
57
hl6 (h1)
8 5 7 4 7 5 9 6 7 4 6 4 7 4 6 4 9 6 7 5 7 5 9 6 7 4 6 4 7 4 6 4 7 4 6 4 7 4
61
hl6 (h1)
7 3 5 2 5 0 3 1 4 2 5 2 4 2 4 0 3 1 4 1 4 0 3 1 4 2 5 2 4 2 6 2 5 2 4


65
hl6 (h1)
3 5 2 5 2 4 0 3 1 4 2 5 2 4 2 4 0 3 1 4 1 6 2 5 2 4 3 5 2 5 0 3 1 4 1
69
hl6 (h1)
6 4 6 4 7 4 9 6 7 4 6 4 7 4 6 4 9 6 8 5 7 4 7 5 6 4 7 4 6 4 6 4 9 6 8 5 7 4
73
hl6 (h1)
7 5 6 4 6 4 8 3 5 3 5 3 8 5 7 4 6 4 7 3 7 2 8 5 6 4 6 4 7 4 6 4 6 4 8 5 7 4 6 2 7 5 6 4 6 4
77
hl6 (h1)
7 3 5 3 5 1 6 4 6 2 5 2 4 3 5 2 5 3 7 3 7 1 4 3 5 2 6 2 5 3 5 2 5 2 4 3 7 1 4 3 5 2
81
hl6 (h1)
6 2 5 3 5 2 5 2 4 3 7 4 6 2 5 3 5 2 5 2 6 2 6 2 5 3 5 1 4 2 5 2 4 3 7 4 6 2 5 2 4 3 5 2
85
hl6 (h1)
6 4 6 4 6 4 8 5 7 4 6 2 6 4 6 4 6 4 8 5 7 2 5 3 7 4 6 4 6 4 8 5 7 4 6 4 7 3 5 3 5 3 8 5 7 4 6 4
89
hl6 (h1)
7 4 6 4 7 4 6 4 6 4 9 6 7 4 7 5 6 4 6 4 7 4 6 4 6 4 6 4 7 4 7 5 7 5 6 4 9 6 7 4 6 4 7 4 6 4
93
hl6 (h1)
7 3 5 2 5 3 5 2 5 0 4 1 4 1 4 3 5 2 6 2 6 2 5 2 4 2 4 2 4 2 4 0 4 1 4 2 5 2 4 2


97
hl6 (h1)
6 2 5 2 4 3 5 2 5 0 4 1 6 2 5 3 5 2 5 2 6 2 6 2 7 2 7 2 7 2 5 2 4 3 5 2 5
101
hl6 (h1)
0 4 2 5 3 7 4 6 4 6 4 9 6 7 4 7 5 6 4 6 4 7 4 6 4 6 4 6 4 6 4 7 4 6 2 5 3 7 4 6 4 6 4 6 4 7 4 6 4 6 4
105
hl6 (h1)
9 6 7 4 6 4 8 5 7 4 6 2 6 4 6 2 5 3 8 5 6 4 6 4 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 4 6 4 8 5 7 4 6 2
109
hl6 (h1)
6 2 5 3 5 2 5 2 4 3 7 4 6 2 6 2 5 2 4 3 5 2 6 2 6 2 7 2 7 2 7 2 5 3 5 2 5 2 4 3 7
113
hl6 (h1)
4 6 2 5 3 5 1 6 4 6 2 5 3 7 3 7 4 6 2 5 3 5 1 6 2 6 2 7 2 7 4 6 2 5 3 5 1 6 4 6 2
117
hl6 (h1)
6 4 6 4 6 4 8 3 5 3 5 3 7 3 7 3 7 4 6 4 6 4 8 3 7 3 7 2 7 2 7 4 6 4 6 4 8 3 5 3 5 3
121
hl6 (h1)
7 4 6 4 7 4 9 6 7 4 9 6 7 4 7 5 9 6 7 5 7 5 9 6 7
125
hl6 (h1)
4 6 2 5 2 4 0 3 1 4 0 3 1 4 1 4 0 3 1 4 1 4 0 3 1

(part - hl 6 - 0111211)

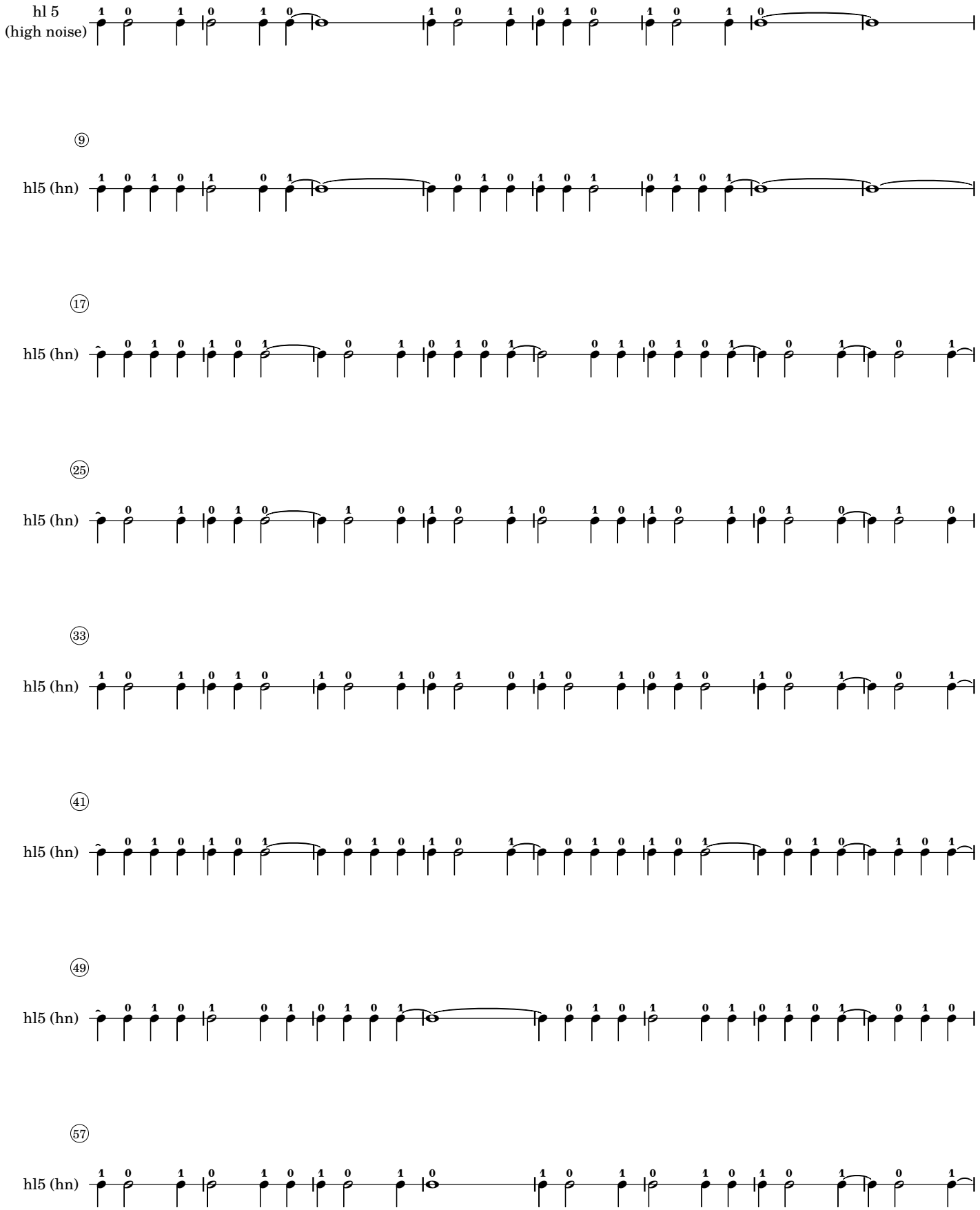


a lot of tiles (trivial scan)

0111211

michael winter
(cdmx, mx and nashville, usa; 2018)

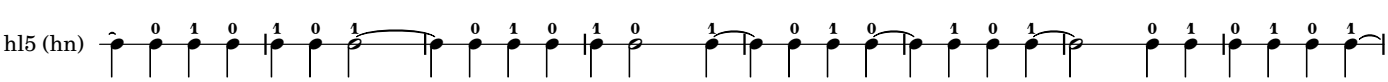
part - hl 5 - 0111211



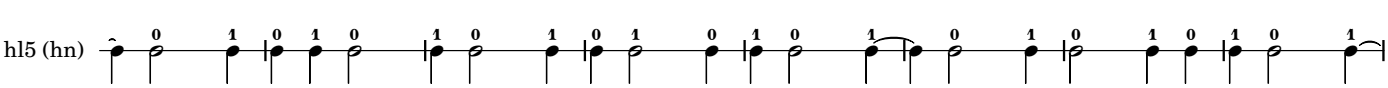
65

73

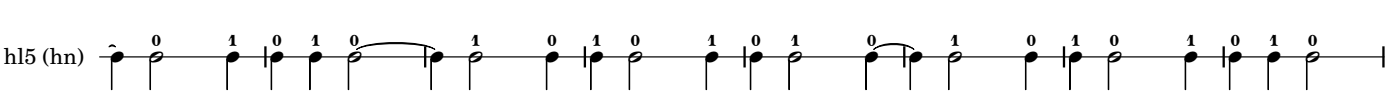
81



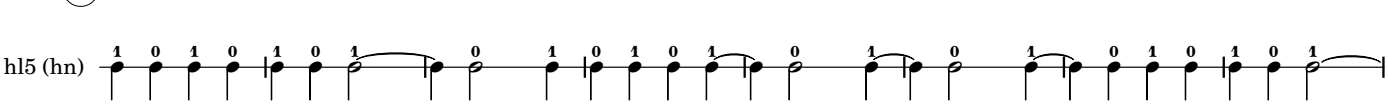
89



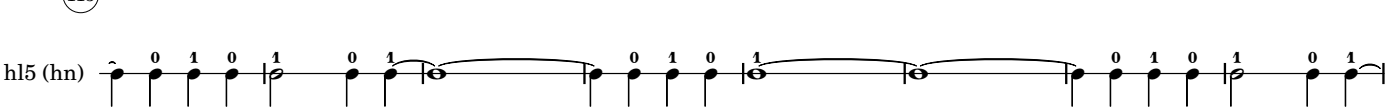
97



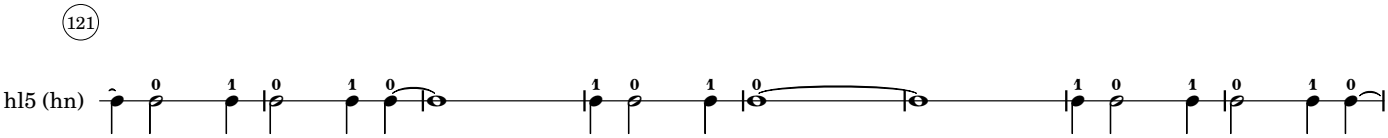
105



113



121



(part - hl 5 - 0111211)

129

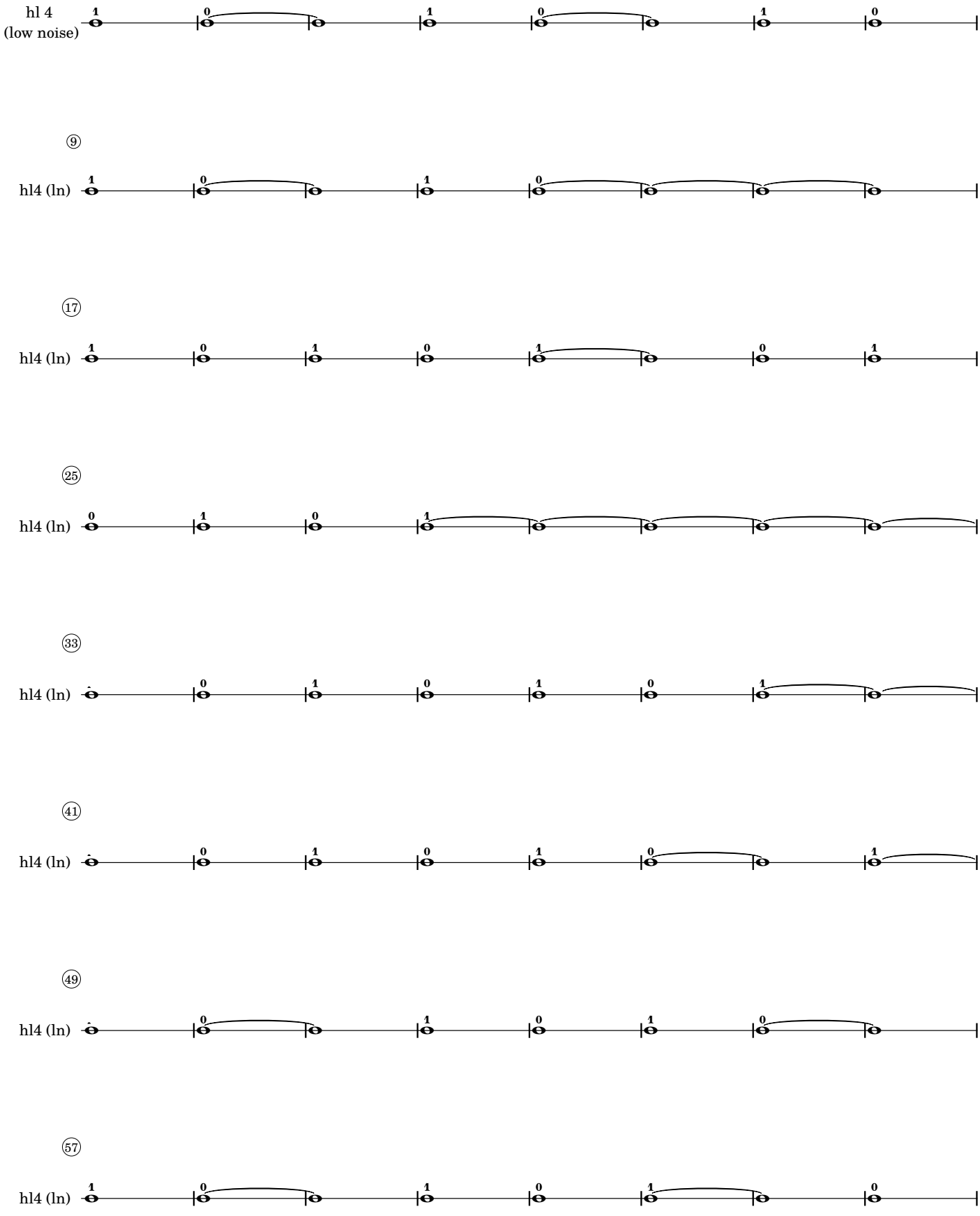
hl5 (hn)

a lot of tiles (trivial scan)

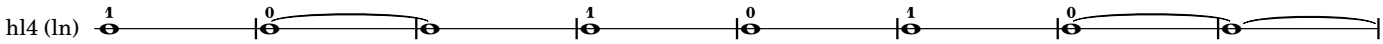
0111211

michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 4 - 0111211

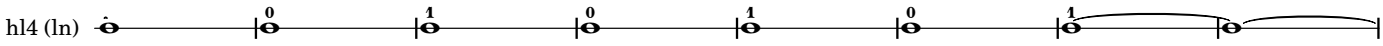


65

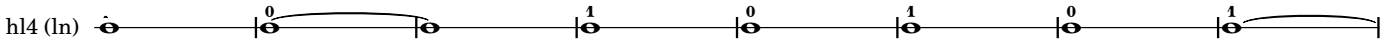


73

81

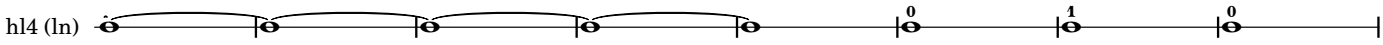


89

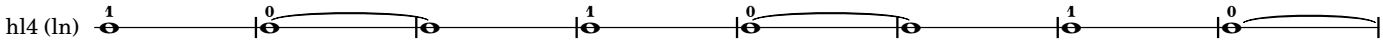


97

105



113



121

(part - hl 4 - 0111211)

129

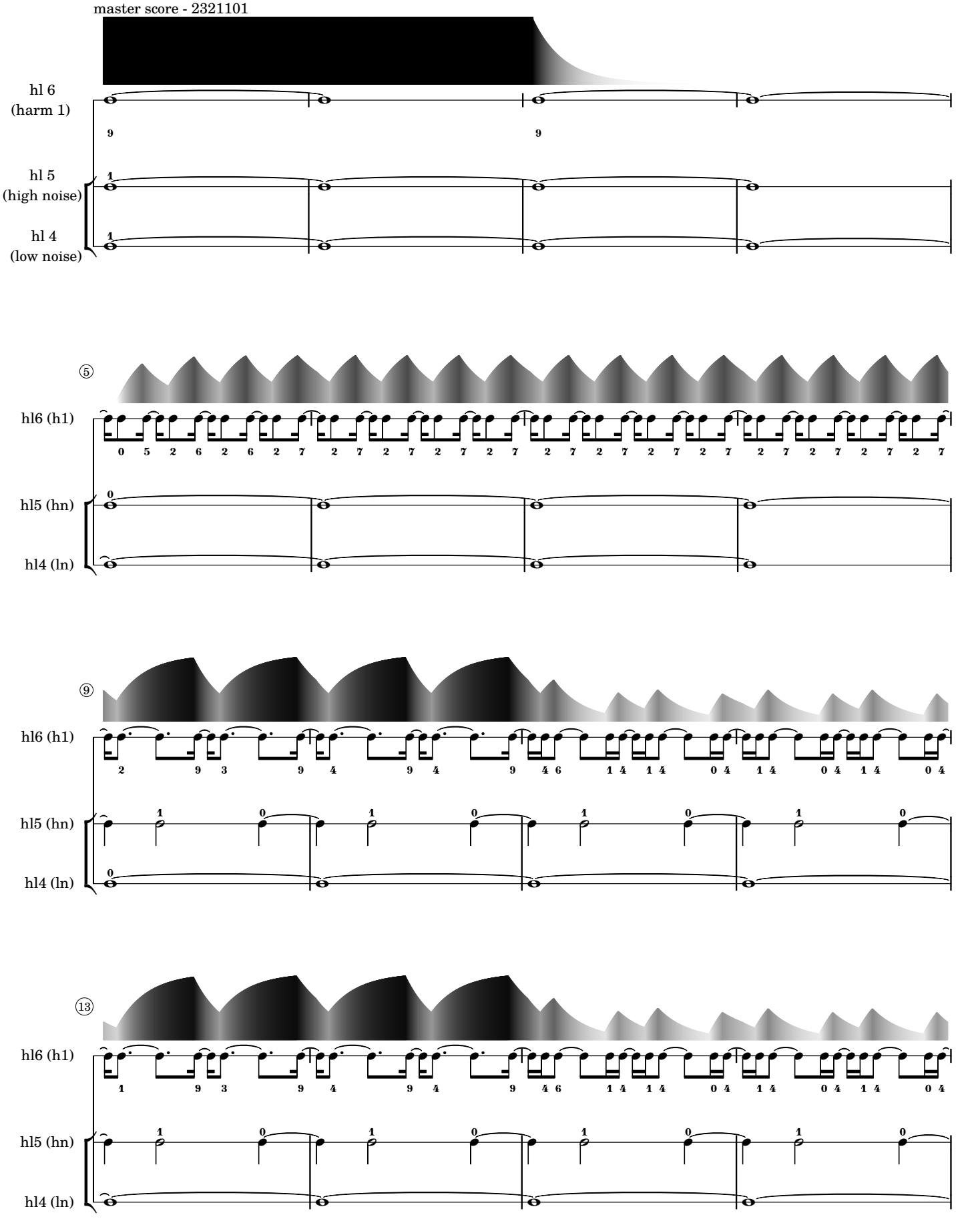
hl4 (ln)

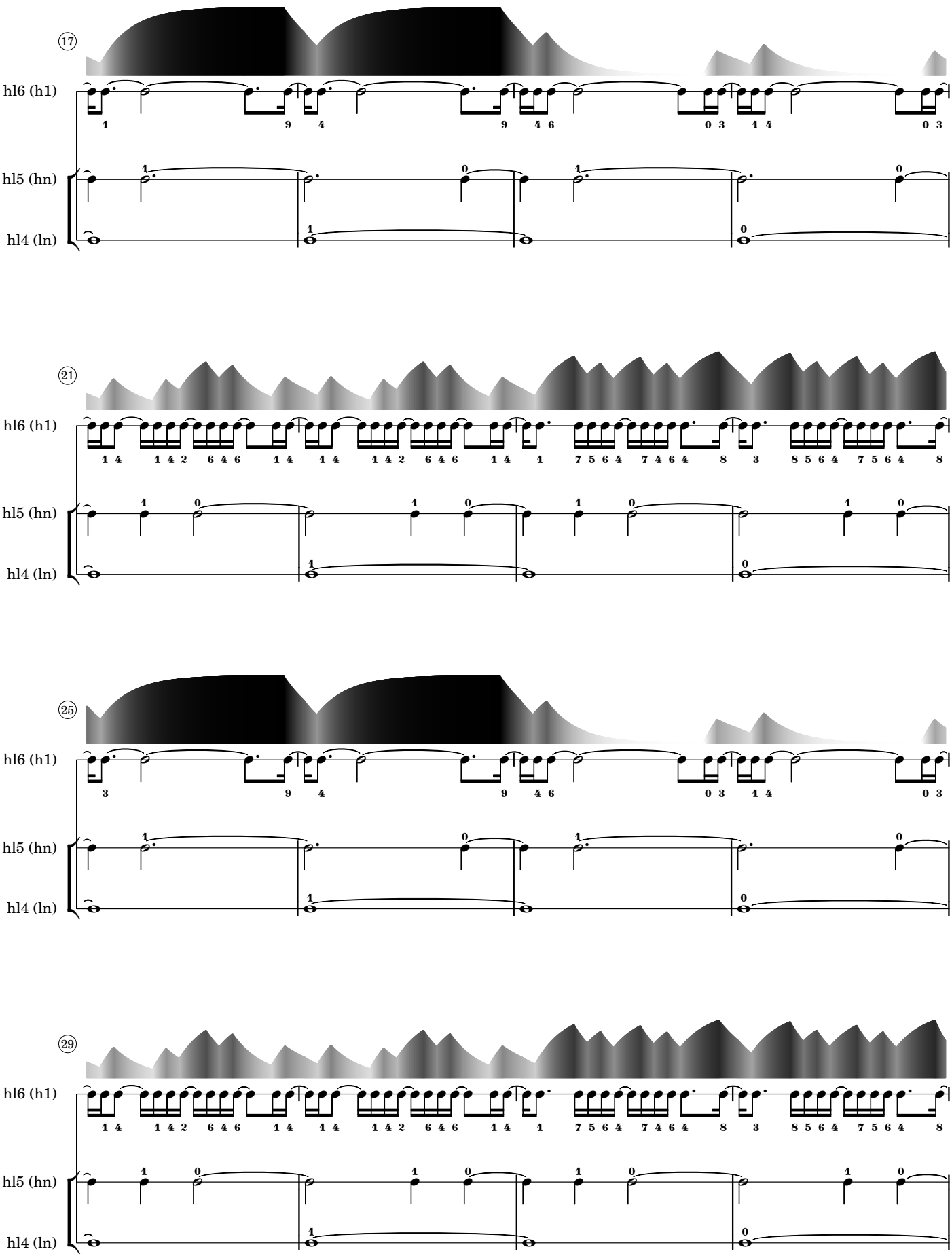
a lot of tiles (trivial scan)

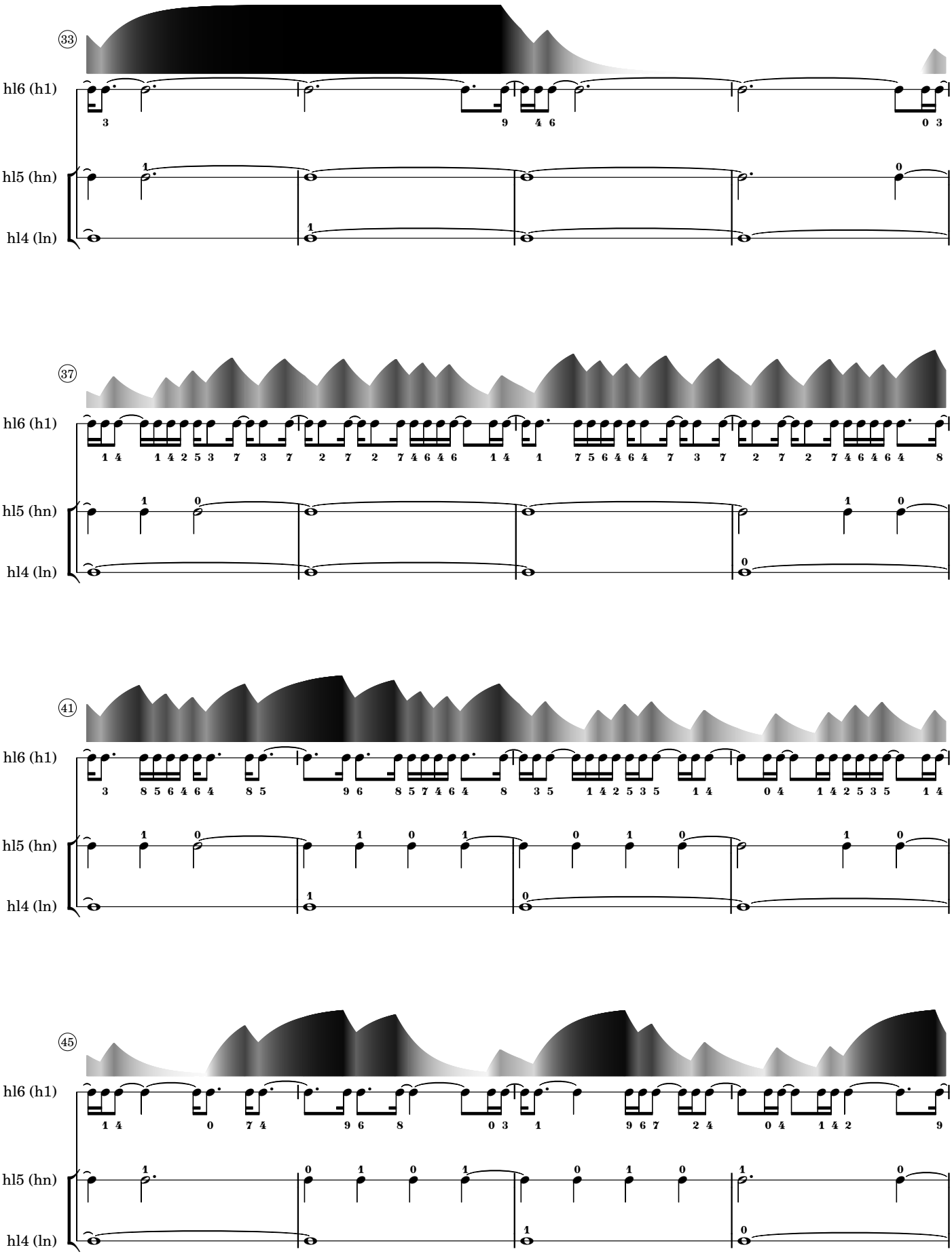
2321101

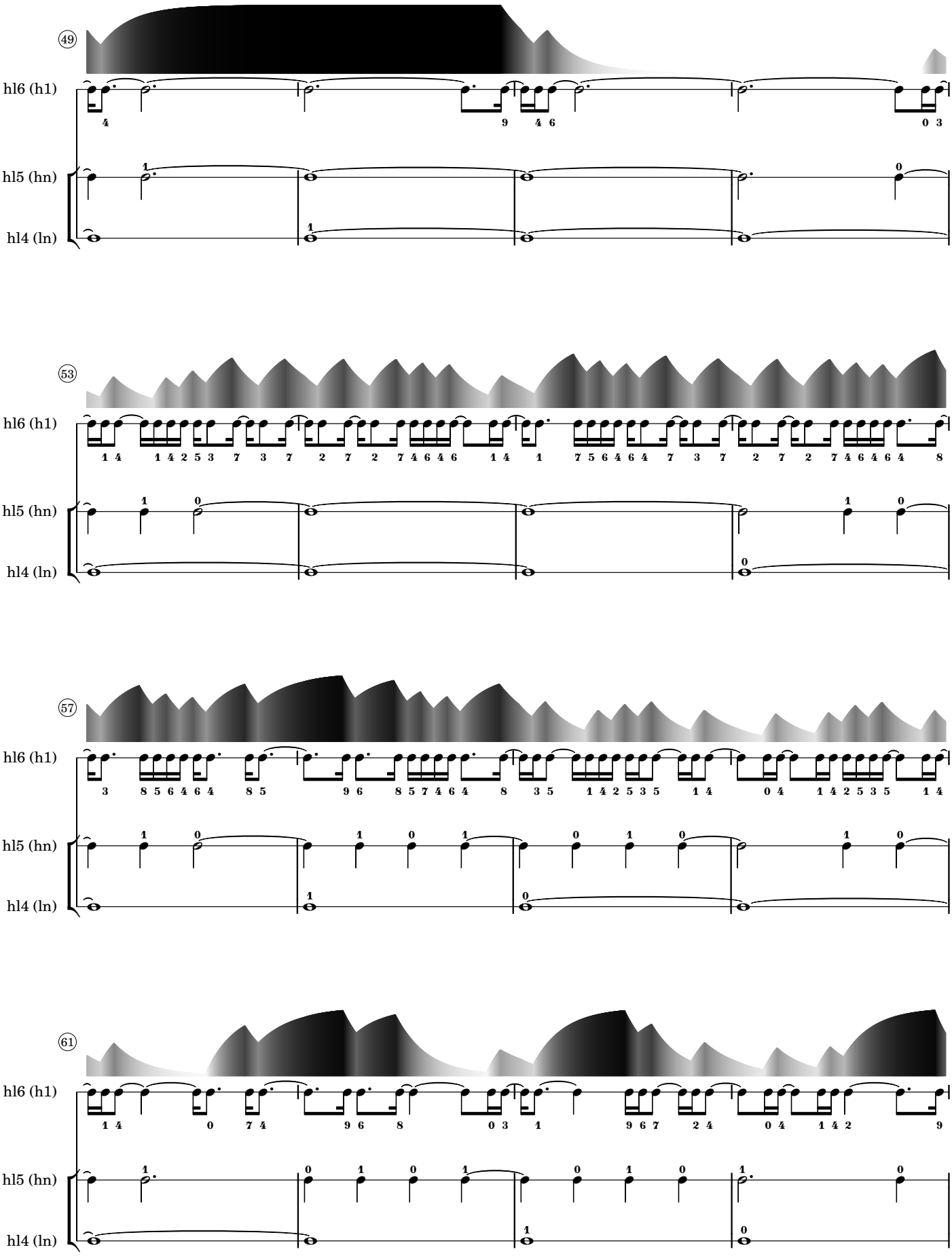
michael winter
(cdmx, mx and nashville, usa; 2018)

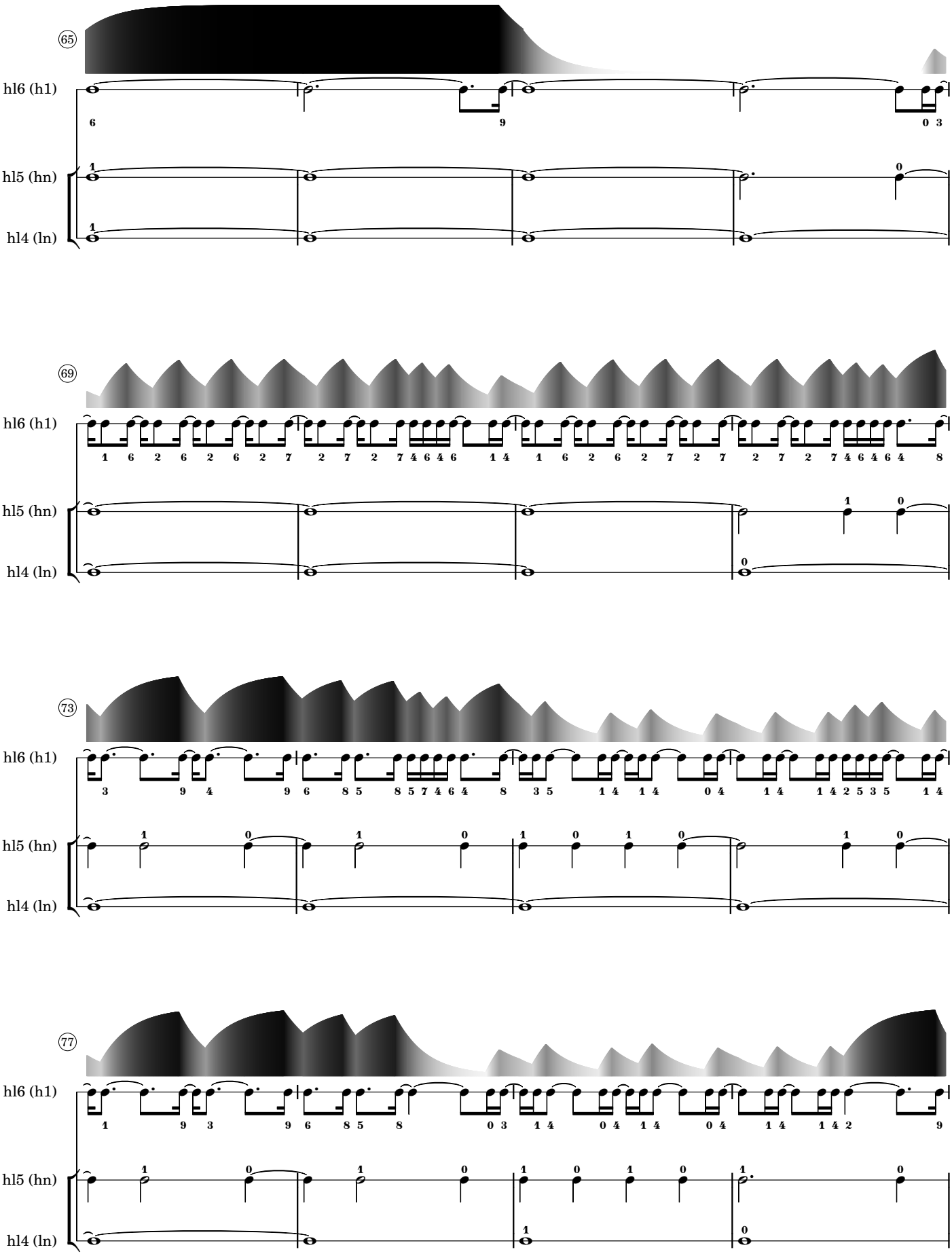
master score - 2321101

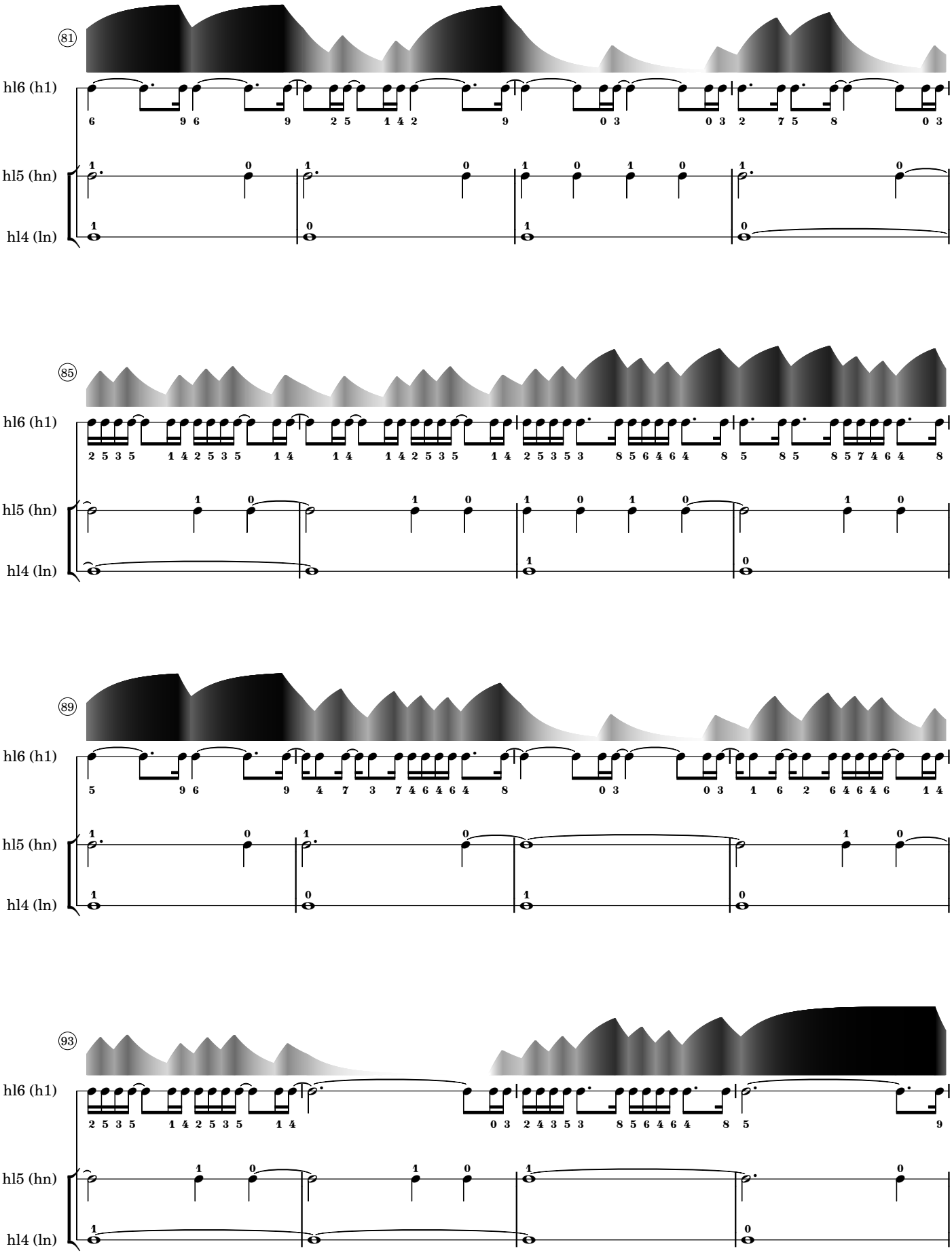


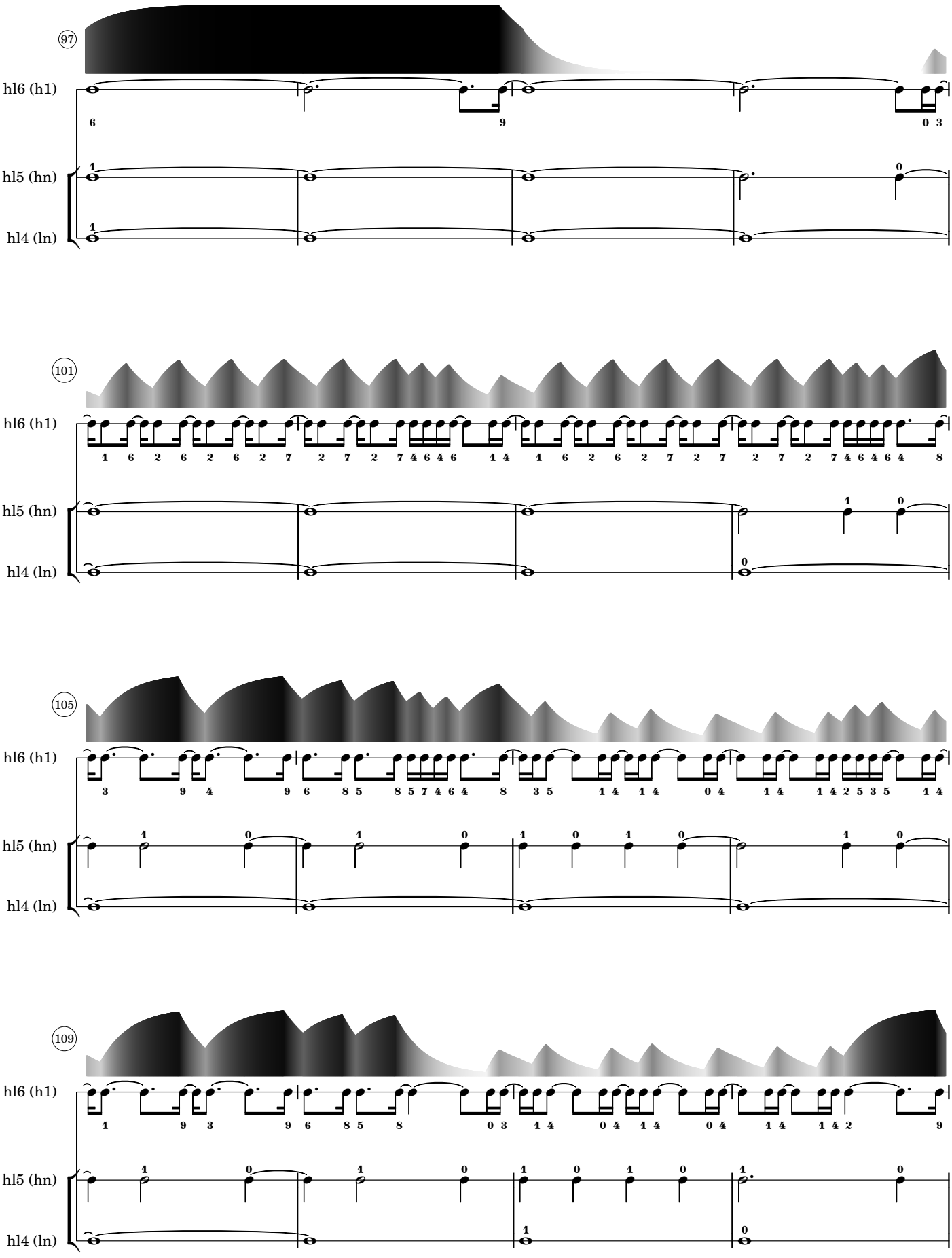

17
hl6 (h1)
hl5 (hn)
hl4 (ln)
21
hl6 (h1)
hl5 (hn)
hl4 (ln)
25
hl6 (h1)
hl5 (hn)
hl4 (ln)
29
hl6 (h1)
hl5 (hn)
hl4 (ln)

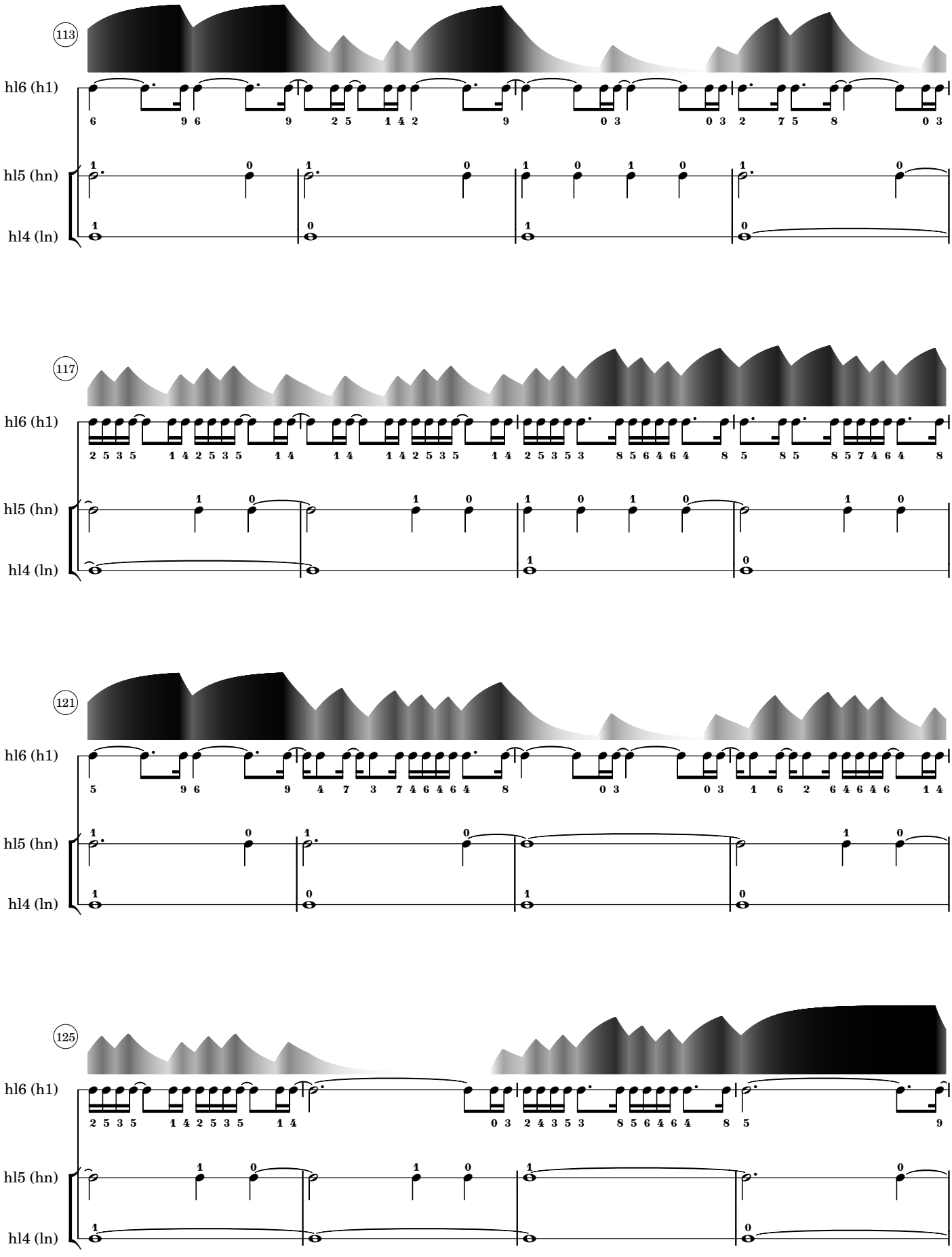

33
hl6 (h1)
hl5 (hn)
hl4 (ln)
37
hl6 (h1)
hl5 (hn)
hl4 (ln)
41
hl6 (h1)
hl5 (hn)
hl4 (ln)
45
hl6 (h1)
hl5 (hn)
hl4 (ln)

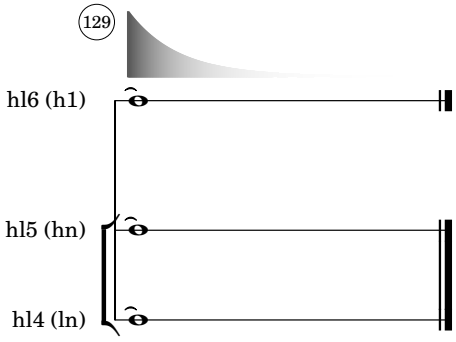

49
hl6 (h1)
hl5 (hn)
hl4 (ln)
53
hl6 (h1)
hl5 (hn)
hl4 (ln)
57
hl6 (h1)
hl5 (hn)
hl4 (ln)
61
hl6 (h1)
hl5 (hn)
hl4 (ln)


65
hl6 (h1)
hl5 (hn)
hl4 (ln)
69
hl6 (h1)
hl5 (hn)
hl4 (ln)
73
hl6 (h1)
hl5 (hn)
hl4 (ln)
77
hl6 (h1)
hl5 (hn)
hl4 (ln)


81
hl6 (h1)
hl5 (hn)
hl4 (ln)
85
hl6 (h1)
hl5 (hn)
hl4 (ln)
89
hl6 (h1)
hl5 (hn)
hl4 (ln)
93
hl6 (h1)
hl5 (hn)
hl4 (ln)


97
hl6 (h1)
hl5 (hn)
hl4 (ln)
101
hl6 (h1)
hl5 (hn)
hl4 (ln)
105
hl6 (h1)
hl5 (hn)
hl4 (ln)
109
hl6 (h1)
hl5 (hn)
hl4 (ln)


113
hl6 (h1)
hl5 (hn)
hl4 (ln)
117
hl6 (h1)
hl5 (hn)
hl4 (ln)
121
hl6 (h1)
hl5 (hn)
hl4 (ln)
125
hl6 (h1)
hl5 (hn)
hl4 (ln)


129
hl6 (h1)
hl5 (hn)
hl4 (ln)

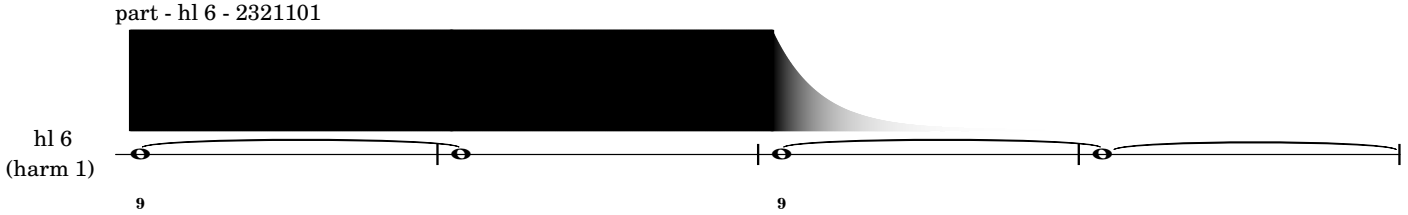
a lot of tiles (trivial scan)

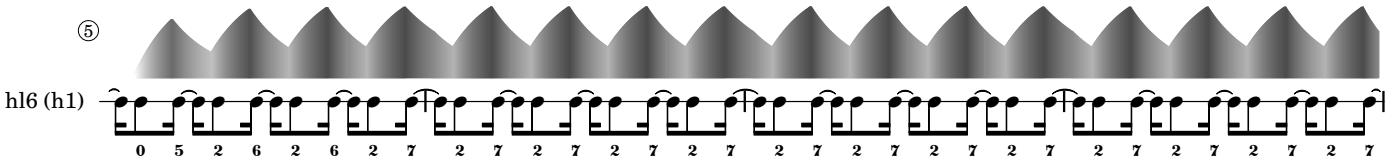
2321101

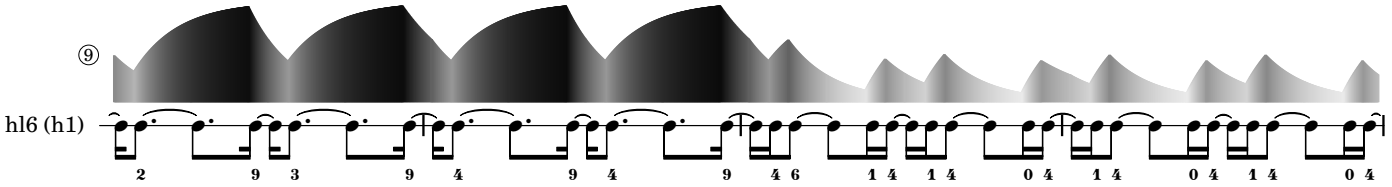
michael winter

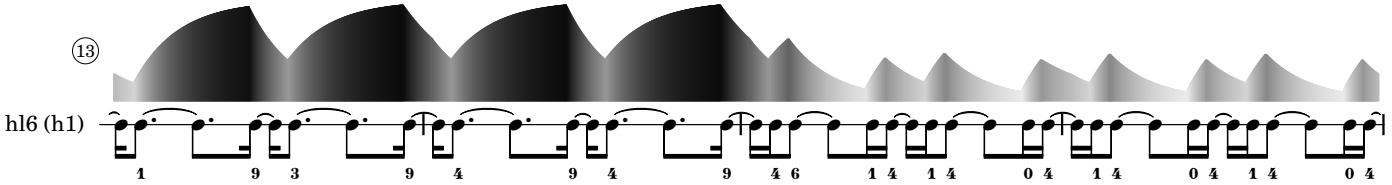
(cdmx, mx and nashville, usa; 2018)

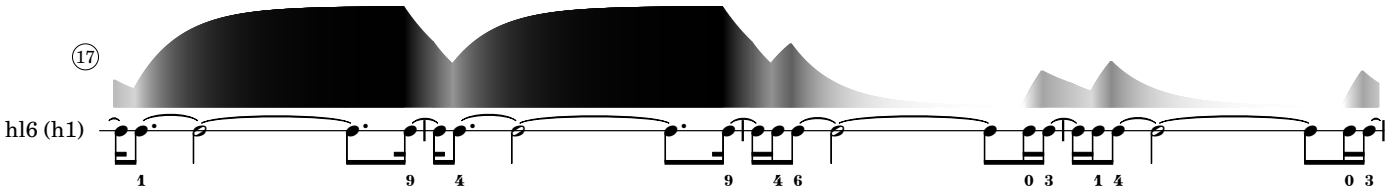
part - hl 6 - 2321101

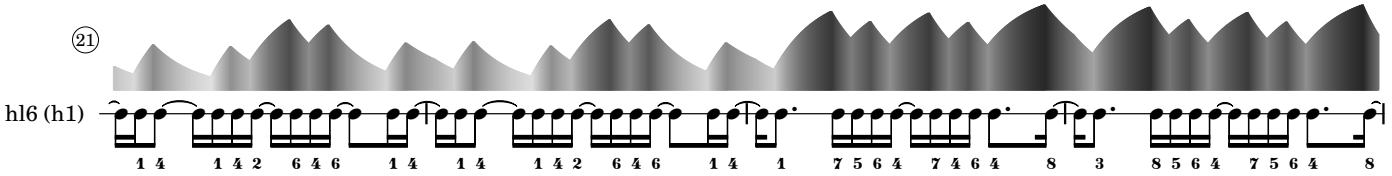


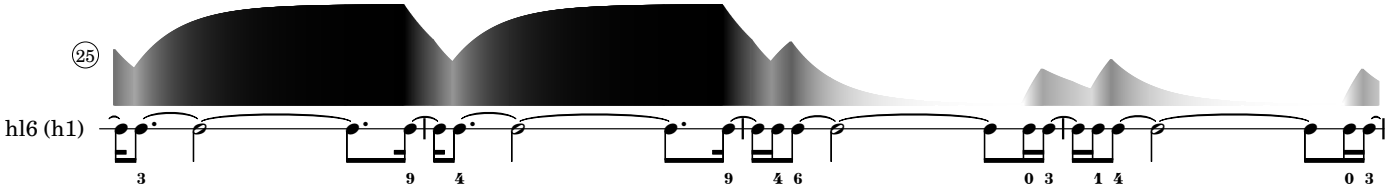


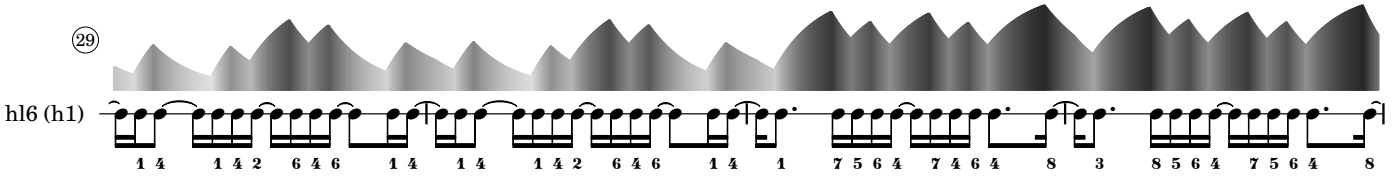


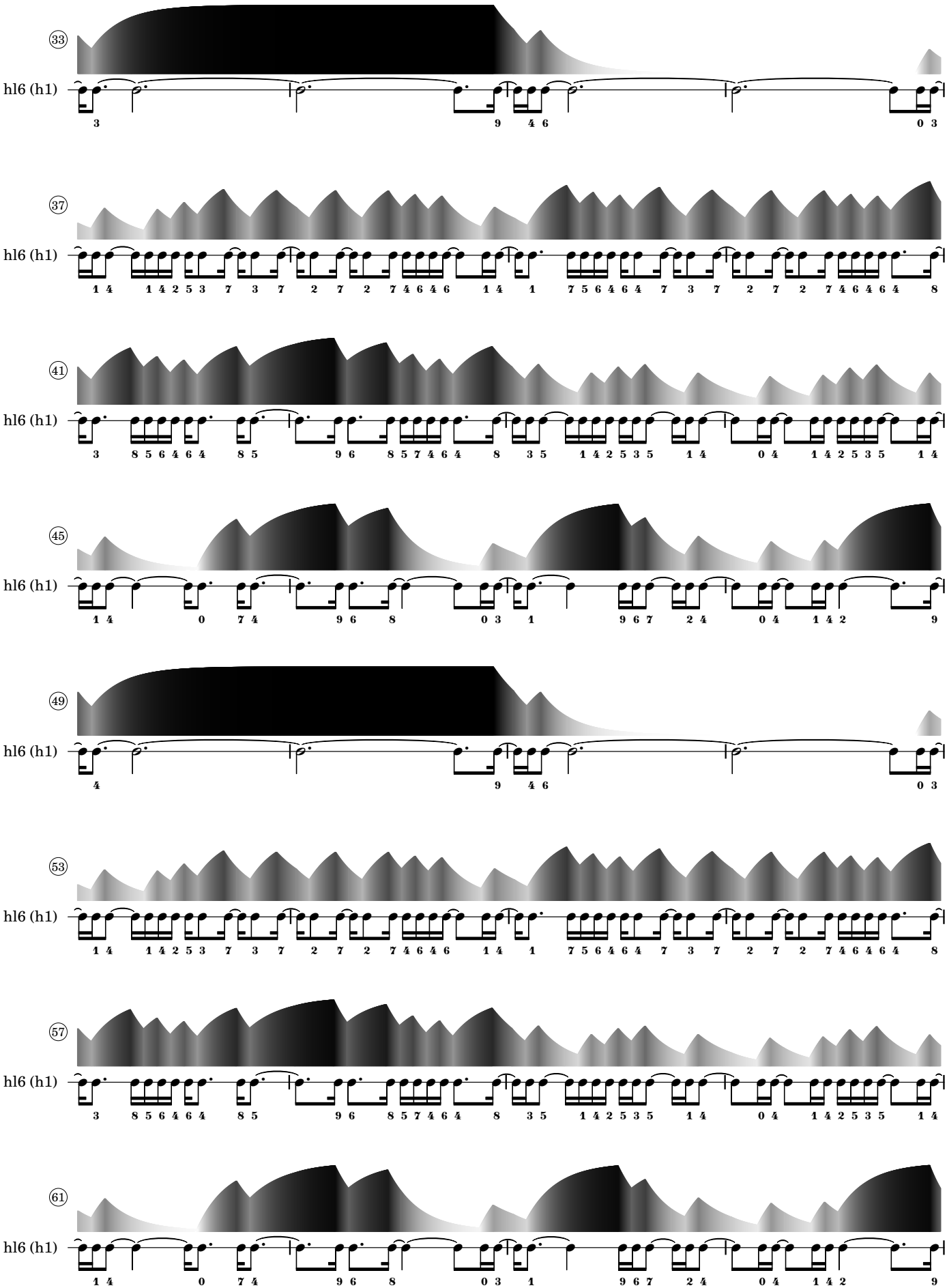


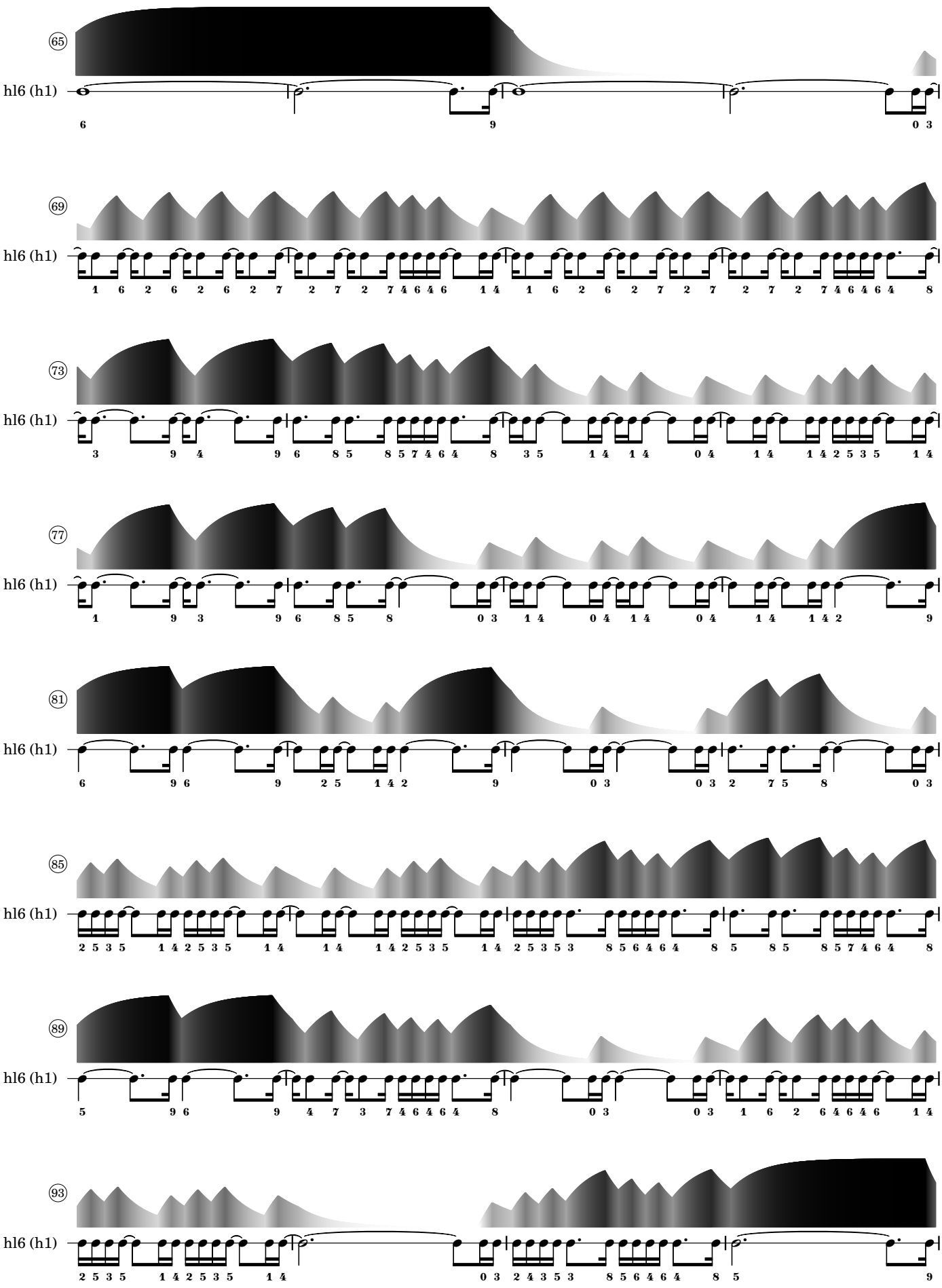


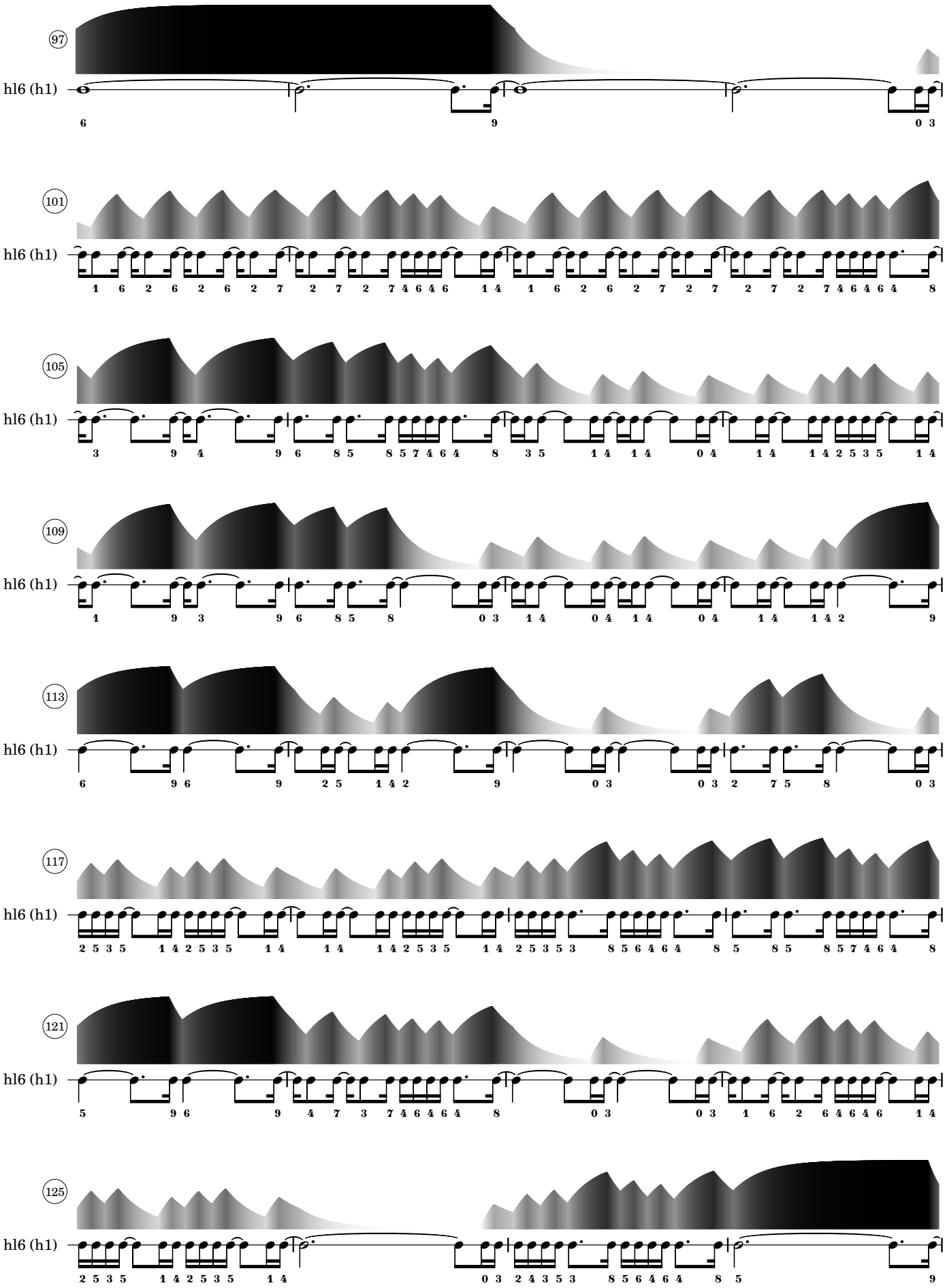




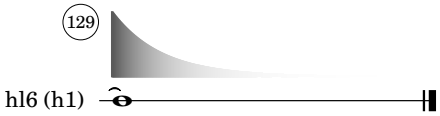



33
hl6 (h1)
3 9 4 6 0 3
37
hl6 (h1)
1 4 1 4 2 5 3 7 3 7 2 7 2 7 4 6 4 6 1 4 1 7 5 6 4 6 4 7 3 7 2 7 2 7 4 6 4 6 4 8
41
hl6 (h1)
3 8 5 6 4 6 4 8 5 9 6 8 5 7 4 6 4 8 3 5 1 4 2 5 3 5 1 4 0 4 1 4 2 5 3 5 1 4
45
hl6 (h1)
1 4 0 7 4 9 6 8 0 3 1 9 6 7 2 4 0 4 1 4 2 9
49
hl6 (h1)
4 9 4 6 0 3
53
hl6 (h1)
1 4 1 4 2 5 3 7 3 7 2 7 2 7 4 6 4 6 1 4 1 7 5 6 4 6 4 7 3 7 2 7 2 7 4 6 4 6 4 8
57
hl6 (h1)
3 8 5 6 4 6 4 8 5 9 6 8 5 7 4 6 4 8 3 5 1 4 2 5 3 5 1 4 0 4 1 4 2 5 3 5 1 4
61
hl6 (h1)
1 4 0 7 4 9 6 8 0 3 1 9 6 7 2 4 0 4 1 4 2 9


65
hl6 (h1)
6 9 0 3
69
hl6 (h1)
1 6 2 6 2 6 2 7 2 7 2 7 4 6 4 6 1 4 1 6 2 6 2 7 2 7 2 7 2 7 4 6 4 6 4 8
73
hl6 (h1)
3 9 4 9 6 8 5 8 5 7 4 6 4 8 3 5 1 4 1 4 0 4 1 4 1 4 2 5 3 5 1 4
77
hl6 (h1)
1 9 3 9 6 8 5 8 0 3 1 4 0 4 1 4 0 4 1 4 1 4 2 9
81
hl6 (h1)
6 9 6 9 2 5 1 4 2 9 0 3 0 3 2 7 5 8 0 3
85
hl6 (h1)
2 5 3 5 1 4 2 5 3 5 1 4 1 4 1 4 2 5 3 5 1 4 2 5 3 5 3 8 5 6 4 6 4 8 5 8 5 8 5 7 4 6 4 8
89
hl6 (h1)
5 9 6 9 4 7 3 7 4 6 4 6 4 8 0 3 0 3 1 6 2 6 4 6 4 6 1 4
93
hl6 (h1)
2 5 3 5 1 4 2 5 3 5 1 4 0 3 2 4 3 5 3 8 5 6 4 6 4 8 5 9


97
hl6 (h1)
6 9 0 3
101
hl6 (h1)
1 6 2 6 2 6 2 7 2 7 2 7 4 6 4 6 1 4 1 6 2 6 2 7 2 7 2 7 2 7 4 6 4 6 4 8
105
hl6 (h1)
3 9 4 9 6 8 5 8 5 7 4 6 4 8 3 5 1 4 1 4 0 4 1 4 1 4 2 5 3 5 1 4
109
hl6 (h1)
1 9 3 9 6 8 5 8 0 3 1 4 0 4 1 4 0 4 1 4 1 4 2 9
113
hl6 (h1)
6 9 6 9 2 5 1 4 2 9 0 3 0 3 2 7 5 8 0 3
117
hl6 (h1)
2 5 3 5 1 4 2 5 3 5 1 4 1 4 1 4 2 5 3 5 1 4 2 5 3 5 3 8 5 6 4 6 4 8 5 8 5 8 5 7 4 6 4 8
121
hl6 (h1)
5 9 6 9 4 7 3 7 4 6 4 6 4 8 0 3 0 3 1 6 2 6 4 6 4 6 1 4
125
hl6 (h1)
2 5 3 5 1 4 2 5 3 5 1 4 0 3 2 4 3 5 3 8 5 6 4 6 4 8 5 9

(part - hl 6 - 2321101)

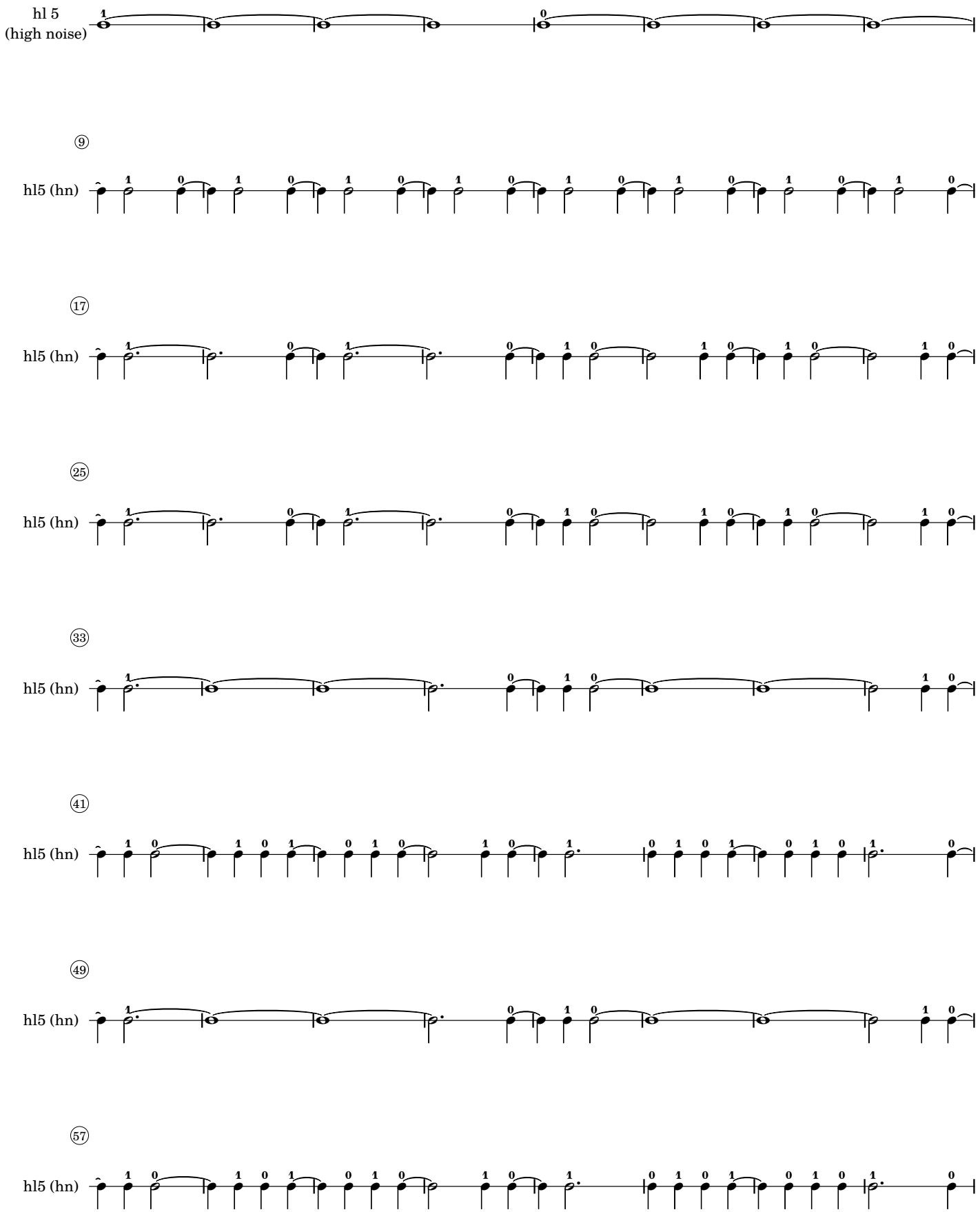


a lot of tiles (trivial scan)

2321101

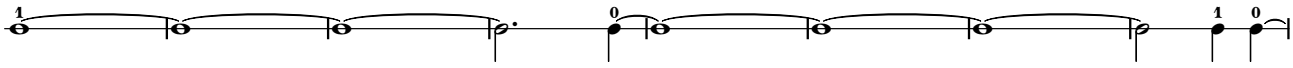
michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 5 - 2321101



65

hl5 (hn)

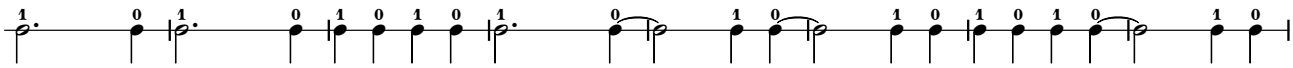


73

hl5 (hn)

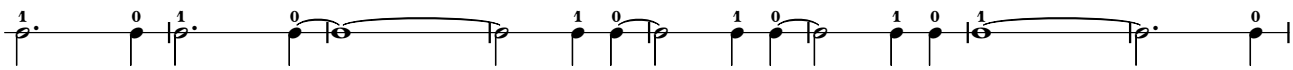
81

hl5 (hn)



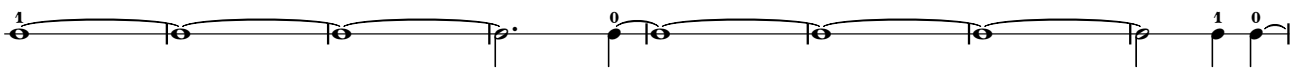
89

hl5 (hn)



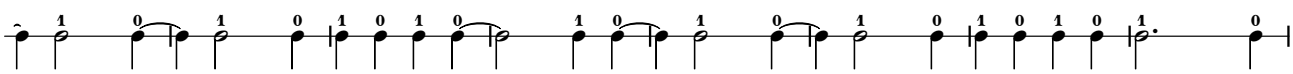
97

hl5 (hn)



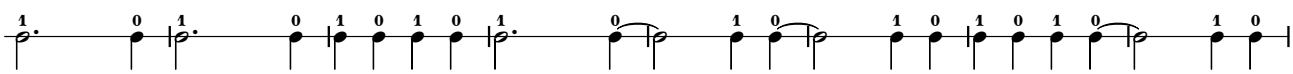
105

hl5 (hn)



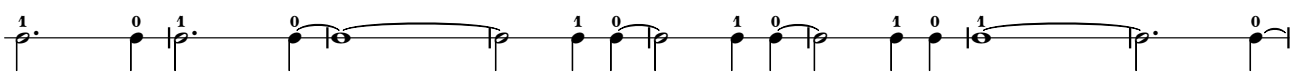
113

hl5 (hn)



121

hl5 (hn)



(part - hl 5 - 2321101)

129

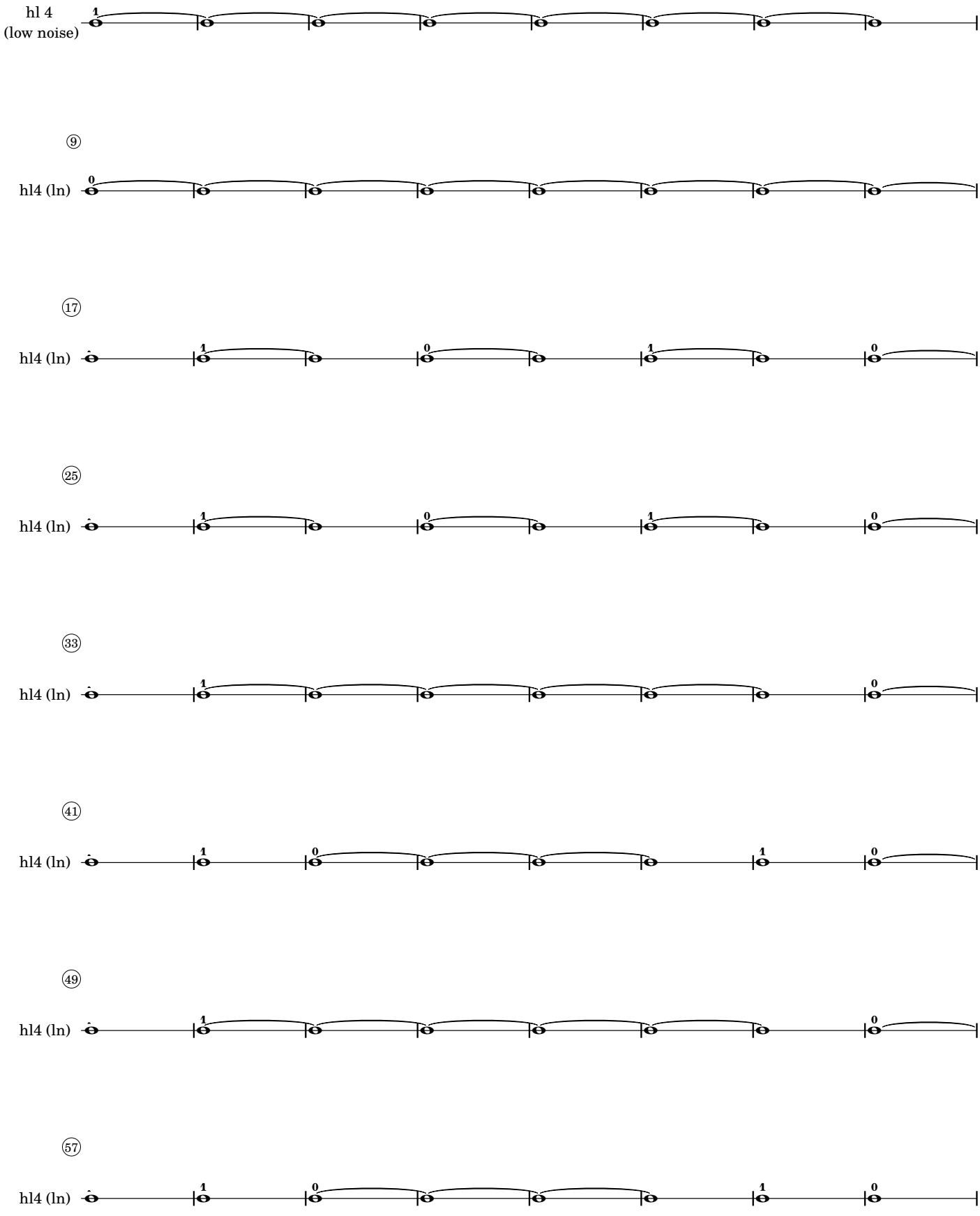
hl5 (hn)

a lot of tiles (trivial scan)

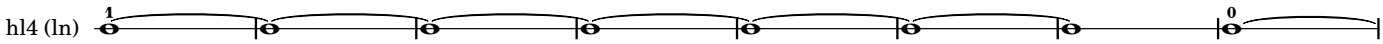
2321101

michael winter
(cdmx, mx and nashville, usa; 2018)

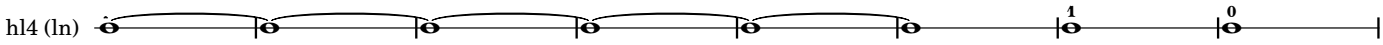
part - hl 4 - 2321101



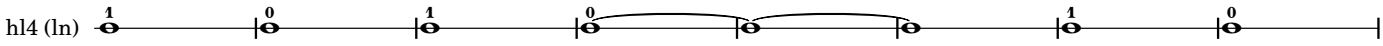
65



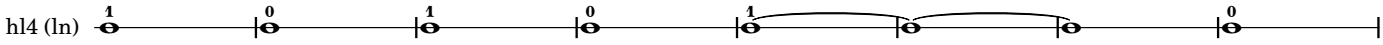
73



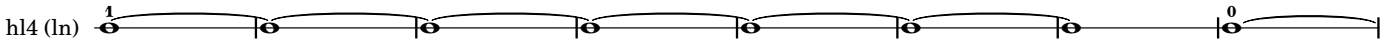
81



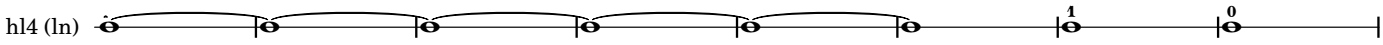
89



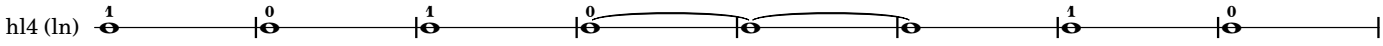
97



105



113



121

(part - hl 4 - 2321101)

129

hl4 (ln)

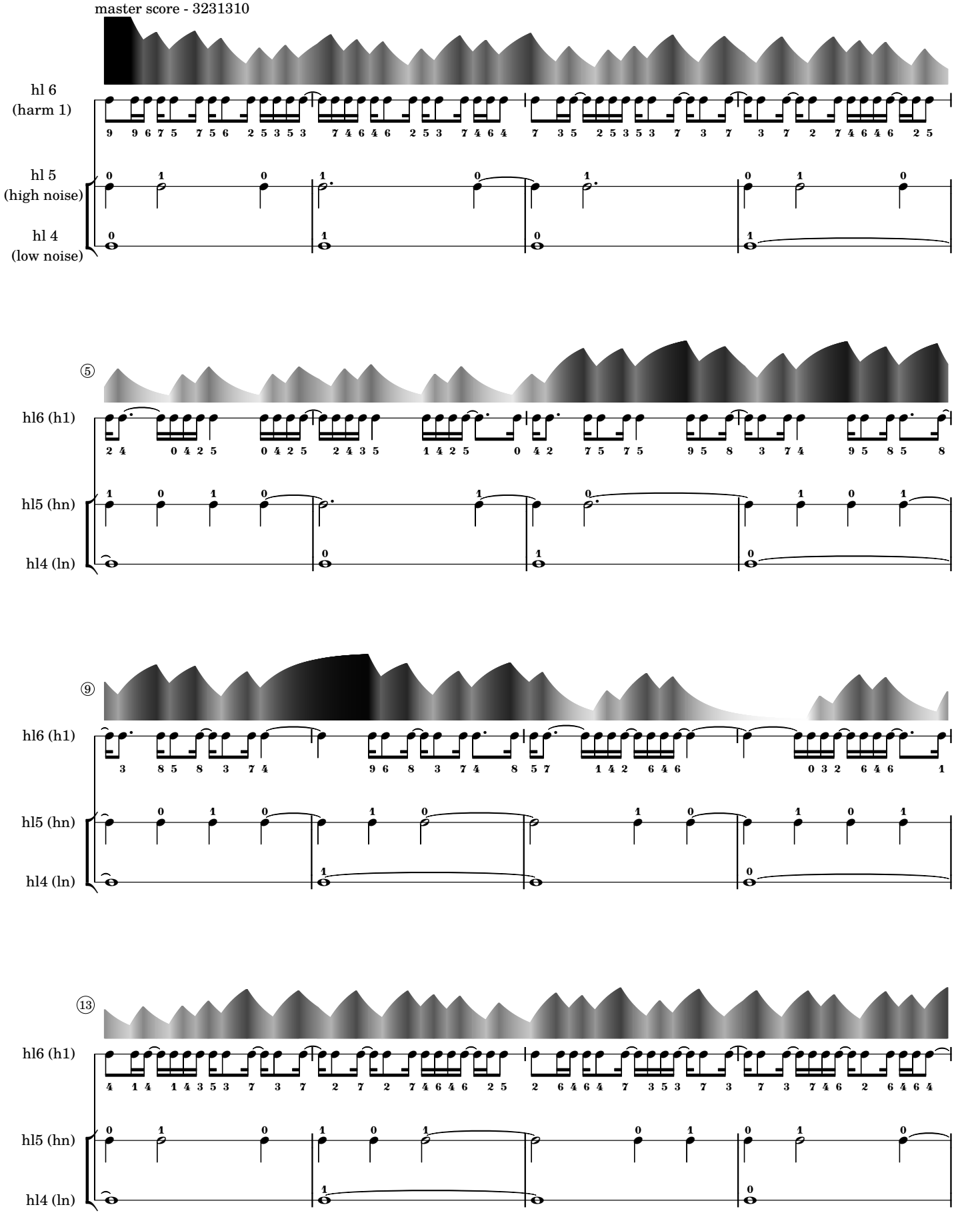
a lot of tiles (trivial scan)

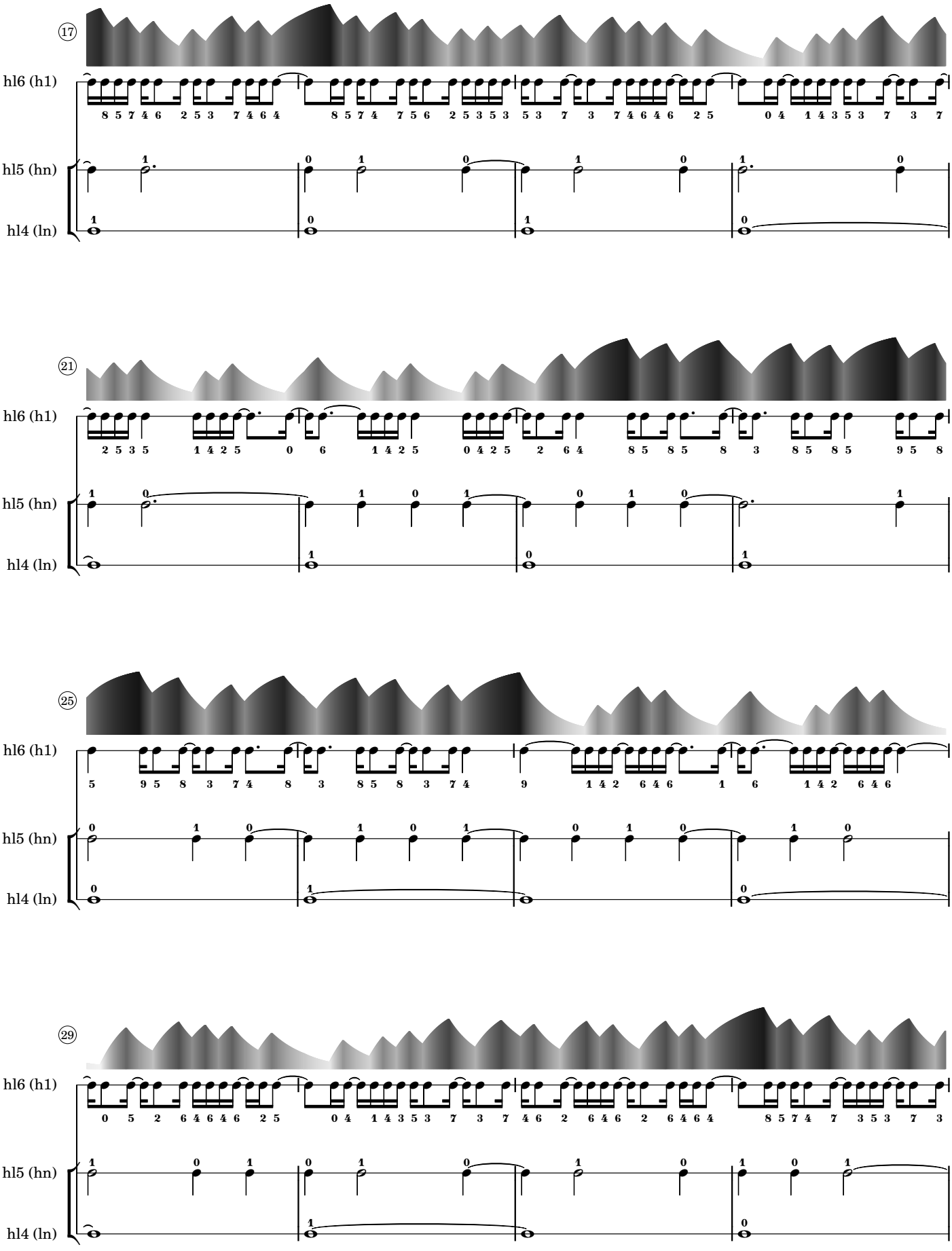
3231310

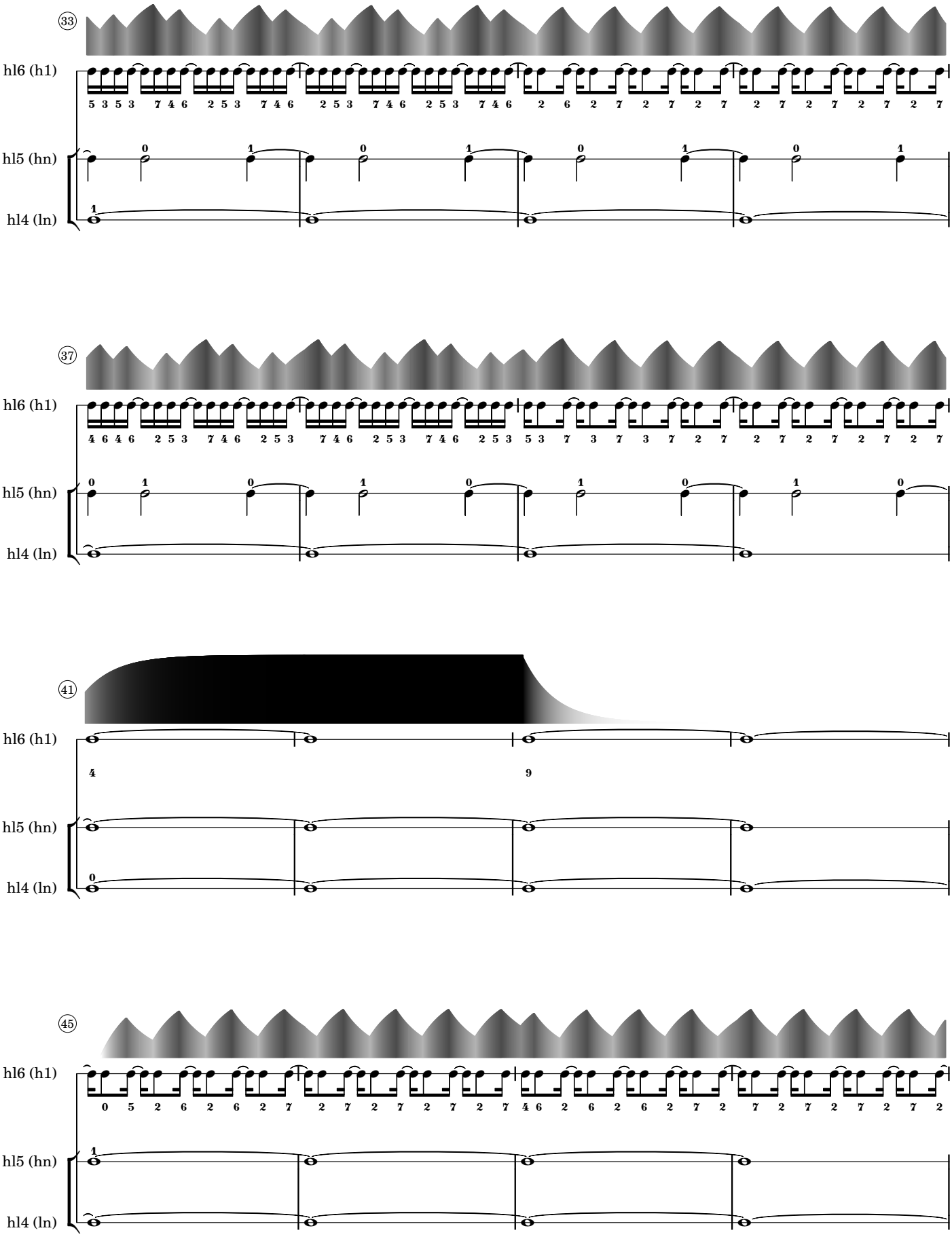
michael winter

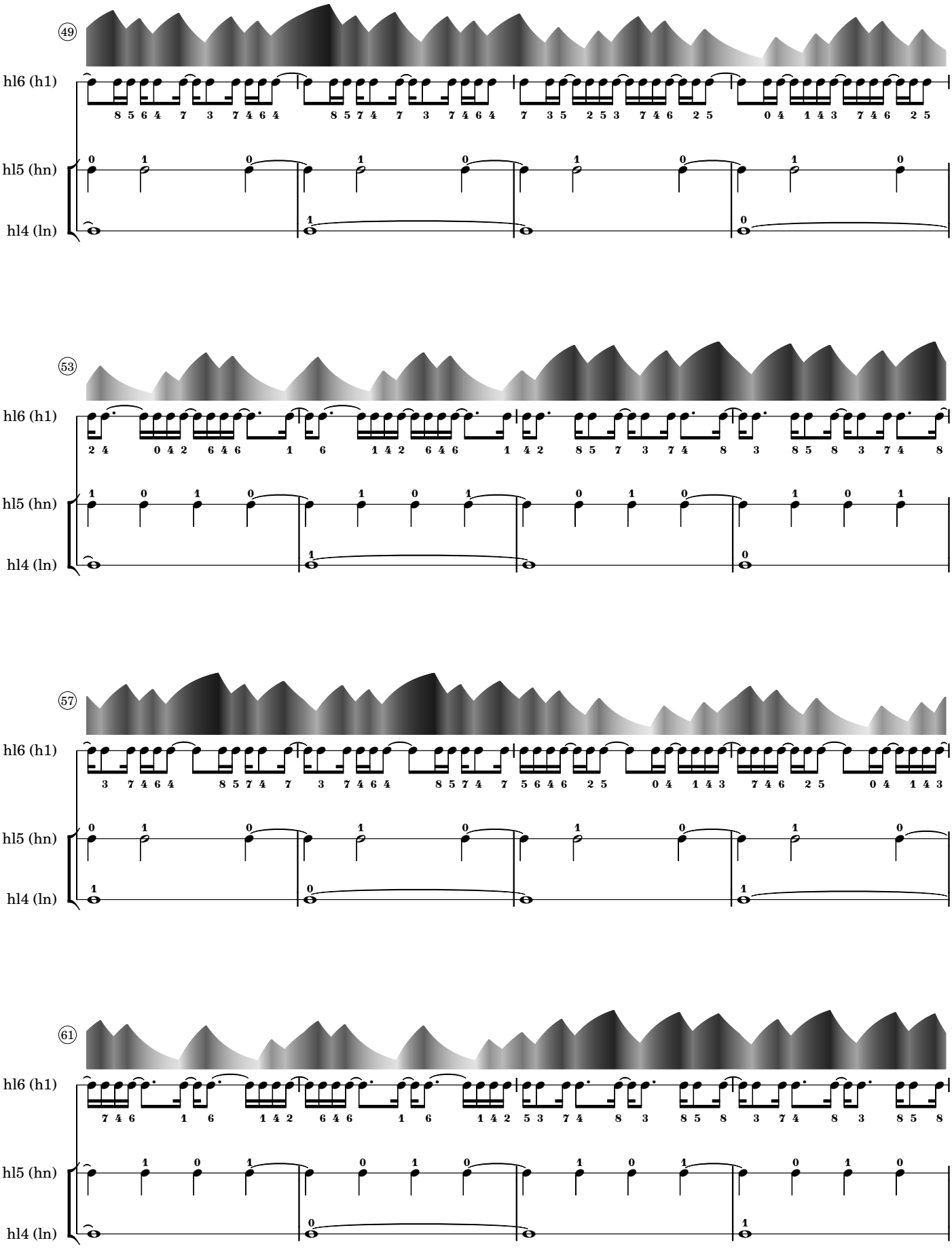
(cdmx, mx and nashville, usa; 2018)

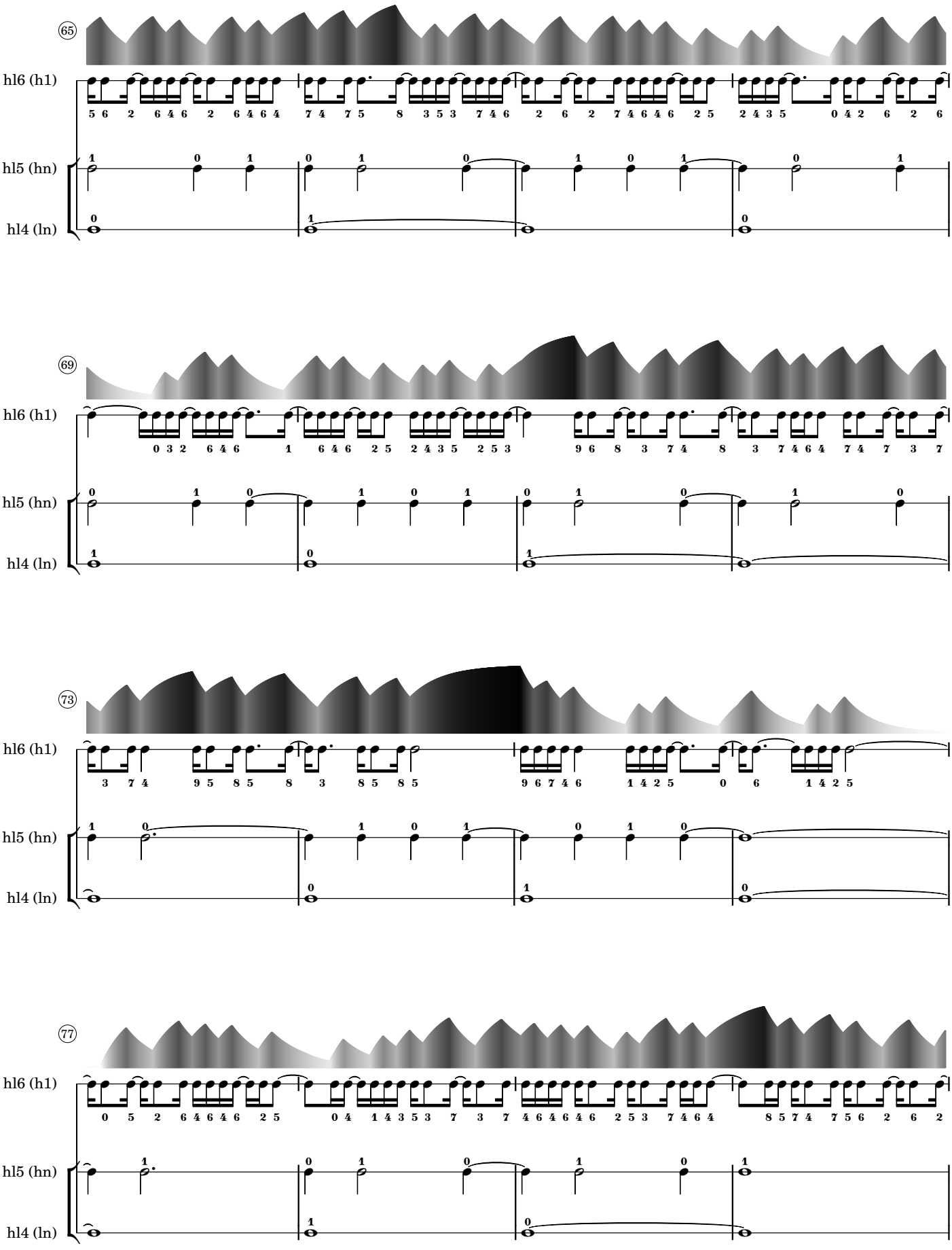
master score - 3231310

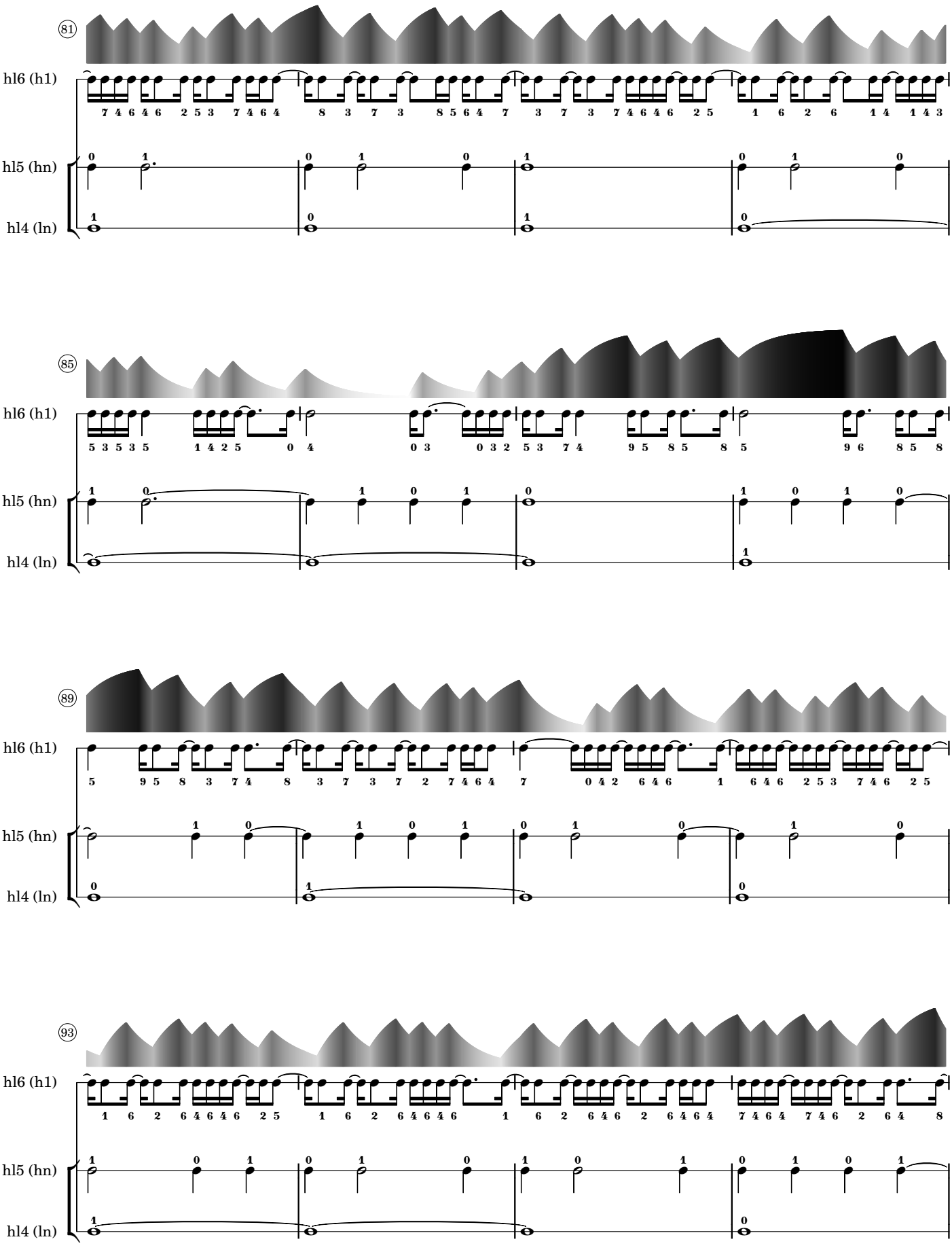


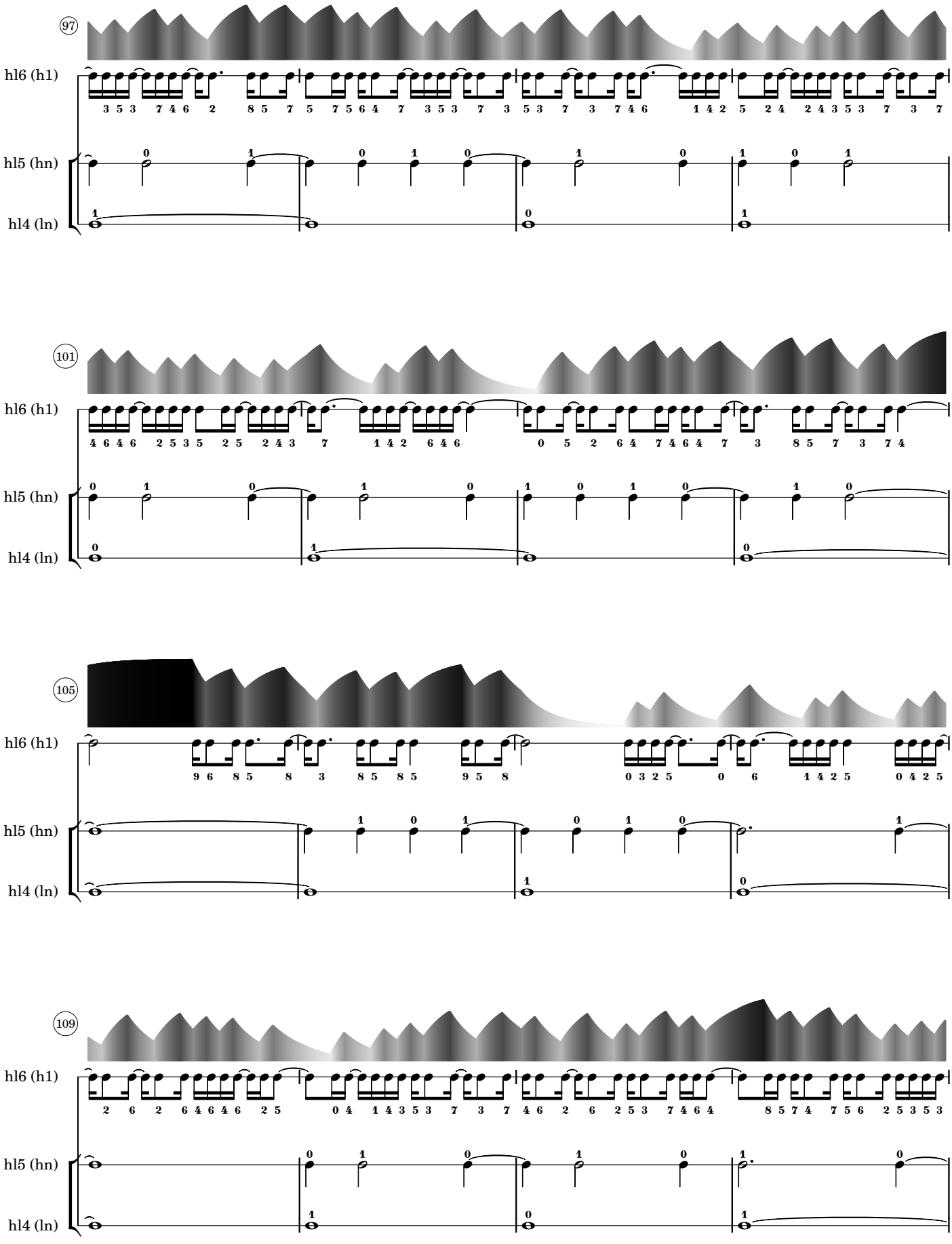

17
hl6 (h1)
hl5 (hn)
hl4 (ln)
21
hl6 (h1)
hl5 (hn)
hl4 (ln)
25
hl6 (h1)
hl5 (hn)
hl4 (ln)
29
hl6 (h1)
hl5 (hn)
hl4 (ln)

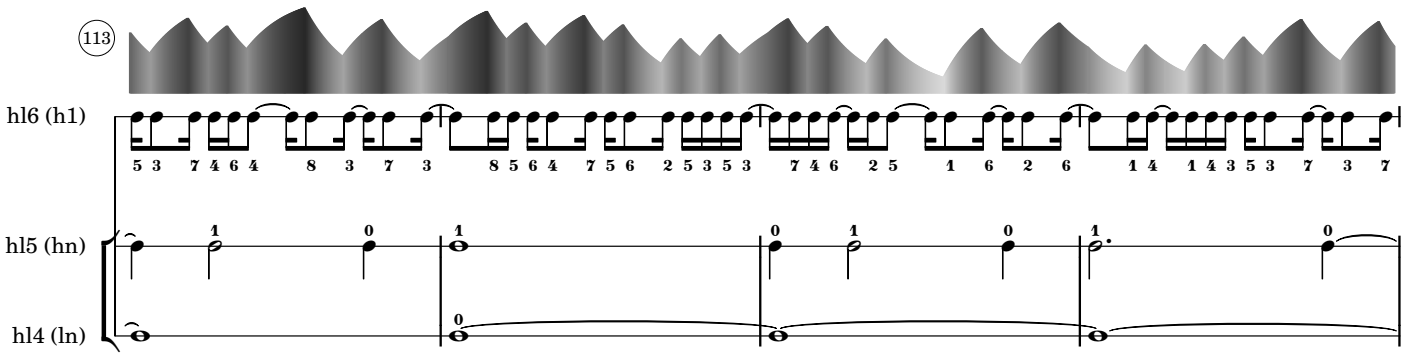

33
hl6 (h1)
5 3 5 3 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 6 2 7 2 7 2 7 2 7 2 7 2 7 2 7
hl5 (hn)
0 1 0 1 0 1 0 1 0
hl4 (ln)
1
37
hl6 (h1)
4 6 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 5 3 5 3 7 3 7 3 7 2 7 2 7 2 7 2 7 2 7
hl5 (hn)
0 1 0 1 0 1 0 1 0 1 0
hl4 (ln)
41
hl6 (h1)
4 9
hl5 (hn)
hl4 (ln)
0
45
hl6 (h1)
0 5 2 6 2 6 2 7 2 7 2 7 2 7 2 7 4 6 2 6 2 6 2 7 2 7 2 7 2 7 2 7 2 7 2
hl5 (hn)
1
hl4 (ln)

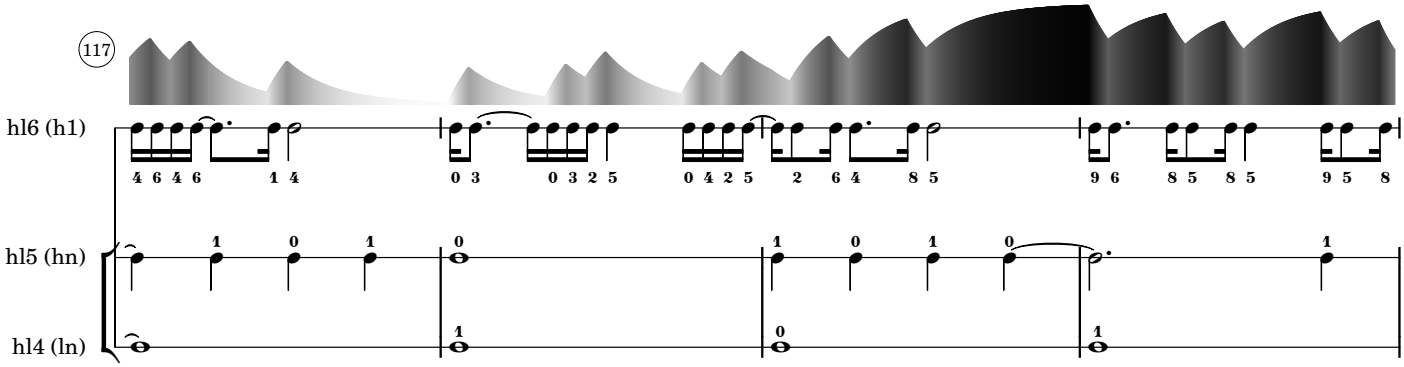

49
hl6 (h1)
hl5 (hn)
hl4 (ln)
53
hl6 (h1)
hl5 (hn)
hl4 (ln)
57
hl6 (h1)
hl5 (hn)
hl4 (ln)
61
hl6 (h1)
hl5 (hn)
hl4 (ln)

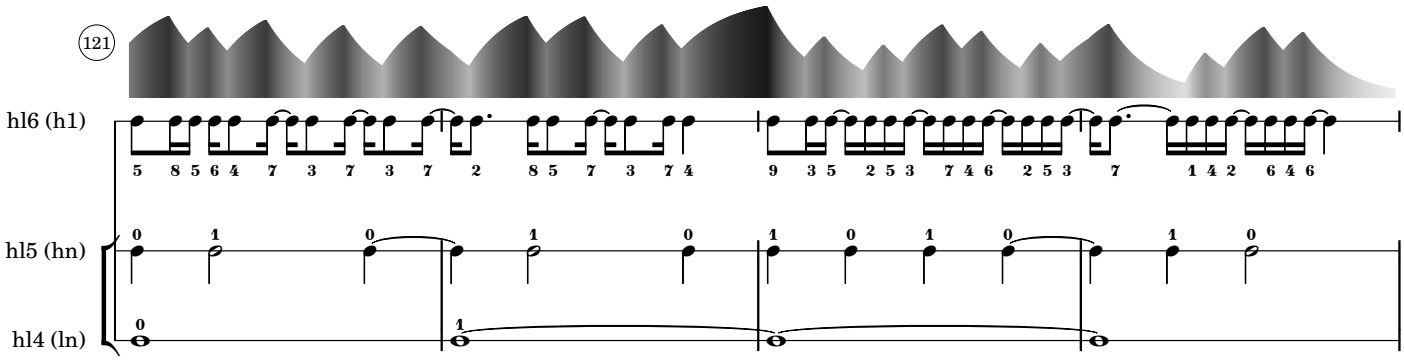

65
hl6 (h1)
5 6 2 6 4 6 2 6 4 6 4 7 4 7 5 8 3 5 3 7 4 6 2 6 2 7 4 6 4 6 2 5 2 4 3 5 0 4 2 6 2 6
hl5 (hn)
hl4 (ln)
69
hl6 (h1)
0 3 2 6 4 6 1 6 4 6 2 5 2 4 3 5 2 5 3 9 6 8 3 7 4 8 3 7 4 6 4 7 4 7 3 7
hl5 (hn)
hl4 (ln)
73
hl6 (h1)
3 7 4 9 5 8 5 8 3 8 5 8 5 9 6 7 4 6 1 4 2 5 0 6 1 4 2 5
hl5 (hn)
hl4 (ln)
77
hl6 (h1)
0 5 2 6 4 6 4 6 2 5 0 4 1 4 3 5 3 7 3 7 4 6 4 6 4 6 2 5 3 7 4 6 4 8 5 7 4 7 5 6 2 6 2
hl5 (hn)
hl4 (ln)

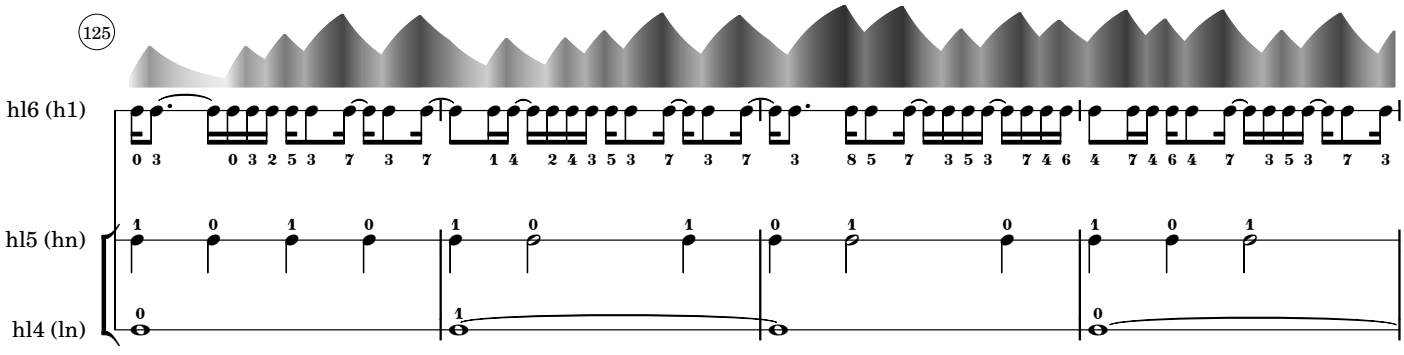

81
hl6 (h1)
7 4 6 4 6 2 5 3 7 4 6 4 8 3 7 3 8 5 6 4 7 3 7 3 7 4 6 4 6 2 5 1 6 2 6 1 4 1 4 3
hl5 (hn)
hl4 (ln)
85
hl6 (h1)
5 3 5 3 5 1 4 2 5 0 4 0 3 0 3 2 5 3 7 4 9 5 8 5 8 5 9 6 8 5 8
hl5 (hn)
hl4 (ln)
89
hl6 (h1)
5 9 5 8 3 7 4 8 3 7 3 7 2 7 4 6 4 7 0 4 2 6 4 6 1 6 4 6 2 5 3 7 4 6 2 5
hl5 (hn)
hl4 (ln)
93
hl6 (h1)
1 6 2 6 4 6 4 6 2 5 1 6 2 6 4 6 4 6 1 6 2 6 4 6 2 6 4 6 4 7 4 6 4 7 4 6 2 6 4 8
hl5 (hn)
hl4 (ln)

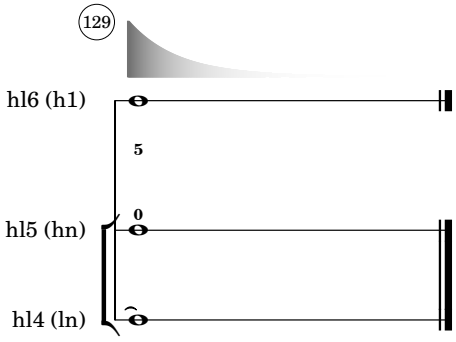

97
hl6 (h1)
3 5 3 7 4 6 2 8 5 7 5 7 5 6 4 7 3 5 3 7 3 5 3 7 3 7 4 6 1 4 2 5 2 4 2 4 3 5 3 7 3 7
hl5 (hn)
hl4 (ln)
101
hl6 (h1)
4 6 4 6 2 5 3 5 2 5 2 4 3 7 1 4 2 6 4 6 0 5 2 6 4 7 4 6 4 7 3 8 5 7 3 7 4
hl5 (hn)
hl4 (ln)
105
hl6 (h1)
9 6 8 5 8 3 8 5 8 5 9 5 8 0 3 2 5 0 6 1 4 2 5 0 4 2 5
hl5 (hn)
hl4 (ln)
109
hl6 (h1)
2 6 2 6 4 6 4 6 2 5 0 4 1 4 3 5 3 7 3 7 4 6 2 6 2 5 3 7 4 6 4 8 5 7 4 7 5 6 2 5 3 5 3
hl5 (hn)
hl4 (ln)


113
hl6 (h1)
5 3 7 4 6 4 8 3 7 3 8 5 6 4 7 5 6 2 5 3 5 3 7 4 6 2 5 1 6 2 6 1 4 1 4 3 5 3 7 3 7
hl5 (hn)
hl4 (ln)


117
hl6 (h1)
4 6 4 6 1 4 0 3 0 3 2 5 0 4 2 5 2 6 4 8 5 9 6 8 5 8 5 9 5 8
hl5 (hn)
hl4 (ln)


121
hl6 (h1)
5 8 5 6 4 7 3 7 3 7 2 8 5 7 3 7 4 9 3 5 2 5 3 7 4 6 2 5 3 7 1 4 2 6 4 6
hl5 (hn)
hl4 (ln)


125
hl6 (h1)
0 3 0 3 2 5 3 7 3 7 1 4 2 4 3 5 3 7 3 7 3 8 5 7 3 5 3 7 4 6 4 7 4 6 4 7 3 5 3 7 3
hl5 (hn)
hl4 (ln)

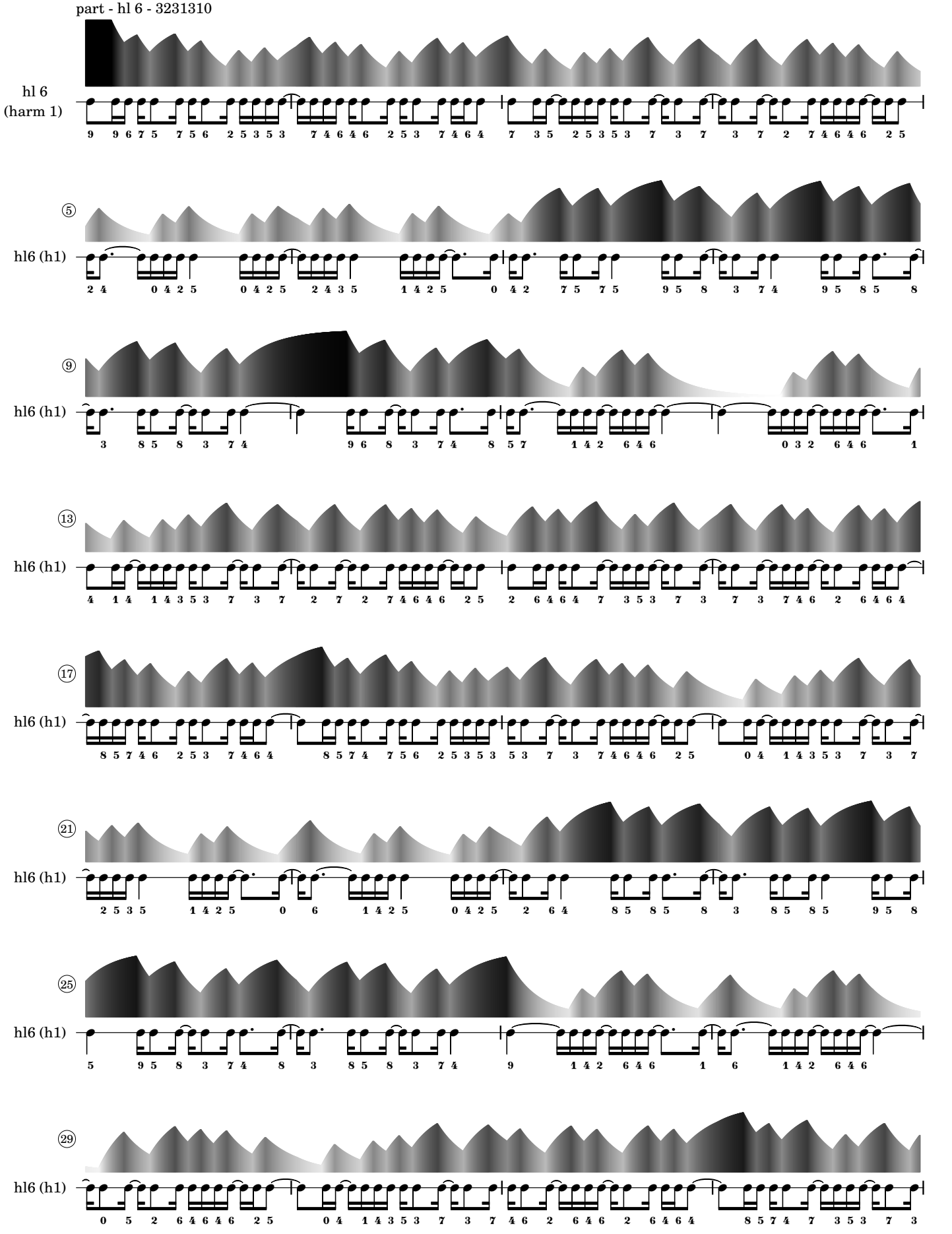

129
hl6 (h1)
5
0
hl5 (hn)
hl4 (ln)

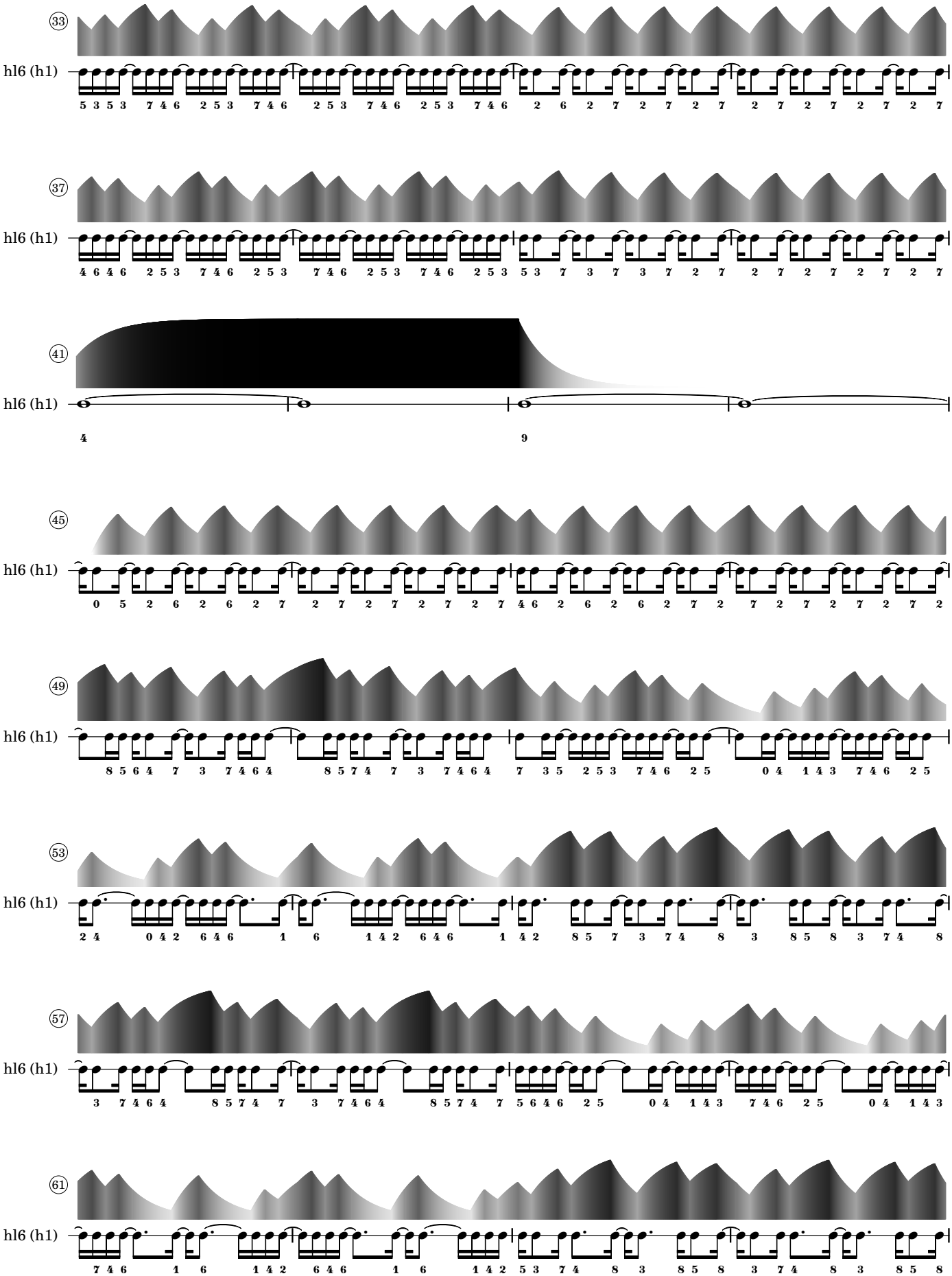
a lot of tiles (trivial scan)

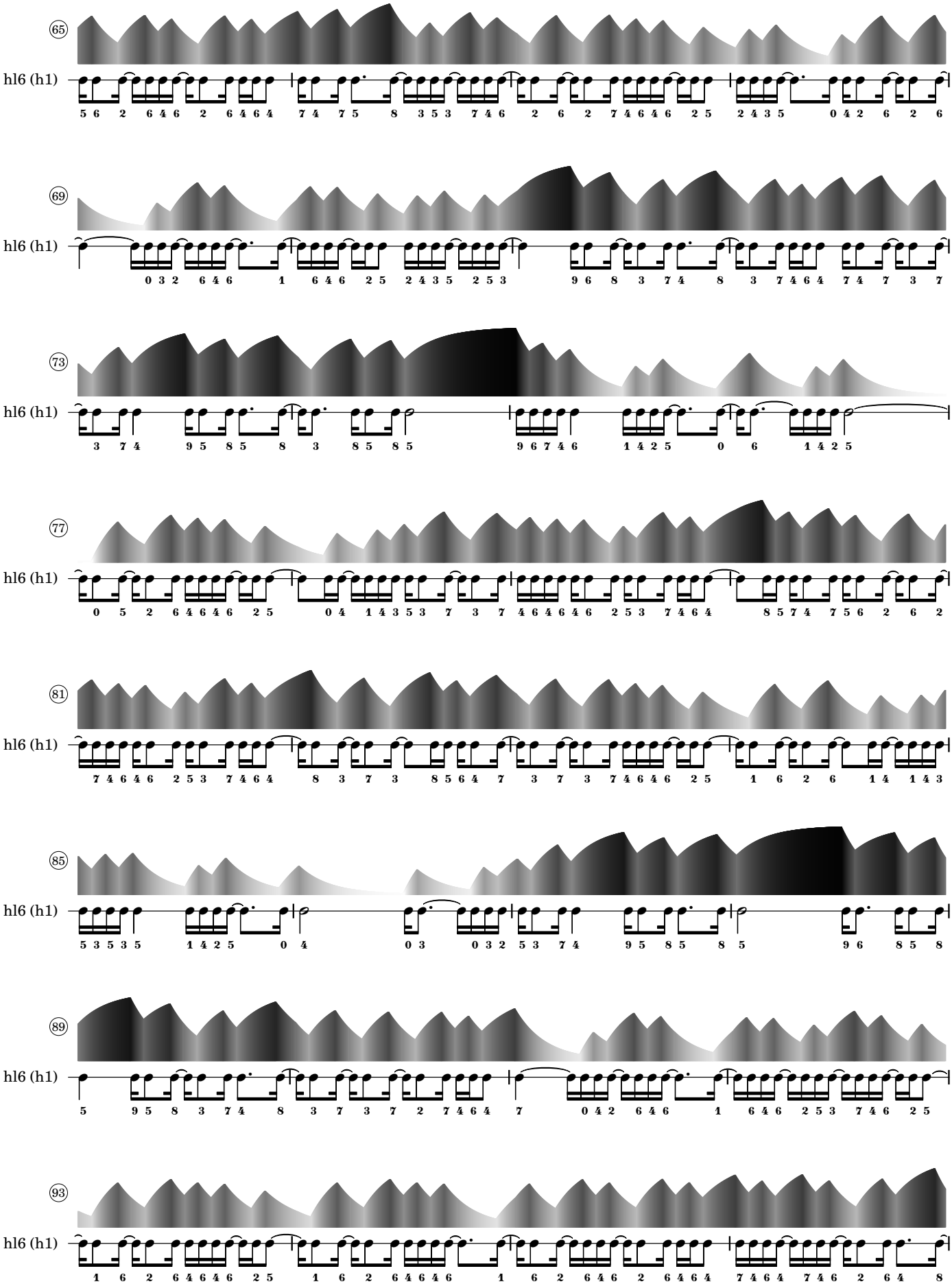
3231310

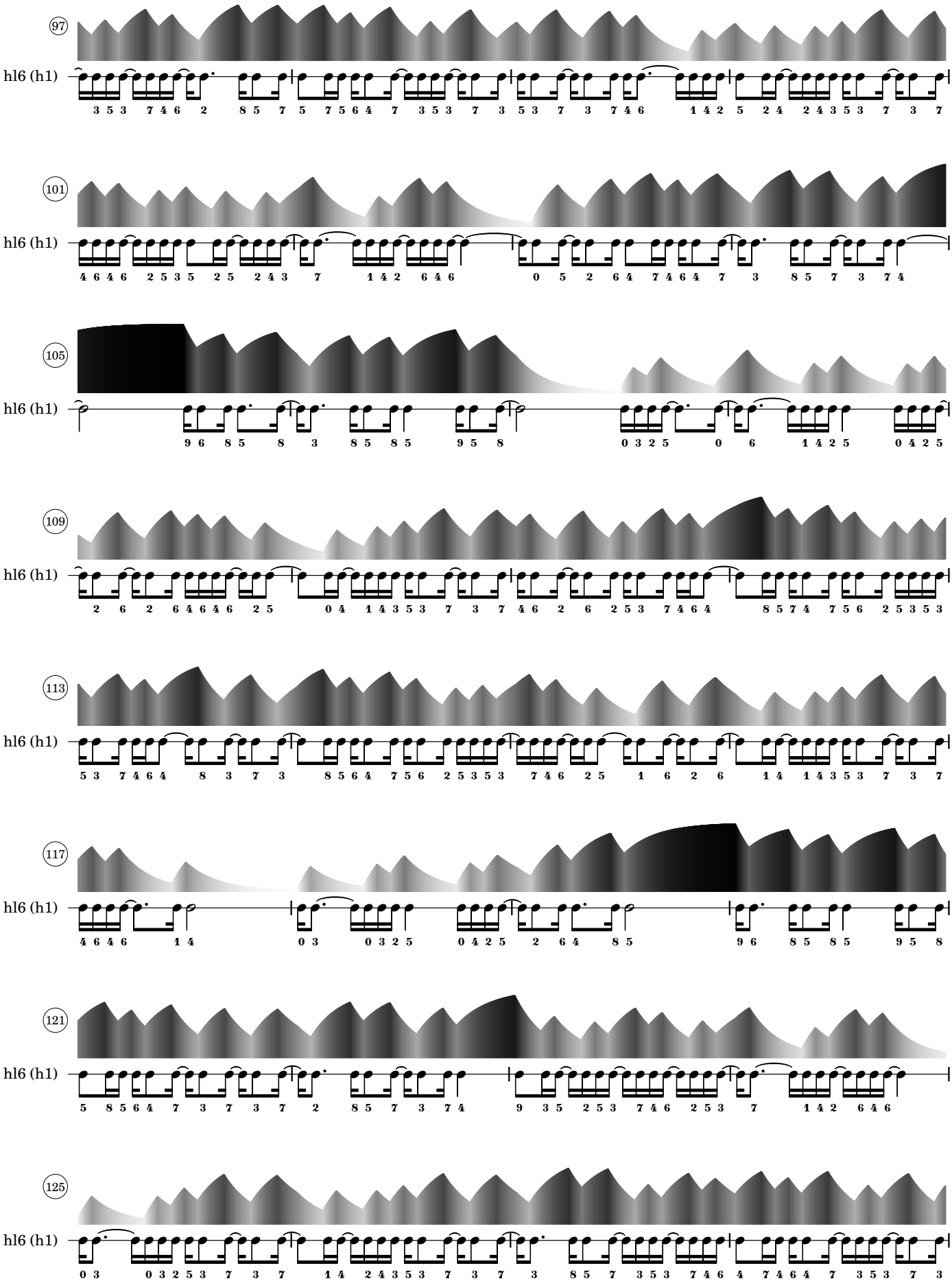
michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 6 - 3231310

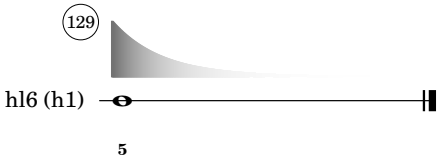



33
hl6 (h1)
5 3 5 3 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 6 2 7 2 7 2 7 2 7 2 7 2 7 2 7
37
hl6 (h1)
4 6 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 5 3 7 4 6 2 5 3 5 3 7 3 7 3 7 2 7 2 7 2 7 2 7 2 7
41
hl6 (h1)
4
9
45
hl6 (h1)
0 5 2 6 2 6 2 7 2 7 2 7 2 7 2 7 2 7 4 6 2 6 2 6 2 7 2 7 2 7 2 7 2 7 2
49
hl6 (h1)
8 5 6 4 7 3 7 4 6 4 8 5 7 4 7 3 7 4 6 4 7 3 5 2 5 3 7 4 6 2 5 0 4 1 4 3 7 4 6 2 5
53
hl6 (h1)
2 4 0 4 2 6 4 6 1 6 1 4 2 6 4 6 1 4 2 8 5 7 3 7 4 8 3 8 5 8 3 7 4 8
57
hl6 (h1)
3 7 4 6 4 8 5 7 4 7 3 7 4 6 4 8 5 7 4 7 5 6 4 6 2 5 0 4 1 4 3 7 4 6 2 5 0 4 1 4 3
61
hl6 (h1)
7 4 6 1 6 1 4 2 6 4 6 1 6 1 4 2 5 3 7 4 8 3 8 5 8 3 7 4 8 3 8 5 8


65
hl6 (h1)
5 6 2 6 4 6 2 6 4 6 4 7 4 7 5 8 3 5 3 7 4 6 2 6 2 7 4 6 4 6 2 5 2 4 3 5 0 4 2 6 2 6
69
hl6 (h1)
0 3 2 6 4 6 1 6 4 6 2 5 2 4 3 5 2 5 3 9 6 8 3 7 4 8 3 7 4 6 4 7 4 7 3 7
73
hl6 (h1)
3 7 4 9 5 8 5 8 3 8 5 8 5 9 6 7 4 6 1 4 2 5 0 6 1 4 2 5
77
hl6 (h1)
0 5 2 6 4 6 4 6 2 5 0 4 1 4 3 5 3 7 3 7 4 6 4 6 4 6 2 5 3 7 4 6 4 8 5 7 4 7 5 6 2 6 2
81
hl6 (h1)
7 4 6 4 6 2 5 3 7 4 6 4 8 3 7 3 8 5 6 4 7 3 7 3 7 4 6 4 6 2 5 1 6 2 6 1 4 1 4 3
85
hl6 (h1)
5 3 5 3 5 1 4 2 5 0 4 0 3 0 3 2 5 3 7 4 9 5 8 5 8 5 9 6 8 5 8
89
hl6 (h1)
5 9 5 8 3 7 4 8 3 7 3 7 2 7 4 6 4 7 0 4 2 6 4 6 1 6 4 6 2 5 3 7 4 6 2 5
93
hl6 (h1)
1 6 2 6 4 6 4 6 2 5 1 6 2 6 4 6 4 6 1 6 2 6 4 6 2 6 4 6 4 7 4 6 4 7 4 6 2 6 4 8


97
hl6 (h1)
3 5 3 7 4 6 2 8 5 7 5 7 5 6 4 7 3 5 3 7 3 5 3 7 3 7 4 6 1 4 2 5 2 4 2 4 3 5 3 7 3 7
101
hl6 (h1)
4 6 4 6 2 5 3 5 2 5 2 4 3 7 1 4 2 6 4 6 0 5 2 6 4 7 4 6 4 7 3 8 5 7 3 7 4
105
hl6 (h1)
9 6 8 5 8 3 8 5 8 5 9 5 8 0 3 2 5 0 6 1 4 2 5 0 4 2 5
109
hl6 (h1)
2 6 2 6 4 6 4 6 2 5 0 4 1 4 3 5 3 7 3 7 4 6 2 6 2 5 3 7 4 6 4 8 5 7 4 7 5 6 2 5 3 5 3
113
hl6 (h1)
5 3 7 4 6 4 8 3 7 3 8 5 6 4 7 5 6 2 5 3 5 3 7 4 6 2 5 1 6 2 6 1 4 1 4 3 5 3 7 3 7
117
hl6 (h1)
4 6 4 6 1 4 0 3 0 3 2 5 0 4 2 5 2 6 4 8 5 9 6 8 5 8 5 9 5 8
121
hl6 (h1)
5 8 5 6 4 7 3 7 3 7 2 8 5 7 3 7 4 9 3 5 2 5 3 7 4 6 2 5 3 7 1 4 2 6 4 6
125
hl6 (h1)
0 3 0 3 2 5 3 7 3 7 1 4 2 4 3 5 3 7 3 7 3 8 5 7 3 5 3 7 4 6 4 7 4 6 4 7 3 5 3 7 3

(part - hl 6 - 3231310)

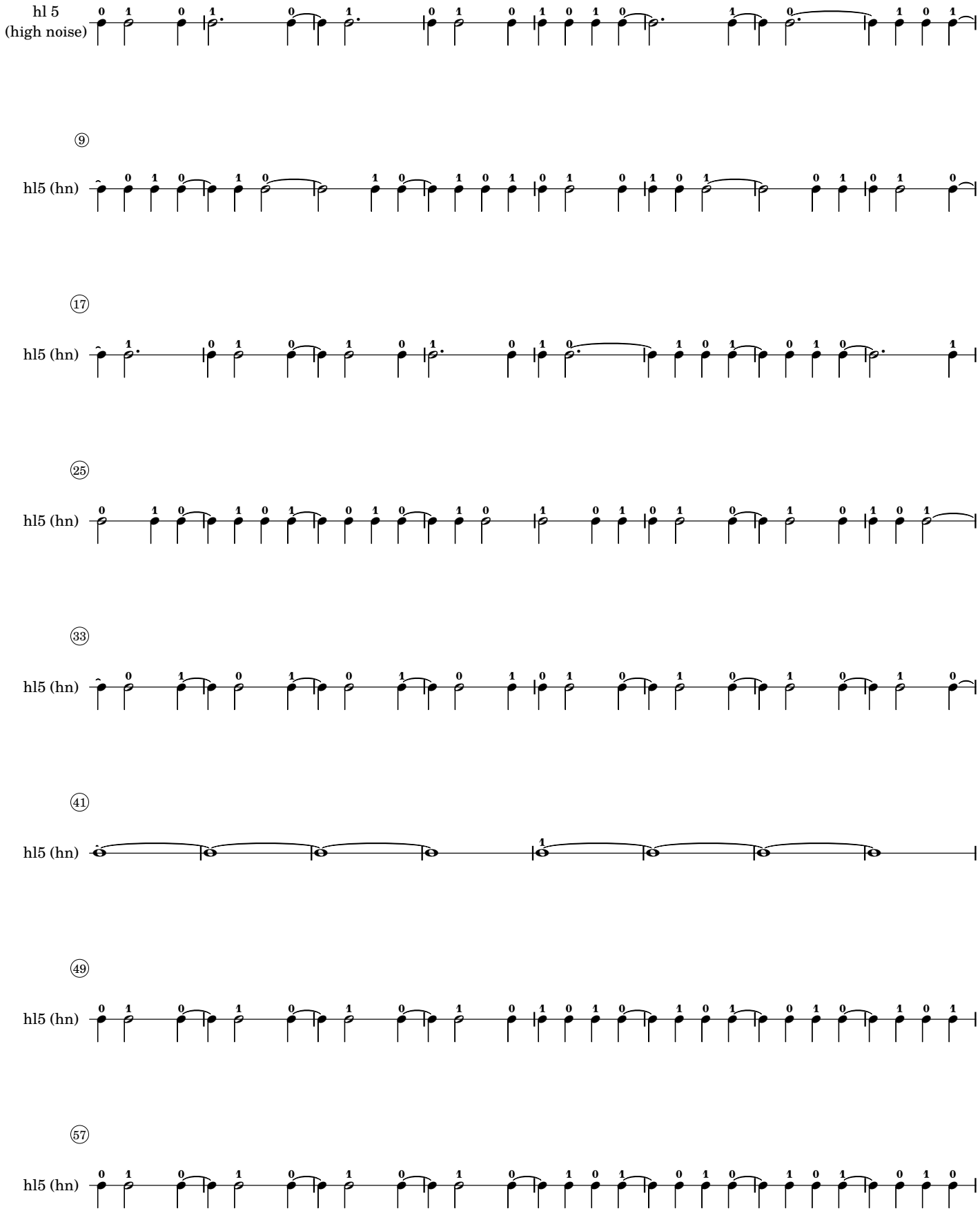


a lot of tiles (trivial scan)

3231310

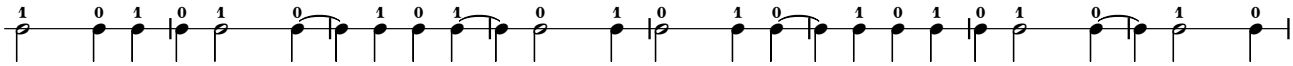
michael winter
(cdmx, mx and nashville, usa; 2018)

part - hl 5 - 3231310



65

hl5 (hn)

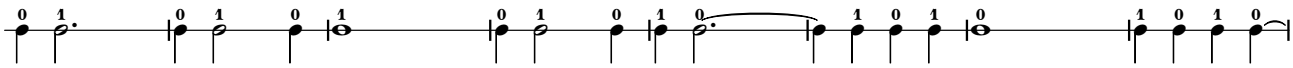


73

hl5 (hn)

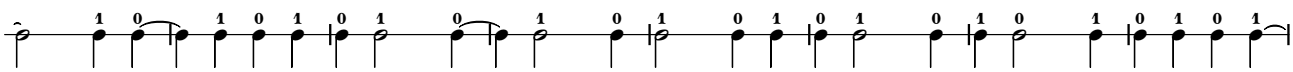
81

hl5 (hn)



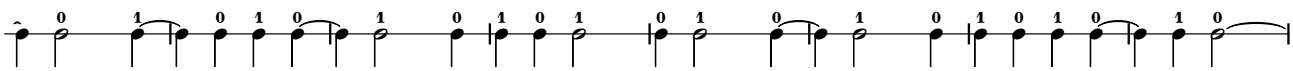
89

hl5 (hn)



97

hl5 (hn)

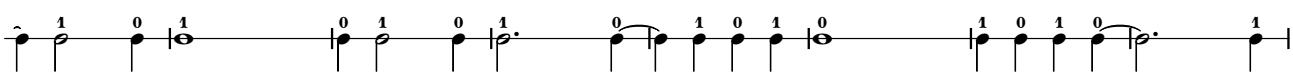


105

hl5 (hn)

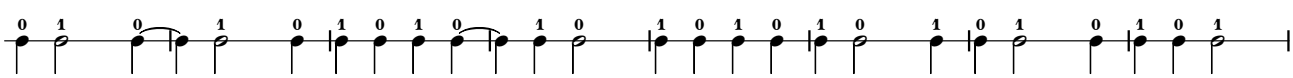
113

hl5 (hn)



121

hl5 (hn)



(part - hl 5 - 3231310)

129

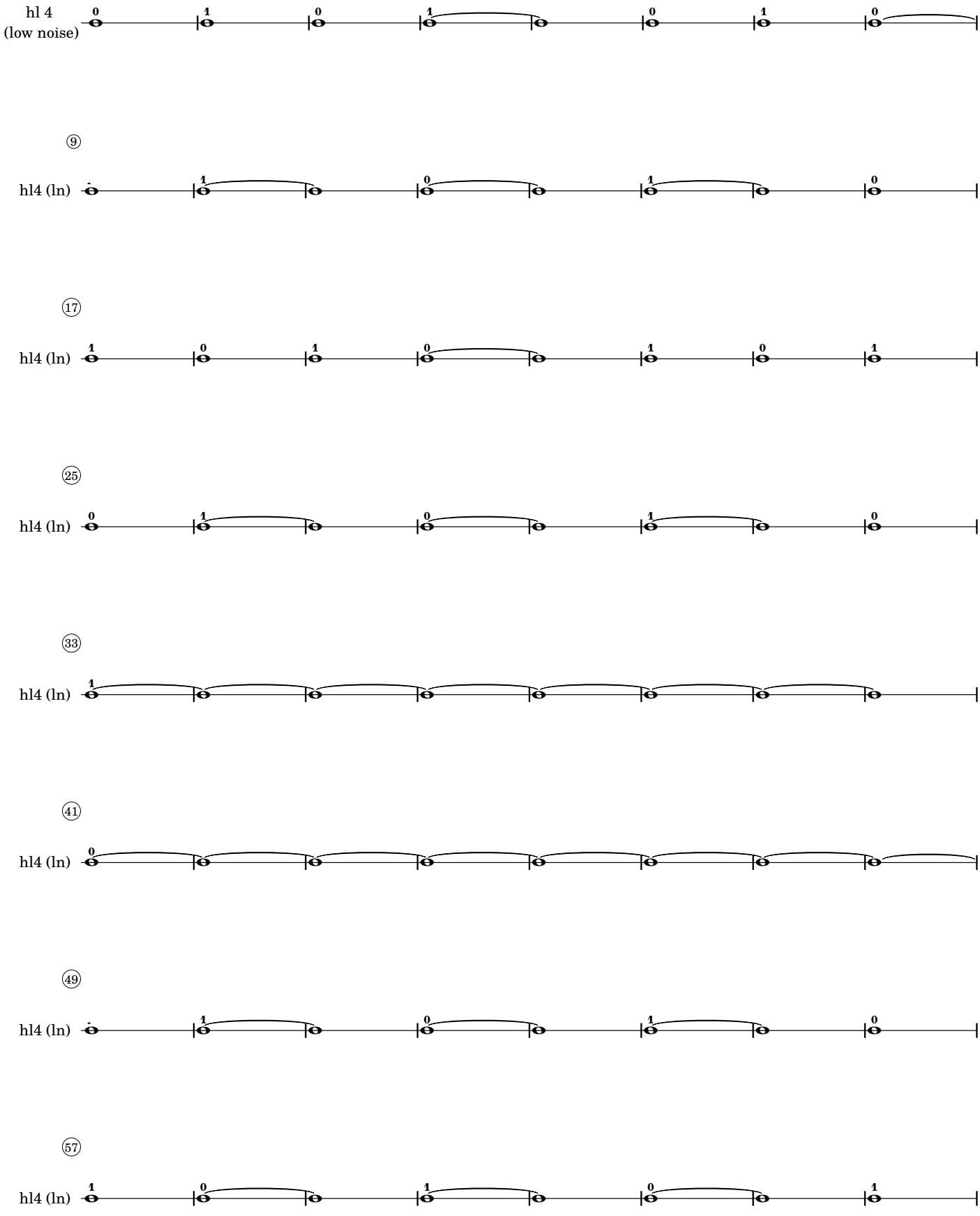
hl5 (hn)

a lot of tiles (trivial scan)

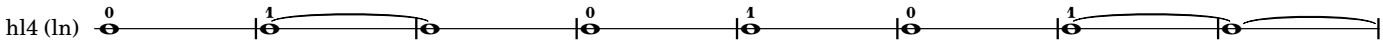
3231310

michael winter
(cdmx, mx and nashville, usa; 2018)

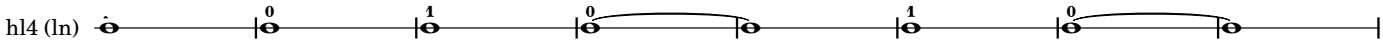
part - hl 4 - 3231310



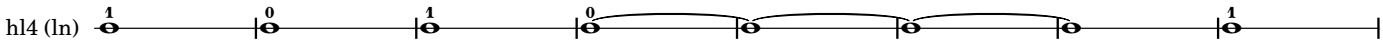
65



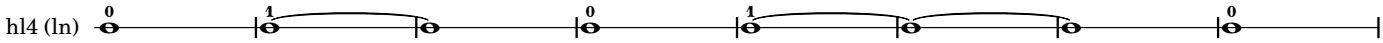
73



81

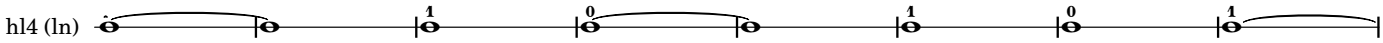


89

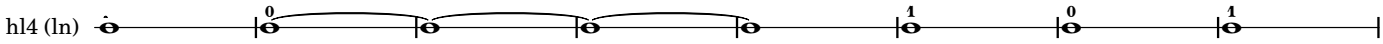


97

105



113



121

(part - hl 4 - 3231310)

129

hl4 (ln)

a_lot_of_tiles_trivial_scan_main.scd

```
(
// MAIN LAUNCH (loads necessary files and definitions)

var appEnvironment, cond;

s.boot;

appEnvironment = Environment.make;
appEnvironment.push;

// load
"a_lot_of_tiles_trivial_scan_tiler.scd".loadRelative;
"a_lot_of_tiles_trivial_scan_visualizer.scd".loadRelative;
"a_lot_of_tiles_trivial_scan_sonifier.scd".loadRelative;
"a_lot_of_tiles_trivial_scan_gui.scd".loadRelative;
"a_lot_of_tiles_trivial_scan_transcriber.scd".loadRelative;

// init
~dir = thisProcess.nowExecutingPath.dirname;
~loadedTransform = nil;
~transform = [2, 1, 0, 1, 0, 0, 0];
~tileMap = ~mapAll.value(6 /*max depth*/, ~transform);
~layoutState = 0;
~dur = 0.125;
~continuousPlay = false;

// launch
~launchTileVisualizer.value;
~launchGui.value;

appEnvironment.pop;
)
```

a_lot_of_tiles_trivial_scan_tiler.scd

```
(
var substitute, resamp, read, readPos = [0, 0];

// this function is the tiler
// it is essentially the generator of the underlying structure of the piece and all the variations
substitute = {arg parent, transform; var s, t, u, v, sub;
    s = 4.collect({arg i; parent[0] + ((parent[i] - parent[0]) / 2)});
    t = [parent[2] + ((parent[3] - parent[2]) / 2), parent[2], s[3], s[2]];
    u = [t[0], s[1], t[0] + ((parent[1] - s[1]) / 2), s[1] + ((parent[1] - s[1]) / 2)];
    v = [parent[1], parent[3], u[3], u[2]];
    sub = [s, t, u, v].collect({arg child, i; if(transform[i] == nil, {nil},
        {4.collect({arg j; child[((j.asDigits(2, 2) + transform[i].asDigits(2, 2)) % 2).convertDigits(2)]})})});
    sub.removeEvery([nil])};

// these functions either tile a specific depth (mapDepth) or up to a specific depth (mapAll)
~mapDepth = {arg depth, transform; var tiles = [[[0, 0], [2, 0], [0, 1], [2, 1]]];
    depth.do({arg i; tiles = tiles.collect({arg tile; substitute.value(tile, transform)}).flatten});
    [tiles]};

~mapAll = {arg depth, transform; var tiles = Array.fill(depth + 1, {[]});
    tiles[0].add([[0, 0], [2, 0], [0, 1], [2, 1]]);
    depth.do({arg i; tiles[i].do({arg tile; tiles[i + 1] = tiles[i + 1].addAll(substitute.value(tile, transform))})});
    tiles};

// utility to resample the matrix output by ~matricize
resamp = {arg matrixOrig, resampLevel; var mult, matrixNew = [];
    mult = pow(2, resampLevel);
    matrixOrig.do({arg rowOrig; var rowNew = [];
        rowOrig.do({arg elem; mult.do({rowNew = rowNew.add(elem)})});
        mult.do({matrixNew = matrixNew.add(rowNew)})});
    matrixNew};

// function to turn the tile map into a matrix of 1s and 0s by reading blocks of 2 vertical or horizontal tiles
~matricize = {
    arg tileMap, depth = 6, resampLevel = 0, transform;
    var rotationState = transform[4], mirrorState = transform[5], invertState = transform[6], matrix;
    matrix = Array.fill2D(pow(2, depth - 1), pow(2, depth), {0});
    tileMap[tileMap.size-1].do({arg tile; var transpose, xMin, xMax, yMin, yMax;
        transpose = (tile * pow(2, depth)).flop;
        xMin = minItem(transpose[0]); xMax = maxItem(transpose[0]);
        yMin = minItem(transpose[1]); yMax = maxItem(transpose[1]);
        if(xMin + yMin % 2 == 1, {if(xMax - xMin == 1,
            {matrix[yMin / 2][xMin / 2] = ((invertState + 1) % 2)}, {matrix[yMin / 2][xMin / 2] = invertState})})});
    if(resampLevel > 0, {matrix = resamp.value(matrix, resampLevel)});
    matrix = switch(rotationState, 0, {matrix}, 1, {matrix.flop},
        2, {matrix.reverse.flop.reverse.flop}, 3, {matrix.reverse.flop});
    if(mirrorState == 1, {matrix = matrix.flop.reverse.flop});
    matrix};

// tape reading model that reads matrix from ~genData function
// wordShift and letterShift are only used if readPos is not reset elsewhere.
read = {
    arg matrix, wordLen = 1, wordShift = [0, 1], letterShift = [0, 1], wrap = true;
    var width = matrix.shape[1], height = matrix.shape[0], startReadPos = readPos, word = [];
    wordLen.do({arg i;
        word = word.add(matrix[readPos[0] % height][readPos[1] % width]);
        readPos = readPos + letterShift});
    readPos = startReadPos + wordShift; word.convertDigits(2)};

// read the matrix - this data is ultimately used to generate the piece
~genData = {arg matrix, transform, partSize, shift = 0, pad = 16, minScale = 0, maxScale = 1;
    var width = matrix.shape[1], height = matrix.shape[0], res;
    readPos = [0, 0];
    res = partSize.collect({arg p; height.collect({arg i;
        readPos[0] = i + p + shift;
        width.collect({arg j; var val = read.value(matrix);
            minScale + ((maxScale - minScale) * val)});
}).flatten.addAll(Array.fill(pad, {0}))})};
)
```

a_lot_of_tiles_trivial_scan_sonifier.scd

```
(
// synth for hl 6
SynthDef(\hl_6_sine, {arg freq, amp = 1, pos = 0;
    Out.ar(0,  Pan2.ar(SinOsc.ar(freq, 0, Lag.kr(amp, 2)), pos))
}).store;

// synth for hl 5
SynthDef(\hl_5_high_noise, {arg amp = 1, pos = 0;
    Out.ar(0,  Pan2.ar(HPF.ar(WhiteNoise.ar(Lag.kr(amp * (1 / 128), 0.05)), 5000), pos))
}).store;

// synth for hl 4
SynthDef(\hl_4_low_noise, {arg amp = 1, pos = 0;
    Out.ar(0,  Pan2.ar(LPF.ar(WhiteNoise.ar(Lag.kr(amp * (1 / 4), 0.05)), 300), pos))
}).store;

// synth for amp curves for score
SynthDef(\lamp, {arg freq, amp = 1;
    Out.ar(0, Lag.ar(K2A.ar(amp), 2))
}).store;

// synth for playback
SynthDef(\play, {arg sinePlayer1, sinePlayer2, highNoisePlayer, lowNoisePlayer,
    sineBuf1 = 0, sineBuf2 = 1, highNoiseBuf = 2, lowNoiseBuf = 3,
    eTracks, eTracksPanned, eMaster,
    eVol = #[0.8, 0.8, 0.8, 0.8], eMute = #[1, 1, 1, 1], ePan = #[0, 0, 0, 0], masterVol = 1, masterMute = 1,
    playRate = 0, startPos = 0, startTrig = 0, hash;
    var dStartTrig, phasor, countOff, imp, curBeat;

    countOff = PulseCount.kr(Impulse.kr(4), startTrig) * startTrig;
    dStartTrig = countOff > 17;
    phasor = Phasor.ar(dStartTrig, Select.kr(playRate * dStartTrig, [0, BufRateScale.kr(sineBuf1)]),
        0, BufFrames.kr(sineBuf1), startPos * BufFrames.kr(sineBuf1));
    curBeat = ((A2K.kr(phasor) / BufFrames.kr(sineBuf1)) * BufDur.kr(sineBuf1) * 4).trunc;
    curBeat = Select.kr(dStartTrig, [countOff - 18, curBeat]);

    sinePlayer1 = PlayBuf.ar(2, sineBuf1, playRate * dStartTrig, dStartTrig, startPos * BufFrames.kr(sineBuf1));
    sinePlayer2 = PlayBuf.ar(2, sineBuf2, playRate * dStartTrig, dStartTrig, startPos * BufFrames.kr(sineBuf2));
    highNoisePlayer = PlayBuf.ar(2, highNoiseBuf, playRate * dStartTrig, dStartTrig, startPos * BufFrames.kr(highNoiseBuf));
    lowNoisePlayer = PlayBuf.ar(2, lowNoiseBuf, playRate * dStartTrig, dStartTrig, startPos * BufFrames.kr(lowNoiseBuf));

    eTracks = {arg i;
        [Mix.new(sinePlayer1), Mix.new(sinePlayer2), Mix.new(highNoisePlayer), Mix.new(lowNoisePlayer)][i] * eVol[i] * eMute[i]} ! 4;
    eTracksPanned = {arg i; Pan2.ar(eTracks[i], ePan[i])} ! 4;
    eMaster = Mix.new(eTracksPanned) * masterVol * masterMute * dStartTrig.lag;
    Out.ar(0, eMaster);

    // optional click — uncomment and send to an output not used
    //Out.ar(2, 10 * BPF.ar(WhiteNoise.ar * EnvGen.kr(Env.perc(0.01, 0.1), curBeat % 2 <= 0), 440 * ((curBeat % 8 <= 0) + 1), 0.02));
    SendTrig.kr(Changed.kr(curBeat), hash, curBeat);
    imp = Impulse.kr(10);
    SendReply.kr(imp, '/masterLevel', [Amplitude.kr(eMaster)], hash);
    SendReply.kr(imp, '/trackLevels', [Amplitude.kr(eTracks)], hash)}).add;

// generate audio for playback
~genPattern = {arg data, ins, durMult, genAudio = false, cond = nil;
    var pattern;
    pattern = Ppar(data.collect({
        arg seq, p;
        switch(ins,
            "hl_6_fundamental", {Pmono(\hl_6_sine, \freq, 31.midicps,
                \dur, ~dur, \amp, Pseq(seq), \pos, 0)},
            "hl_6_harmonics", {Pmono(\hl_6_sine, \freq, 31.midicps + ((31.midicps * 4) * (p + 1)),
                \dur, ~dur, \amp, Pseq(seq * (1 / pow(1 + ((p + 1) * 4), 1))), \pos, 0)},
            "hl_5_high_noise", {Pmono(\hl_5_high_noise, \dur, ~dur * durMult, \amp, Pseq(seq), \pos, 0)},
            "hl_4_low_noise", {Pmono(\hl_4_low_noise, \dur, ~dur * durMult, \amp, Pseq(seq), \pos, 0)})}));
    if(genAudio, {File.mkdir(~dir +/+ ".." +/+ "audio" +/+ "transform_" ++ ~transform.join ++ "_audio");
        pattern.render(~dir +/+ ".." +/+ "audio" +/+ "transform_" ++ ~transform.join ++ "_audio" +/+ ins ++ ".wav", ~dur * durMult * data[0].size,
            headerFormat: "WAV", sampleRate: s.sampleRate, action: {if(cond != nil, {cond.unhang})})});
    pattern};
)
```

a_lot_of_tiles_trivial_scan_visualizer.scd

```
(
var normalizeTile, getMasterRect, tileDrawFunc;

// provide scaled coordinate for the tiles
normalizeTile = {arg child, parent;
    var parentXMin, parentYMin, parentWidth, parentHeight, p1, p2, p3, p4;
    parentXMin = minItem(parent.slice(nil, 0));
    parentYMin = minItem(parent.slice(nil, 1));
    parentWidth = (maxItem(parent.slice(nil, 0)) - parentXMin);
    parentHeight = (maxItem(parent.slice(nil, 1)) - parentYMin);
    p1 = [(minItem(child.slice(nil, 0)) - parentXMin) / parentWidth, (minItem(child.slice(nil, 1)) - parentYMin) / parentHeight];
    p2 = [(maxItem(child.slice(nil, 0)) - parentXMin) / parentWidth, p1[1]];
    p3 = [p1[0], (maxItem(child.slice(nil, 1)) - parentYMin) / parentHeight];
    p4 = [p2[0], p3[1]];
    [p1, p2, p3, p4]};

// get the master rectangle for the tiling based on orientation
getMasterRect = {arg tileWin, rotationState;
    var rectWidth, rectHeight;
    case({(rotationState % 2 == 0) && (tileWin.bounds.width <= (tileWin.bounds.height * 2))},{
            rectWidth =  tileWin.bounds.width - 20;
            rectHeight = rectWidth / 2},
        {(rotationState % 2 == 0) && (tileWin.bounds.width > (tileWin.bounds.height * 2))},{
            rectHeight = tileWin.bounds.height - 20;
            rectWidth =  rectHeight * 2},
        {(rotationState % 2 == 1) && (tileWin.bounds.height <= (tileWin.bounds.width * 2))},{
            rectWidth = tileWin.bounds.height - 20;
            rectHeight =  rectWidth / 2},
        {(rotationState % 2 == 1) && (tileWin.bounds.height > (tileWin.bounds.width * 2))},{
            rectWidth = tileWin.bounds.width - 20;
            rectHeight =  rectWidth / 2});
    Rect.new((tileWin.bounds.width - rectWidth) / 2, (tileWin.bounds.height - rectHeight) / 2, rectWidth, rectHeight)};

// draw the tiling
~tileDrawFunc = {arg tileWin, tileMap, transform, layoutState;
    var rotationState = transform[4], mirrorState = transform[5], invertState = transform[6],
    masterRect, parentRects, depthLevels;

    masterRect = getMasterRect.value(tileWin, rotationState);
    parentRects = [[masterRect], [], []];

    Pen.width = (masterRect.width + masterRect.width) / 8000;
    Pen.scale(abs(mirrorState - 1) * 2 - 1, 1);
    Pen.translate(mirrorState * -1 * tileWin.bounds.width, 0);
    Pen.rotate(0.5 * rotationState * pi, tileWin.bounds.width / 2, tileWin.bounds.height / 2);

    depthLevels = switch(layoutState, 0, {[6]}, 1, {[5]}, 2, {[4]}, 3, {[4, 5, 6]});
    depthLevels.do({arg depth, dIndex;
        tileMap[depth].do({arg childTile, cIndex;
            var childTileNorm, parentRect, p1, p2, childRect;

            childTileNorm = if(dIndex > 0, {normalizeTile.value(childTile, tileMap[depth - 1][(cIndex / 4).trunc])}, {childTile});
            parentRect = if(dIndex > 0, {parentRects[dIndex][(cIndex / 4).trunc]}, {parentRects[0][0]});
            if((layoutState == 3) && (dIndex > 0), {parentRect = parentRect.insetBy(parentRect.width * 0.04 , parentRect.height * 0.04)});

            p1 = Point.new(childTileNorm[0][0] * parentRect.width / if(dIndex > 0, {1}, {2}), childTileNorm[0][1] * parentRect.height);
            p2 = Point.new(childTileNorm[3][0] * parentRect.width / if(dIndex > 0, {1}, {2}), childTileNorm[3][1] * parentRect.height);

            childRect = Rect.fromPoints(p1, p2).moveBy(parentRect.left, parentRect.top);
            if((layoutState == 3) && (depth < 6), {parentRects[dIndex + 1] = parentRects[dIndex + 1].add(childRect)});

            if(if(invertState == 0, {abs(p1.x - p2.x) > abs(p1.y - p2.y)}, {abs(p1.x - p2.x) < abs(p1.y - p2.y)}),
                {Pen.color = Color.white; Pen.fillRect(childRect); Pen.color = Color.black; Pen.strokeRect(childRect)},
                {Pen.color = Color.black; Pen.fillRect(childRect); Pen.color = Color.white; Pen.strokeRect(childRect)});
    })});

// launch the visualization
~launchTileVisualizer = {
    arg border = true, isLaunch = true;
    var tileWin;
    ~tileWin = Window("tile", if(isLaunch, {Rect(200, 500, 1000, 500)}, {~tileWin.bounds}), true, border);
    ~tileWin.view.background_(Color.gray(0.2));
    ~tileWin.drawFunc = {arg win; ~tileDrawFunc.value(win, ~tileMap, ~transform, ~layoutState)}.inEnvir;
    ~tileWin.front;
    ~tileWin.refresh}
)
```

```
(
~transcribe = {arg data, durMult, subName, drawAmps = false;

    data.do({arg seq, p;
        fork {var basePath, cond, pattern, ampFilePath, ampSampleRate, ampBuf, amps,
            lilyFile, lastVal, curTime = 0, lilyString = "";

            basePath = ~dir +/+ ".." +/+ "score" +/+ "transform_" ++ ~transform.join ++ "_score" +/+ "lilypond";

            // writes the template files for the master score and parts
            if(File.exists(basePath).not, {arg template, writeTemplate;
                writeTemplate = {arg template, part; var fileAppend, scoreType, systemsPerPage, lileFile, formattedTemplate;
                    fileAppend = switch(part, nil, {"master_score"}, 6, {"part_h16"}, 5, {"part_h15"}, 4, {"part_h14"});
                    scoreType = switch(part, nil, {"master score"}, 6, {"part — h1 6"}, 5, {"part — h1 5"}, 4, {"part — h1 4"});
                    lilyFile = File(basePath +/+ "a_lot_of_tiles_trivial_scan_transform_" ++ ~transform.join
                        ++ "-" ++ fileAppend ++ ".ly".standardizePath,"w");
                    formattedTemplate = template.replace("~~score_type~~", scoreType);
                    if(part != nil, {formattedTemplate = formattedTemplate
                        .replace("systems-per-page = 4", "systems-per-page = 8");
                        [4, 5, 6].difference([part]).do({arg p; formattedTemplate = formattedTemplate
                            .replace("%~~h1" ++ p ++ "-start~~", "%{~~h1" ++ p.asString ++ "~~")
                            .replace("%~~h1" ++ p.asString ++ "-end~~", "%}")});
                        if(part != 6, {formattedTemplate = formattedTemplate
                            .replace("unfold 33", "unfold 17").replace("unfold 3", "unfold 7")})});
                    lilyFile.write(formattedTemplate);
                    lilyFile.close};

                File.mkdir(basePath);
                File.mkdir(basePath +/+ "includes");
                File.mkdir(basePath +/+ ".." +/+ "pdf");
                template = File.readAllString(~dir +/+ ".." +/+ "score" +/+ "score_template"
                    +/+ "a_lot_of_tiles_trivial_scan_score_template.ly".standardizePath);
                template = template.replace("~~transform~~", ~transform.join);

                writeTemplate.value(template, nil);
                writeTemplate.value(template, 6);
                writeTemplate.value(template, 5);
                writeTemplate.value(template, 4);
            });

            cond = Condition.new;

            // generate pattern and get amplitude values for h16
            pattern = Pmono(\lamp, \dur, ~dur * durMult, \amp, Pseq(seq));
            if(drawAmps, {
                ampFilePath = basePath +/+ "amp_curve_tmp_h1_6_harm_" ++ (1 + (p * 4)).asString() ++ ".wav";
                pattern.render(ampFilePath, ~dur * durMult * seq.size, action: {cond.unhang}); cond.hang;
                ampSampleRate = SoundFile(ampFilePath).sampleRate;
                Buffer.readChannel(s, ampFilePath, channels: [0], action: {arg ampBuf;
                    ampBuf.loadToFloatArray(action: {arg array; amps = array; cond.unhang})}); cond.hang});

            lilyFile = if(data.size > 1, {lilyFile = subName ++ "_harm_" ++ (1 + (p * 4)).asString()}, {subName});
            lilyFile = File(basePath +/+ "includes" +/+ "a_lot_of_tiles_trivial_scan_" ++ lilyFile ++ ".ly".standardizePath,"w");

            // main transcribe code which groups data into beats and creates lilypond code accordingly
            lastVal = nil;
            seq.collect({arg v; durMult.collect({v})}).flat.clump(4).do({arg beat, i;
                beat.separate({arg a, b; a != b}).do({arg group; var noteLength, target = 0, colors = "", lilyNote;
                    noteLength = group.size;
                    lilyNote = switch(noteLength, 1, {"b'16 "}, 2, {"b'8 "}, 3, {"b'8. "}, 4, {"b'4 "});

                    // lilypond directives to draw amplitude gradients for h16
                    if(drawAmps, {
                        colors = amps[((ampSampleRate * curTime).asInteger)..((ampSampleRate * (curTime + (noteLength / 8))).asInteger)];
                        target = colors[0];
                        colors = 1 - (colors.resamp0(noteLength * 50));
                        colors = colors.collect({arg color; color.round(0.0001).asString});
                        colors = "\\override Glissando.details.colors = #'" ++ "(" ++ colors.join(" ") ++ ") "});

                    // 0 or 1 above every note based on data
                    if(drawAmps.not, {lilyNote = lilyNote ++ "^" ++ group[0].asString ++ " "});

                    // ties notes
                    lilyString = lilyString ++ if(group[0] == lastVal, {"~ "}, {""});
                    lilyString = lilyString ++ if((curTime % 2 == 0), {"|"}, {""});
                    lilyString = lilyString ++ colors ++ lilyNote;
                    lilyString = lilyString ++ if(drawAmps && (group[0] != lastVal),
                        {"-\\tweak Y-offset #-5.5 -" ++ (target.trunc(0.1) * 10).asString}, {""});
                    lilyString = lilyString ++ if(drawAmps, {"\n"}, {""});

                    lastVal = group[0];
                    curTime = curTime + (noteLength / 8);
                });
            });

            // consolidates tied quarters for h14 and h15
            if(drawAmps.not, {
                lilyString.findRegexp("b'4 \\^(0|1) ~ b'4 \\^(0|1) ~ b'4 \\^(0|1) ~ b'4 \\^(0|1)").clump(5).do(
                    {arg match; lilyString = lilyString.replace(match[0][1], "b'1 ^" ++ match[1][1])});
                lilyString.findRegexp("b'4 \\^(0|1) ~ b'4 \\^(0|1) ~ b'4 \\^(0|1)").clump(4).do(
                    {arg match; lilyString = lilyString.replace(match[0][1], "b'2. ^" ++ match[1][1])});
                lilyString.findRegexp("b'4 \\^(0|1) ~ b'4 \\^(0|1)").clump(3).do(
                    {arg match; lilyString = lilyString.replace(match[0][1], "b'2 ^" ++ match[1][1])});
                lilyString.findRegexp("~ \\|b'(1|2.|2|4) \\^(0|1)").clump(3).do(
                    {arg match; lilyString = lilyString.replace(match[0][1], "~ |b'" ++ match[1][1])})});

            // prettify
            lilyString = lilyString.replace("|", "\n|\n");
            lilyString = lilyString.replace("\n~ ", " ~\n");

            // consolidates tied quarters for h16 (very tricky)
            if(drawAmps, {var replaceFirst, tmp;
                replaceFirst = { arg string, find, replace = "", offset = 0;
                    var index, out = "", array = string, findSize = max(find.size, 1);
                    index = array.find(find, offset: offset);
                    out = out ++ array.keep(index) ++ replace;
                    array = array.drop(index + findSize);
                    out ++ array};

                tmp = lilyString;
                lilyString.split($|).do({arg bar; var beats, groups, offset = 0;
                    beats = bar.split($\n);
                    groups = beats.separate({arg a, b; (a.contains("b'4") && a.contains("~") && b.contains("b'4")).not });
                    groups.do({arg group;
                        if(group.size > 1, {
                            var ampList, noteLength, tweak, tie, replace, find;
                            ampList = group.collect({arg note; note[(note.find("(") + 1)..(note.find(")") - 1)]}).join(" ");
                            noteLength = switch(group.size, 2, {"2 "}, 3, {"2. "}, 4, {"1 "});
                            tweak = if(group[0].contains("tweak"), {group[0][(group[0].find("-\\tweak"))..(group[0].find("#-5.5") + 8)]}, {""});
                            tie = if(group.last.contains("~"), {"~ "}, {" "});
                            replace = "\\override Glissando.details.colors = #'(" ++ ampList ++ ") b'" ++ noteLength ++ tweak ++ tie;
```

```
                            find = group.join("\n");
                            tmp = replaceFirst.value(tmp, find, replace, offset)});
                        offset = lilyString.find("|", offset: offset)})});
                lilyString = tmp});


            // final lilypond directives and write files
            if(drawAmps, {
                lilyString = "\\override Glissando.stencil = #my—gliss \\addGliss { \\override Glissando.details.thickness = 1.15 " ++ lilyString ++ " }"});
            lilyString = "{ " ++ lilyString ++ " \\bar \"|.\" \\stopStaff \\hide b'1}";
            lilyFile.write(lilyString);
            lilyFile.close;

            // this deletes the tmp files and creates the pdfs on the last call.
            if(drawAmps, {
                File.delete(ampFilePath);
                if(p == 3, {
                    "which lilypond".unixCmd({ arg res1, pid1;
                        if(res1 == 0, {
                            PathName(basePath).files.do({arg path;
                                ("lilypond —o " ++ basePath +/+ ".." +/+ "pdf " ++ path.fullPath).unixCmd({ arg res2, pid2;
                                    if(res2 != 0, {{var win, text;
                                        win = Window.new.front;
                                        text = StaticText(win, Rect(10, 10, 300, 200));
                                        text.string = "Something went wrong creating the pdfs, \n" ++
                                        "but the lilypond files likely generated just fine"}.defer});
                                    [\done, res2, pid2].postln})})}, {{var win, text;
                            win = Window.new.front;
                            text = StaticText(win, Rect(10, 10, 300, 200));
                            text.string = "Something went wrong creating the pdfs. \nLikely Lilypond is not installed, \n" ++
                            "but the Lilypond files probably generated just fine."}.defer});
                        [\done, res1, pid1].postln})})})});
};
)
```

```
(
var modelView, transportView, mixerView, guiLayout, playButton, randButton;

// model / transform
modelView = {var transformGrid, transformButtons, layouts, layoutGrid, layoutButtons,
    fullScreenButton, decoratorButton, transcribeButton, contPlayButton, exportButton, displayGrid, model;

    // sets transform
    transformGrid = GridLayout();
    transformButtons = Array.fill2D(7, 4, {arg r, c; Button().states_(
        [[c.asString, Color.grey, Color.white], [c.asString, Color.white, Color.grey]])});
    transformButtons.do({arg row, rIndex;
        row.do({arg button, cIndex;
            if((rIndex < 5) || (cIndex < 2), {
                transformGrid.add(button, rIndex, cIndex + 1);
                button.action = {transformButtons[rIndex].do(
                    {arg b, i; if(button == b, {b.value = 1}, {b.value = 0})});
                ~transform[rIndex] = cIndex; if(rIndex < 4, {~tileMap = ~mapAll.value(6, ~transform)}); ~tileWin.refresh}.inEnvir})});
        transformGrid.add(if(rIndex < 4,
            {StaticText().string_("child " ++ rIndex.asString).align_(\center)},
            {StaticText().string_(["rotation", "mirror", "invert"][rIndex - 4])
                .align_(\center)}), rIndex, 0)});
    ~transform.do({arg v, rIndex; transformButtons[rIndex][v].value = 1});

    // sets layout
    layouts = ["depth level 6 only (max depth)", "depth level 5 only", "depth level 4 only", "hierarchical (embedded)",
        "", "", ""];
    layoutGrid = GridLayout();
    layoutButtons = layouts.collect({arg l, i; Button().states_(
        [[layouts[i], Color.grey, Color.white], [l, Color.white, Color.grey]])});
    layoutButtons.do({arg button, bIndex; if(bIndex < 4,
        {layoutGrid.add(button, bIndex, 0);
        button.action = {layoutButtons.do({arg b; if(button == b, {b.value = 1}, {b.value = 0});
            ~layoutState = bIndex; ~tileWin.refresh})}.inEnvir},
        {layoutGrid.add(StaticText().string_(""), bIndex, 0)})});
    layoutButtons[0].value = 1;

    randButton = Button().states_([["random transform", Color.grey, Color.white]]).action_({
        ~transform = Array.fill(7, {arg i; if(i < 5, {4.rand}, {2.rand})});
        ~tileMap = ~mapAll.value(6, ~transform);
        ~tileWin.refresh;
        transformButtons.do({arg row; row.do({arg button; button.value = 0})});
        ~transform.do({arg v, rIndex; transformButtons[rIndex][v].value = 1})}.inEnvir);

    // sets display
    fullScreenButton = Button().states_([["fullscreen", Color.grey, Color.white],
        ["fullscreen", Color.white, Color.grey]]).action_({arg button;
        if(button.value == 0, {~tileWin.endFullScreen}, {~tileWin.fullScreen})}.inEnvir);

    ~tileWin.view.keyDownAction = {arg doc, char, mod, unicode, keycode, key;
        case
        // <ctrl + f> = enter full screen
        {mod == 262144 && key == 70} {~tileWin.fullScreen; fullScreenButton.value = 1}
        // <esc> = exit full screen
        {mod == 0 && key == 16777216} {~tileWin.endFullScreen; fullScreenButton.value = 0}}.inEnvir;

    decoratorButton = Button().states_([["window decorator", Color.grey, Color.white],
        ["window decorator",Color.white, Color.grey]]).action_({arg button;
        ~tileWin.close;
        if(button.value == 0, {~launchTileVisualizer.value(false, false)},
            {~launchTileVisualizer.value(true, false)})}.inEnvir);
    decoratorButton.value = 1;

    // set continuous play option (disables transform buttons when set)
    contPlayButton = Button().states_([["continuous play", Color.grey, Color.white],
        ["continuous play", Color.white, Color.grey]]).action_({arg elem;
        if(elem.value == 0, {~continuousPlay = false;
            transformButtons.do({arg row; row.do({arg button; button.enabled = true})});
            randButton.enabled = true}, {
            transformButtons.do({arg row; row.do({arg button; button.enabled = false})});
            randButton.enabled = false;
            if(playButton.value == 0, {
                fork({
                    ~continuousPlay = true;
                    randButton.valueAction = 0;
                    1.wait;
                    playButton.valueAction = 1}, AppClock)})})}.inEnvir);

    // transcribe
    transcribeButton = Button().states_([["transcribe", Color.grey, Color.white]]).action_({
        var matrixH4, matrixH5, matrixH6,dataH4, dataH5, dataH6;
        matrixH4 = ~matricize.value([~tileMap[4]], 4, 0, ~transform);
        matrixH5 = ~matricize.value([~tileMap[5]], 5, 0, ~transform);
        matrixH6 = ~matricize.value([~tileMap[6]], 6, 0, ~transform);
        dataH4 = ~genData.value(matrixH4, ~transform, 1, 0, 1);
        dataH5 = ~genData.value(matrixH5, ~transform, 1, 0, 4);
        dataH6 = ~genData.value(matrixH6, ~transform, 4, 0, 16);
        ~transcribe.value(dataH4, 16, "h1_4");
        ~transcribe.value(dataH5, 4, "h1_5");
        ~transcribe.value(dataH6, 1, "h1_6", true)}.inEnvir);

    // export tile image
    exportButton = Button().states_([["export image", Color.grey, Color.white]]).action_({var img;
        img = if(~transform[4] % 2 == 0, {Image.new(6000, 3000)}, {Image.new(3000, 6000)});
        img.draw({~tileDrawFunc.value(img, ~tileMap, ~transform, ~layoutState)});
        FileDialog({ arg path; img.write(path[0])}, {}, 0, 1)}.inEnvir);

    displayGrid = GridLayout();
    displayGrid.add(fullScreenButton, 0, 0);
    displayGrid.add(decoratorButton, 1, 0);
    5.do({arg i; displayGrid.add(StaticText().string_(""), i + 2, 0)});

    model = GridLayout();
    model.add(StaticText().string_("transform code"), 0, 0);
    model.add(transformGrid, 1, 0); model.add(randButton, 2, 0);
    model.add(StaticText().string_(""), 3, 0); model.add(StaticText().string_(""), 4, 0);
    model.add(StaticText().string_(""), 0, 1);
    model.add(StaticText().string_("layout"), 0, 2);
    model.add(layoutGrid, 1, 2);
    model.add(StaticText().string_(""), 0, 3);
    model.add(StaticText().string_("display"), 0, 4);
    model.add(displayGrid, 1, 4); model.add(contPlayButton, 2, 4); model.add(transcribeButton, 3, 4); model.add(exportButton, 4, 4);
    model};

// transport
transportView = {
    arg hash; var clockStringFunc, metronomeStringFunc, metronomeColorFunc,
    transport, posSlider, startPos = 0, startPosText, pauseButton, clock, metronome;

    // update clock and metronome
    clockStringFunc = {
```

```
        arg curBeat; var measure, beat;
        beat = (((curBeat / 2) % 4) + 1).trunc.asString;
        measure = ((curBeat / 8) + 1).trunc.asString;
        if(measure.size == 1, {measure = "  " ++ measure});
        if(measure.size == 2, {measure = " " ++ measure});
        measure ++ ":" ++ beat};
    metronomeStringFunc = {arg curBeat; case {curBeat % 8 < 1} {"   "} {curBeat % 2 < 1} {"   "} {true} {" "}};
    metronomeColorFunc = {arg curBeat; case {curBeat % 32 < 1} {Color.red} {curBeat % 8 < 1} {Color.blue} {true} {Color.black}};

    // get current time and call update clock and metronome
    OSCFunc({arg msg, time; {
        if(msg[2] == hash,{
            if(msg[3] > 0, {clock.string = clockStringFunc.value(msg[3])});
            metronome.stringColor = metronomeColorFunc.value(msg[3]);
            metronome.string = metronomeStringFunc.value(msg[3]);
            // if continuous play, when finished, delete all autogenerated files, reselect a transform, and then start
            if(msg[3] == 1031, {
                playButton.valueAction = 0;
                fork({
                    if(~continuousPlay, {var baseDir;
                        baseDir = ~dir +/+ ".." +/+ "audio" +/+ "transform_" ++ ~transform.join ++ "_audio";
                        File.delete(baseDir +/+ "hl_6_fundamental.wav");
                        File.delete(baseDir +/+ "hl_6_harmonics.wav");
                        File.delete(baseDir +/+ "hl_5_high_noise.wav");
                        File.delete(baseDir +/+ "hl_4_low_noise.wav");
                        File.delete(baseDir);
                        randButton.valueAction = 0;
                        posSlider.valueAction = 0;
                        2.wait;
                        playButton.valueAction = 1
                    })}, AppClock)})})}.inEnvir.defer}.inEnvir,'/tr', s.addr);

    // transport gui items
    transport = HLayout(
        // play / stop button
        playButton = Button().states_([["play", Color.black], ["stop", Color.black, Color.grey]]).action_({arg elem;
            fork {var cond, baseDir;
                cond = Condition.new;
                baseDir = ~dir +/+ ".." +/+ "audio" +/+ "transform_" ++ ~transform.join ++ "_audio";
                // generate audio files if they do not exist
                if(File.exists(baseDir +/+ "hl_6_fundamental.wav").not ||
                        File.exists(baseDir +/+ "hl_6_harmonics.wav").not ||
                        File.exists(baseDir +/+ "hl_5_high_noise.wav").not ||
                        File.exists(baseDir +/+ "hl_4_low_noise.wav").not, {
                            var matrixH4, matrixH5, matrixH6, dataH4, dataH5, dataH6,
                            patternH4, patternH5, patternH6Fund, patternH6Harms;
                            matrixH4 = ~matricize.value([~tileMap[4]], 4, 0, ~transform);
                            matrixH5 = ~matricize.value([~tileMap[5]], 5, 0, ~transform);
                            matrixH6 = ~matricize.value([~tileMap[6]], 6, 0, ~transform);
                            dataH4 = ~genData.value(matrixH4, ~transform, 1, 0, 1, 0.5, 1);
                            dataH5 = ~genData.value(matrixH5, ~transform, 1, 0, 4, 0.5, 1);
                            dataH6 = ~genData.value(matrixH6, ~transform, 4, 0, 16);
                            patternH4 = ~genPattern.value(dataH4, "hl_4_low_noise", 16, true, cond); cond.hang;
                            patternH5 = ~genPattern.value(dataH5, "hl_5_high_noise", 4, true, cond); cond.hang;
                            patternH6Fund = ~genPattern.value([dataH6[0]], "hl_6_fundamental", 1, true, cond); cond.hang;
                            patternH6Harms = ~genPattern.value(dataH6[1..3], "hl_6_harmonics", 1, true, cond); cond.hang});

                // load buffers if the transform has changed
                if(~loadedTransform != ~transform, {var baseDir;
                    baseDir = ~dir +/+ ".." +/+ "audio" +/+ "transform_" ++ ~transform.join ++ "_audio";
                    ~buf1 = Buffer.read(s, baseDir +/+ "hl_6_fundamental.wav", action: {cond.unhang}); cond.hang;
                    ~buf2 = Buffer.read(s, baseDir +/+ "hl_6_harmonics.wav", action: {cond.unhang}); cond.hang;
                    ~buf3 = Buffer.read(s, baseDir +/+ "hl_5_high_noise.wav", action: {cond.unhang}); cond.hang;
                    ~buf4 = Buffer.read(s, baseDir +/+ "hl_4_low_noise.wav", action: {cond.unhang}); cond.hang});
                    ~loadedTransform = ~transform.deepCopy;


                // play / stop functionality (create synth if it does not exist
                {if(~play == nil, {~play = Synth.new(\play, [\hash, hash, \playRate, 0, \startTrig, 0,
                        \sineBuf1, ~buf1, \sineBuf2, ~buf2, \highNoiseBuf, ~buf3, \lowNoiseBuf, ~buf4])});
                    pauseButton.value = 0;
                    if(elem.value == 0, {
                        clock.string = clockStringFunc.value((startPos * 129).trunc * 8);
                        ~play.set(\playRate, 0, \startTrig, 0,
                            \sineBuf1, ~buf1, \sineBuf2, ~buf2, \highNoiseBuf, ~buf3, \lowNoiseBuf, ~buf4);
                        clock.string = clockStringFunc.value((startPos * 129).trunc * 8)},{
                        ~play.set(\startPos, startPos, \playRate, 1, \startTrig, 1,
                            \sineBuf1, ~buf1, \sineBuf2, ~buf2, \highNoiseBuf, ~buf3, \lowNoiseBuf, ~buf4)})}.inEnvir.defer}}.inEnvir),

        // pause button
        pauseButton = Button().states_([["pause", Color.black], ["pause", Color.black, Color.grey]]).action_({arg elem;
            if(elem.value == 1, {~play.set(\playRate, 0)},{~play.set(\playRate, 1)})}.inEnvir),

        // start position slider
        StaticText().string_("    start measure:beat"),[
            posSlider = Slider(bounds: Rect(0, 0, 30, 5)).action_({arg pos; var min, sec;
            startPosText.string = clockStringFunc.value((pos.value * 129).trunc * 8);
            startPos = ((pos.value * 129).trunc * 8) / (129 * 8);
            clock.string = clockStringFunc.value((startPos * 129).trunc * 8)}.inEnvir), stretch: 1],
        startPosText = StaticText().string_("  1:1").font_(Font("Monaco", 15)), nil);

    HLayout([VLayout(HLayout(
        clock = StaticText().string_("  1:1").font_(Font("Monaco", 200)),
        StaticText().string_("|").font_(Font("Monaco", 200)),
        metronome = StaticText().string_("   ").font_(Font("Monaco", 300)).stringColor_(Color.red)), nil, transport),
    alignment: \top])};

// mixer
mixerView = {arg hash; var masterIndicators, trackIndicators, master, tracks,
    eVol = [0.8, 0.8, 0.8, 0.8], eMute = [1, 1, 1, 1], ePan = [0, 0, 0, 0],
    eNames = ["hl 6 (fundamental)", "hl 6 (harmonics)", "hl 5 (high noise)", "hl 4 (low noise)"];

    // init indicators
    masterIndicators = {LevelIndicator()} ! 2;
    trackIndicators = {LevelIndicator()} ! 4;

    // get amp values for indicators
    OSCFunc.new({arg msg; {
        if(msg[2] == hash, {{arg i;
            masterIndicators[i].value = msg[3 + i].ampdb.linlin(-60, 0, 0, 1)} ! 2})}.defer},
    '/masterLevel', s.addr);
    OSCFunc.new({arg msg; {
        if(msg[2] == hash, {{arg i;
            trackIndicators[i].value = msg[3 + i].ampdb.linlin(-60, 0, 0, 1)} ! 4})}.defer},
    '/trackLevels', s.addr);

    // master faders
    master = HLayout(
        VLayout([HLayout(
            Slider().value_(0.8).action_({arg elem;
                ~play.set(\masterVol, elem.value * 1.25)}.inEnvir),
            masterIndicators[0], masterIndicators[1]), stretch: 2],
        Button().states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_({arg elem;
```

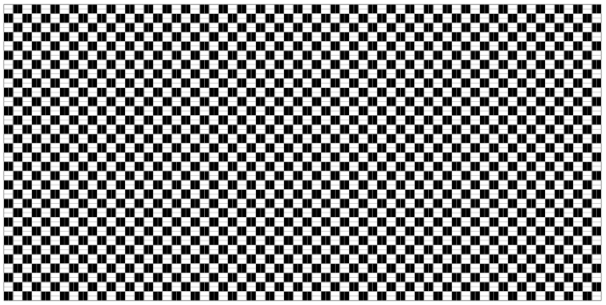
```
            ~play.set(\masterMute, (1 - elem.value).abs)}.inEnvir),
        StaticText().string_("          master          ").align_(\center)), nil);
    // track faders
    tracks = {arg part;
        HLayout(VLayout(HLayout(
            Slider().value_(0.8).action_({arg elem;
                eVol[part] = elem.value * 1.25; ~play.set("eVol", eVol)}.inEnvir),
            trackIndicators[part]),
        Button().states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_({arg elem;
            eMute[part] = (1 - elem.value).abs; ~play.set("eMute", eMute)}.inEnvir),
        StaticText().string_("pan").align_(\center),
        Knob().value_(0.5).action_({arg elem;
            ePan[part] = elem.value * 2 - 1; ~play.set("ePan", ePan)}.inEnvir),
        StaticText().string_(eNames[part]).align_(\center)), nil)} ! 4;
    HLayout(master, nil, *tracks)};

// all gui
guiLayout = {var tabButtonReset, modelViewButton, transportViewButton, mixerViewButton, hash, tabs;
    tabButtonReset = {modelViewButton.value = 1; transportViewButton.value = 1; mixerViewButton.value = 1};
    VLayout(HLayout(
        modelViewButton = Button().states_([["model", Color.white, Color.grey], ["model", Color.black]]).action_(
            {tabButtonReset.value; modelViewButton.value = 0; tabs.index = 0 }).value_(0),
        transportViewButton = Button().states_([["transport", Color.white, Color.grey], ["transport", Color.black]]).action_(
            {tabButtonReset.value; transportViewButton.value = 0; tabs.index = 1 }).value_(1),
        mixerViewButton = Button().states_([["mixer", Color.white, Color.grey], ["mixer", Color.black]]).action_(
            {tabButtonReset.value; mixerViewButton.value = 0; tabs.index = 2 }).value_(1)),
    hash = 10000000.rand;
    tabs = StackLayout();
    tabs.add(Window.new.layout_(mixerView.value(hash)));
    tabs.insert(Window.new.layout_(transportView.value(hash)));
    tabs.insert(Window.new.layout_(modelView.value(hash)));
    tabs.index = 0)};

// launch the gui
~launchGui = {var guiWin;
    guiWin = Window.new("a lot of tiles (trivial scan) - gui", Rect(200, 25, 1000, 450)).front;
    guiWin.layout = guiLayout.value;
    guiWin.front}
)
```

a_lot_of_tiles_trivial_score_template.ly

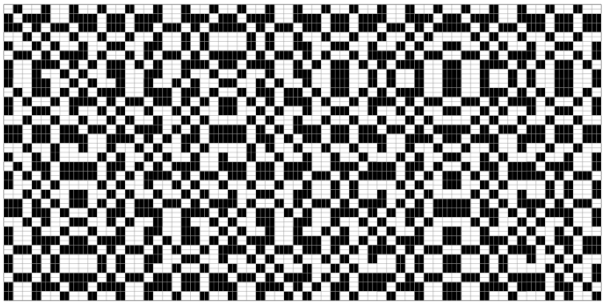
```
\version "2.19.81"

% systems-per-page, unfold vars, and strings delimited by tildes
% are changed by the Supercollider transcriber

\paper {
  #(set-paper-size "a4" 'portrait)
  top-margin = 1 \cm
  bottom-margin = 1 \cm
  left-margin = 2.5 \cm

  top-system-spacing =
  #'((basic-distance . 25 )
  (minimum-distance . 25 )
  (padding . 0 )
  (stretchability . 0))

  last-bottom-spacing =
  #'((basic-distance . 15 )
  (minimum-distance . 15 )
  (padding . 0 )
  (stretchability . 0))

  % manually change systems-per-page to 2 if the harmonics of hl 6 are uncommented below
  systems-per-page = 4

  print-page-number = ##t
  oddHeaderMarkup = \markup { \on-the-fly #not-first-page "(~~score_type~~ - ~~transform~~)" }
  evenHeaderMarkup = \markup { \on-the-fly #not-first-page "(~~score_type~~ - ~~transform~~)" }
  oddFooterMarkup = \markup { \fill-line {
    \on-the-fly #not-first-page
    \concat {
      "-"
      \fontsize #1.5
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
      "-"}}}
  evenFooterMarkup = \markup { \fill-line {
    \on-the-fly #not-first-page
    \concat {
      "-"
      \fontsize #1.5
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
      "-"}}}
}

\header {
  title = \markup { \italic {a lot of tiles (trivial scan)}}
  subtitle = \markup { \normal-text {~~transform~~}}
  composer = \markup \left-column {"michael winter" "(cdmx, mx and nashville, usa; 2018)"}
  piece = \markup { \normal-text {~~score_type~~ - ~~transform~~}}
  tagline = ""
}

#(set-global-staff-size 15)

\layout {
  indent = 0.0\cm
  line-width = 17\cm
  ragged-last = ##t

  \context {
    \Score
      \override BarNumber.extra-offset = #'(0 . 5)
      \override BarNumber.stencil =
      #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  }
  \context {
    \Staff
      \override StaffSymbol.line-count = #1
      \override Clef.stencil = #point-stencil
      \override Clef.space-alist.first-note = #'(extra-space . 1)
      \remove "Time_signature_engraver"
  }
  \context {
    \Voice
      \override Glissando.minimum-length = #0
      \override Glissando.layer = 500
      \override Glissando.breakable = ##t
      \override Glissando.bound-details =
      #'((right
          (attach-dir . -1)
          (end-on-accidental . #f)
          (padding . 0))
         (right-broken
          (padding . 0.5))
         (left-broken
          (padding . 0.5)
          (attach-dir . 1))
         (left
          (attach-dir . -1)
          (padding . -0.25)
          (start-at-dot . #f)))
    }
}

% this draws a curve and a gradient
#(define (make-grey-filled-curve-stencil-list x-coords colors half-thick rl)
  (if (null? (cdr x-coords))
      rl
      (make-grey-filled-curve-stencil-list
        (cdr x-coords)
        (cdr colors)
        half-thick
        (cons
          (stencil-with-color
            (make-filled-box-stencil
              (interval-widen (cons (car x-coords) (cadr x-coords)) 0.1)
              (cons (- half-thick) (- 9 (* 10 (car colors)))))
            (list (car colors) (car colors) (car colors)))
          rl))))

% this draws a just a line with a gradient
#(define (make-grey-filled-box-stencil-list x-coords colors half-thick rl)
  (if (null? (cdr x-coords))
      rl
      (make-grey-filled-box-stencil-list
        (cdr x-coords)
        (cdr colors)
        half-thick
```

```
        (cons
          (stencil-with-color
            (make-filled-box-stencil
              (interval-widen (cons (car x-coords) (cadr x-coords)) 0.1)
              (cons (- half-thick) half-thick))
            (list (car colors) (car colors) (car colors)))
          rl))))

#(define my-gliss
  (lambda (grob)
    (if (ly:stencil? (ly:line-spanner::print grob))
        (let* ((stencil (ly:line-spanner::print grob))
               (X-ext (ly:stencil-extent stencil X))
               (Y-ext (ly:stencil-extent stencil Y))
               (Y-length (- (cdr Y-ext) (car Y-ext)))
               (left-bound-info (ly:grob-property grob 'left-bound-info))
               (left-Y (assoc-get 'Y left-bound-info))
               (thick
                 (assoc-get 'thickness (ly:grob-property grob 'details) 0.5))
               (layout (ly:grob-layout grob))
               (blot (ly:output-def-lookup layout 'blot-diameter))
               (right-bound (ly:spanner-bound grob RIGHT))
               (right-par (ly:grob-parent right-bound X))
               (stem
                 (if (grob::has-interface right-par 'note-column-interface)
                     (ly:grob-object right-par 'stem)
                     '()))
               (stem-stencil
                 (if (ly:grob? stem)
                     (ly:grob-property stem 'stencil)
                     #f))
               (stem-thick
                 (if (ly:stencil? stem-stencil)
                     (interval-length (ly:stencil-extent stem-stencil X))
                     0))
               (corr-delta-X (- (interval-length X-ext)
                    Y-length
                    blot
                    stem-thick
                    ;; mmh, why this value??
                    -0.05))
               (colors
                 (assoc-get 'colors (ly:grob-property grob 'details)
                   (list 0 .5 .8)))
               (steps
                 (length colors))
               (raw-x-coords
                 (iota (1+ (abs steps)) 0 (/ corr-delta-X (abs steps))))
               (x-coords
                 (map
                   (lambda (e)
                     (+ (car X-ext) Y-length blot e))
                   raw-x-coords)))

          ;; create a flat glissando
          (ly:grob-set-nested-property! grob '(right-bound-info Y) left-Y)

          ;; return the stencil of added boxes
          (ly:stencil-translate-axis
             (apply
               ly:stencil-add
               ;; change this to make-grey-filled-box-stencil-list to eleminate curve and just use gradient
               (make-grey-filled-curve-stencil-list
                 x-coords
                 colors
                 thick
                 '()))
            ;; the actual offset is TODO, hardcoded here
              3.5
            Y))
        #f)))

#(define (add-gliss m)
   (case (ly:music-property m 'name)
     ((NoteEvent)
      (set! (ly:music-property m 'articulations)
            (append
              (ly:music-property m 'articulations)
              (list (make-music 'GlissandoEvent))))
      m)
     (else #f)))

addGliss =
#(define-music-function (music)
  (ly:music?)
  (map-some-music add-gliss music))


\new Score
\with {proportionalNotationDuration = #(ly:make-moment 1 16)}
<<
  \new StaffGroup \with{
    \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 13)
       (minimum-distance . 13)
       (padding . 0)
       (stretchability . 0))
    \override StaffGrouper.staffgroup-staff-spacing =
    #'((basic-distance . 13)
       (minimum-distance . 13)
       (padding . 0)
       (stretchability . 0))
  }
  <<

    % uncomment these lines to view harmonics of h1 6
    % also change systems-per-page to 2 manually above
    %{
    \new Staff \with {
      instrumentName = \markup \center-column {"h1 6 " "(harm 13)  "}
      shortInstrumentName = #"h16 (h13)  "
    }
    <<
      \repeat unfold 33 { \repeat unfold 3 { s1 \noBreak } s1 \break }
      \include "includes/a_lot_of_tiles_trivial_scan_h1_6_harm_13.ly"
    >>
    \new Staff \with {
      instrumentName = \markup \center-column {"h1 6 " "(harm 9)   "}
      shortInstrumentName = #"h16 (h9)  "
    }
    <<
      \repeat unfold 33 { \repeat unfold 3 { s1 \noBreak } s1 \break }
      \include "includes/a_lot_of_tiles_trivial_scan_h1_6_harm_9.ly"
```

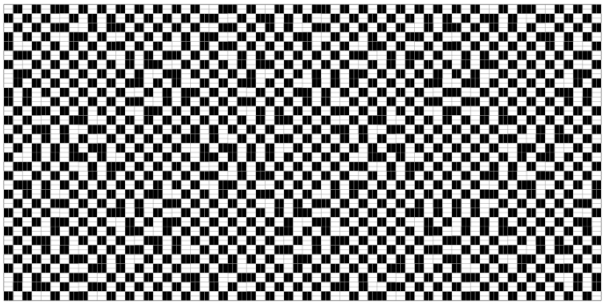
```
    >>
    \new Staff \with {
      instrumentName = \markup \center-column {"hl 6 " "(harm 5)   "}
      shortInstrumentName = #"hl6 (h5)  "
    }
    <<
      \repeat unfold 33 { \repeat unfold 3 { s1 \noBreak } s1 \break }
      \include "includes/a_lot_of_tiles_trivial_scan_hl_6_harm_5.ly"
    >>
    %}

     %--hl6-start--
    \new Staff \with {
      instrumentName = \markup \center-column {"hl 6  " "(harm 1)   "}
      shortInstrumentName = #"hl6 (h1)  "
    }
    <<
      \repeat unfold 33 { \repeat unfold 3 { s1 \noBreak } s1 \break }
      \include "includes/a_lot_of_tiles_trivial_scan_hl_6_harm_1.ly"
    >>
    %--hl6-end--
  >>

  \new StaffGroup \with{
    \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 9)
        (minimum-distance . 9)
        (padding . 0)
        (stretchability . 0))
  }
  <<
    %--hl5-start--
    \new Staff \with {
      instrumentName = \markup \center-column {"hl 5" "(high noise)"}
      shortInstrumentName = #"hl5 (hn)  "
    }
    <<
      \repeat unfold 33 { \repeat unfold 3 { s1 \noBreak } s1 \break }
      \include "includes/a_lot_of_tiles_trivial_scan_hl_5.ly"
    >>
    %--hl5-end--

    %--hl4-start--
    \new Staff \with {
      instrumentName = \markup \center-column {"hl 4 " "(low noise)"}
      shortInstrumentName = #"hl4 (ln)  "
    }
    <<
      \repeat unfold 33 { \repeat unfold 3 { s1 \noBreak } s1 \break }
      \include "includes/a_lot_of_tiles_trivial_scan_hl_4.ly"
    >>
    %--hl4-end--
  >>
>>
```



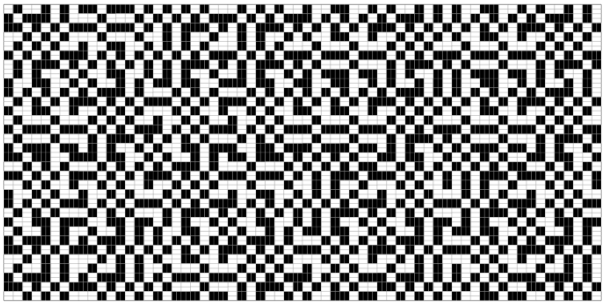
0000



0001

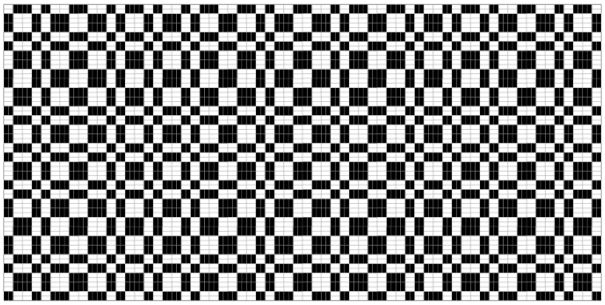


0002

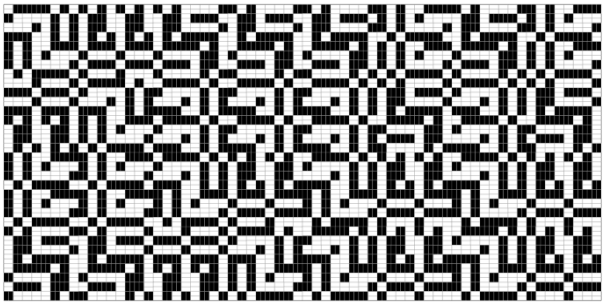


0003

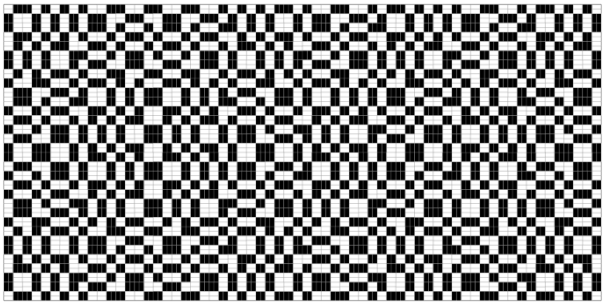
0010



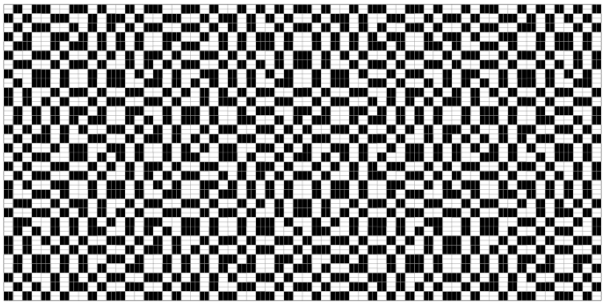
0011



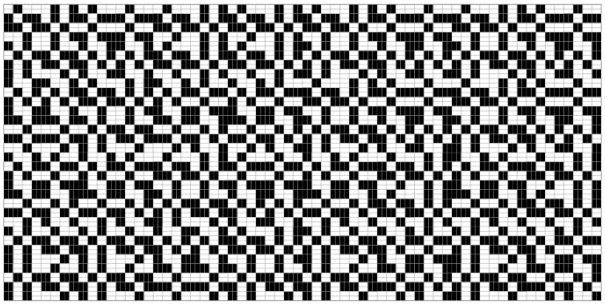
0012



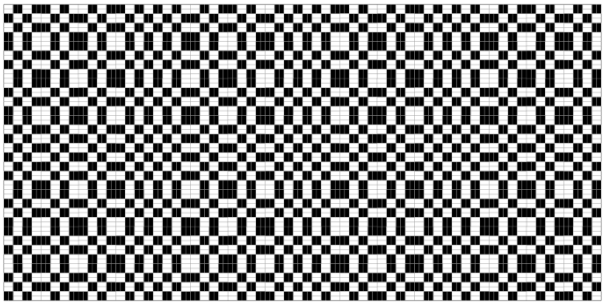
0013



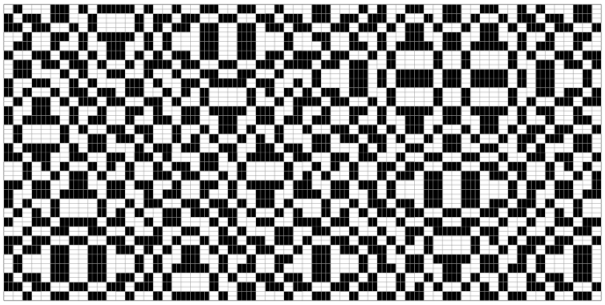
0020



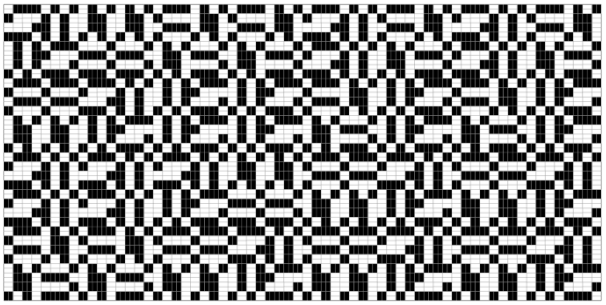
0021



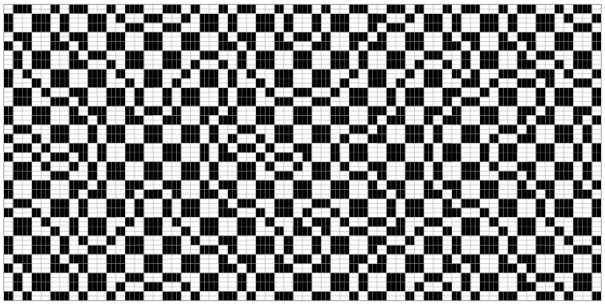
0022



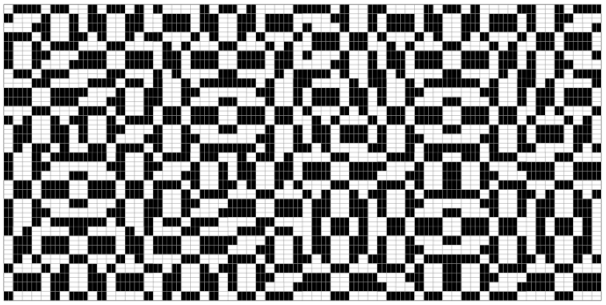
0023



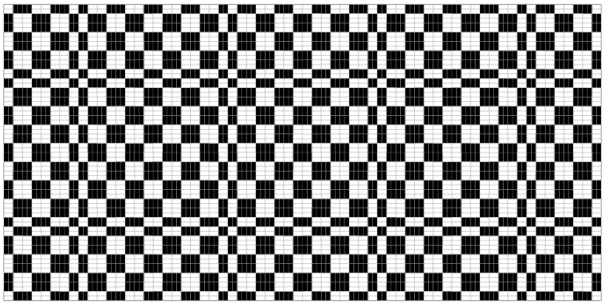
0030



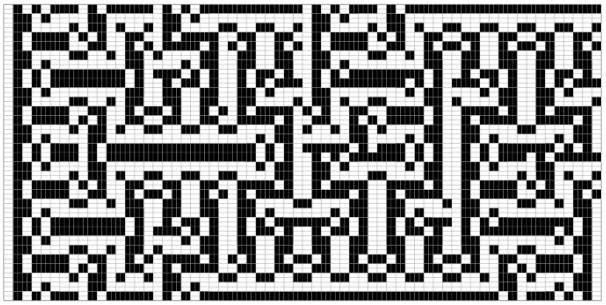
0031



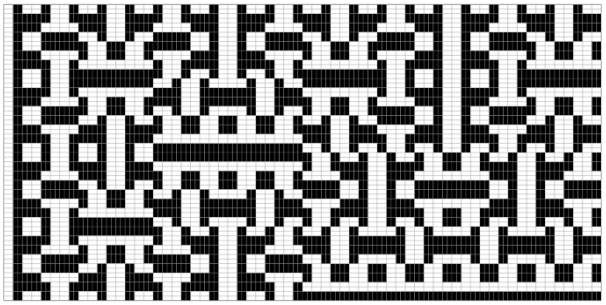
0032



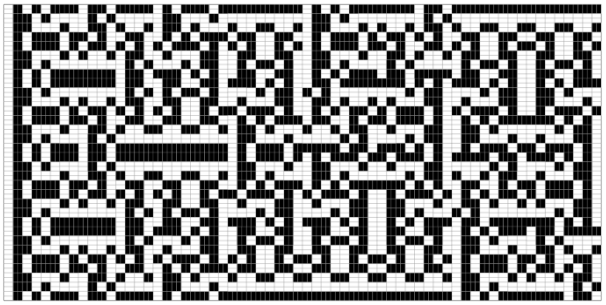
0033



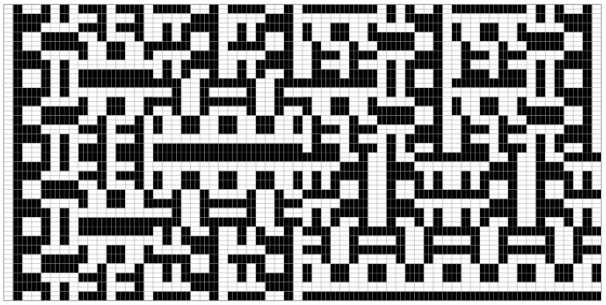
0100



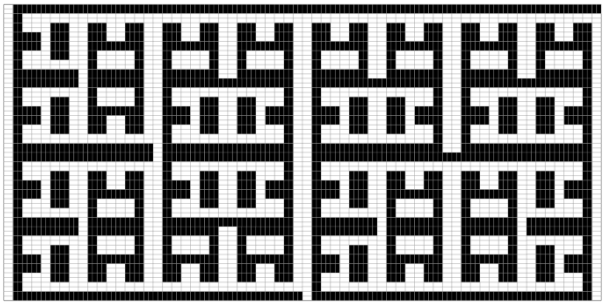
0101



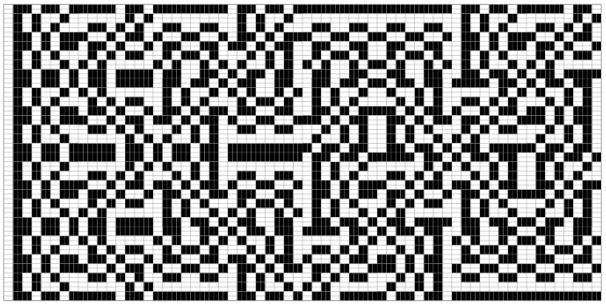
0102



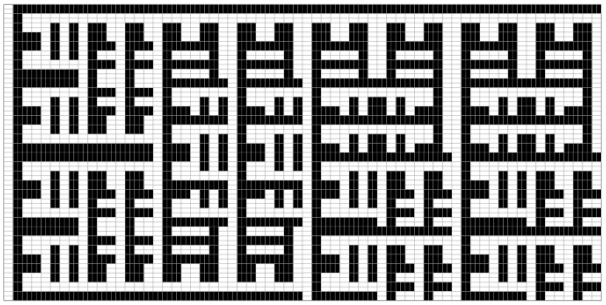
0103



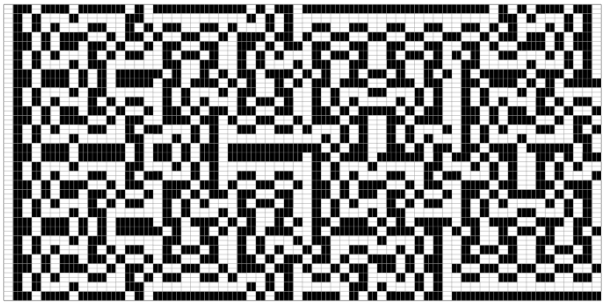
0110



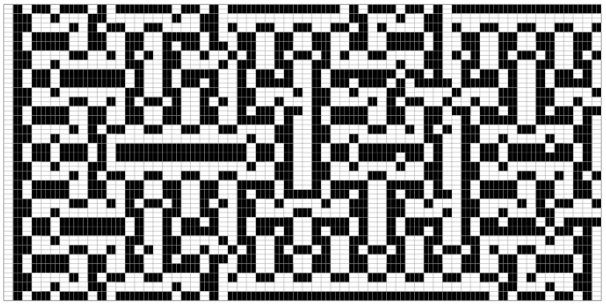
0111



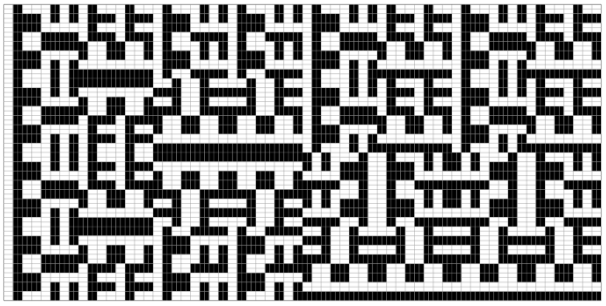
0112



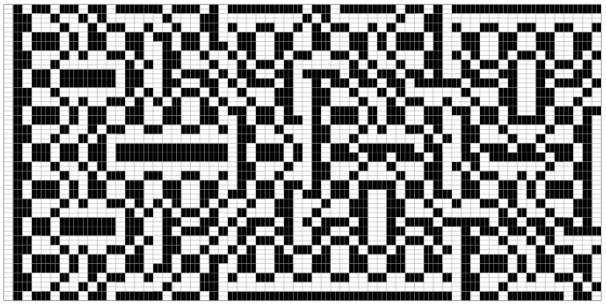
0113



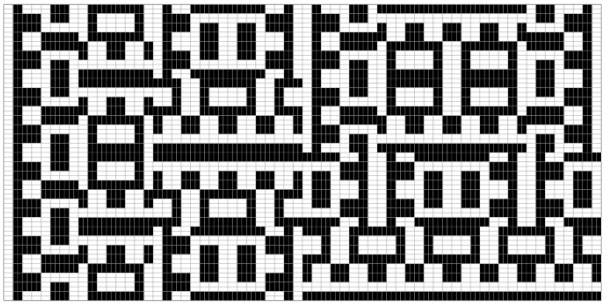
0120



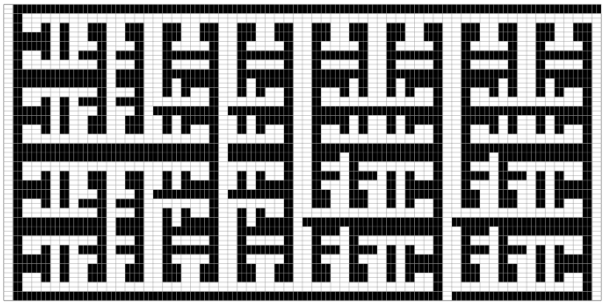
0121



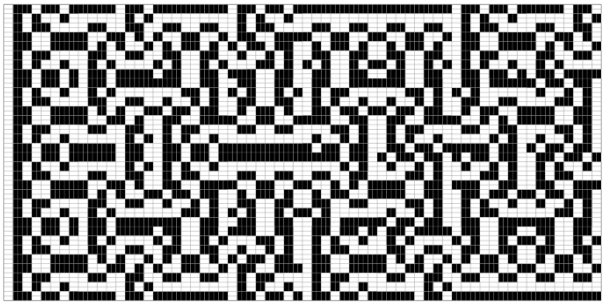
0122



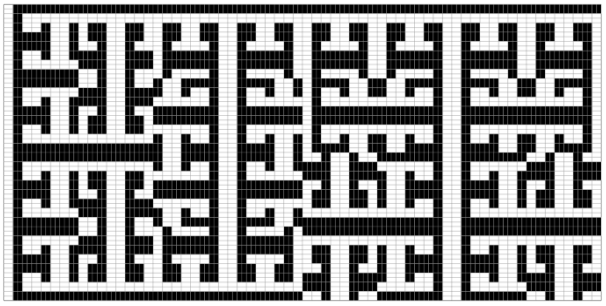
0123



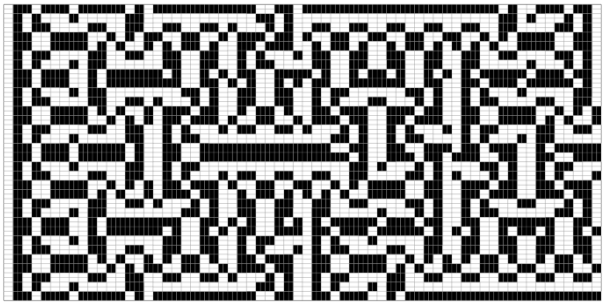
0130



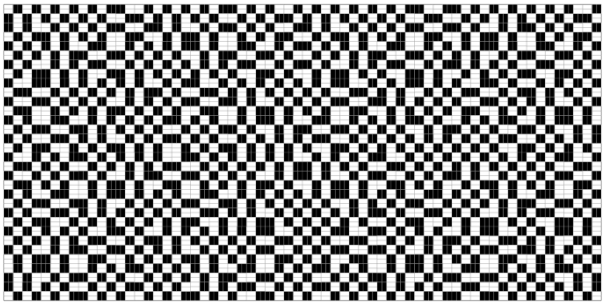
0131



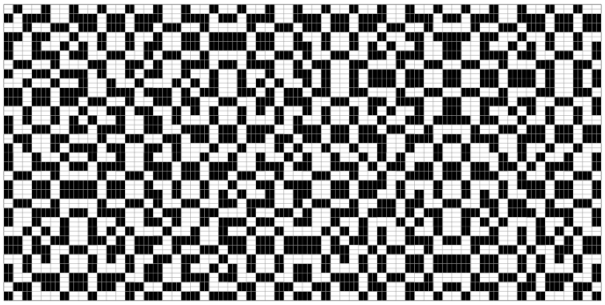
0132



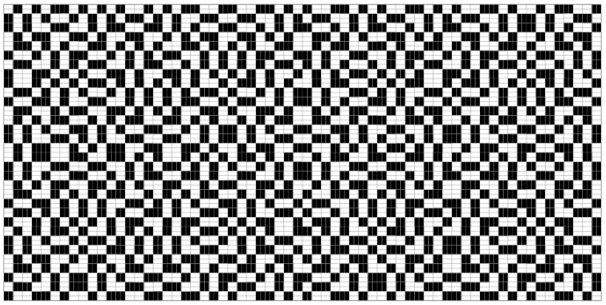
0133



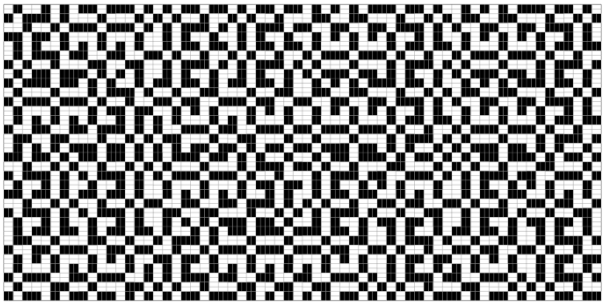
0200



0201

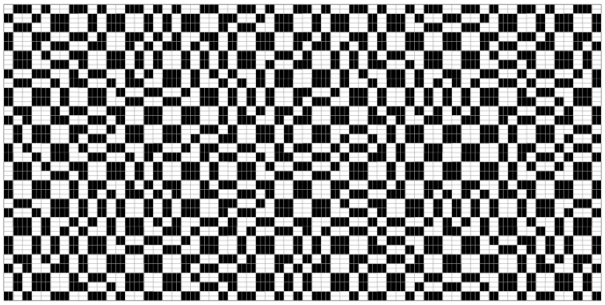


0202

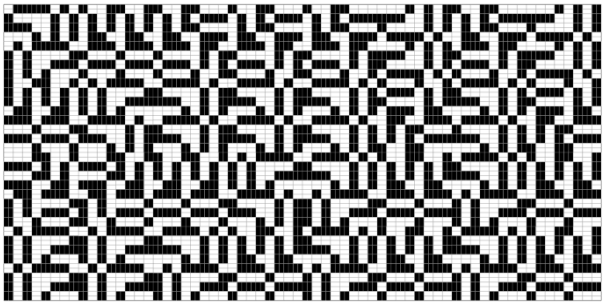


0203

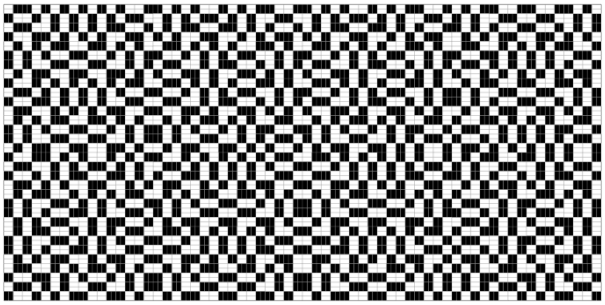
0210



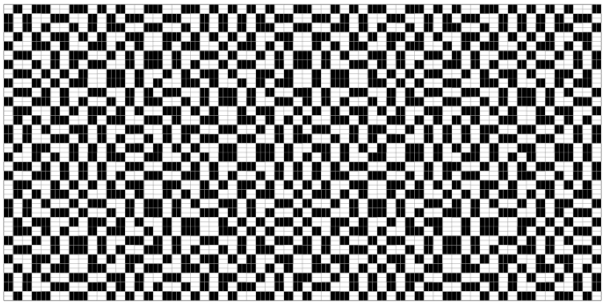
0211



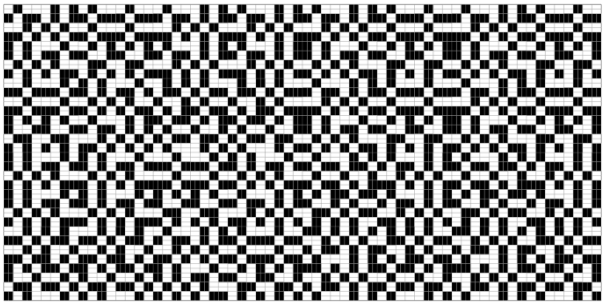
0212



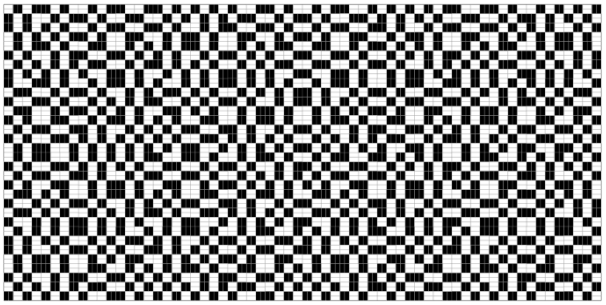
0213



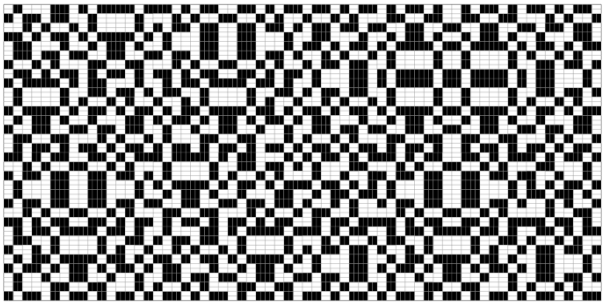
0220



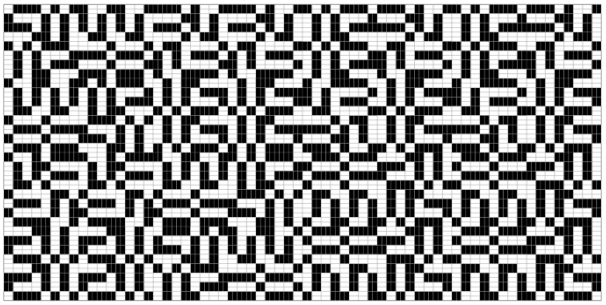
0221



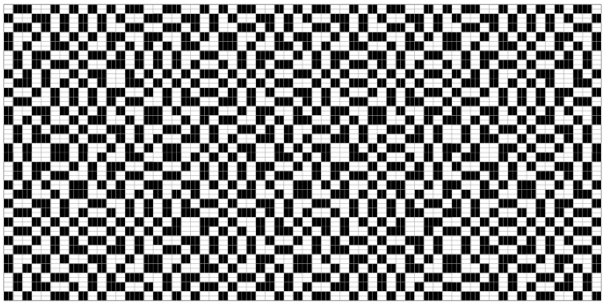
0222



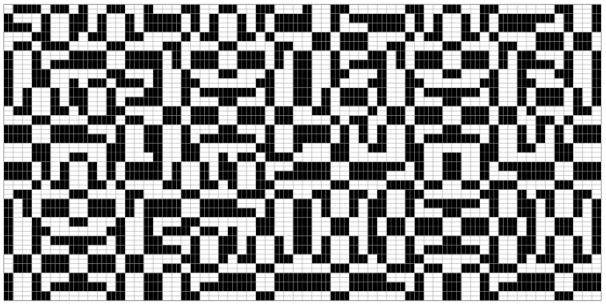
0223



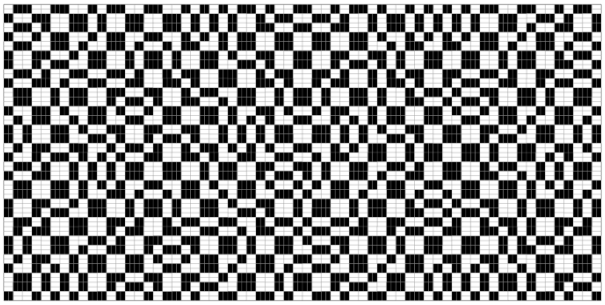
0230



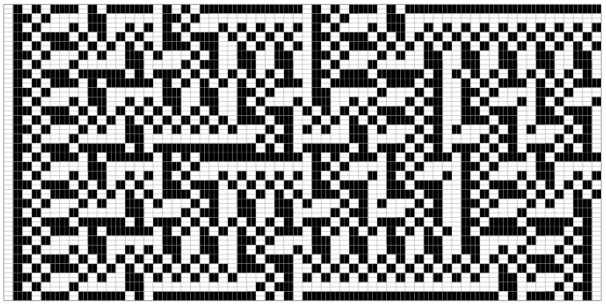
0231



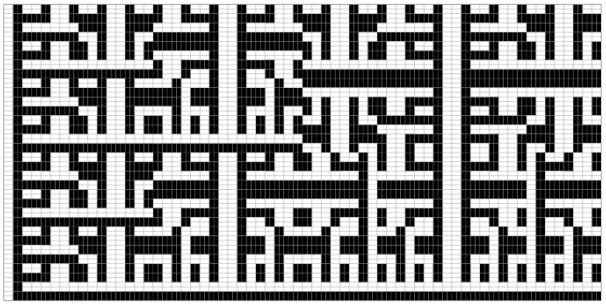
0232



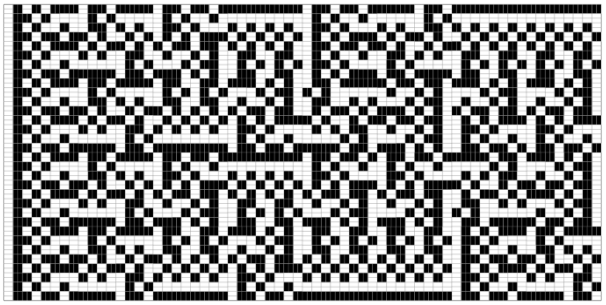
0233



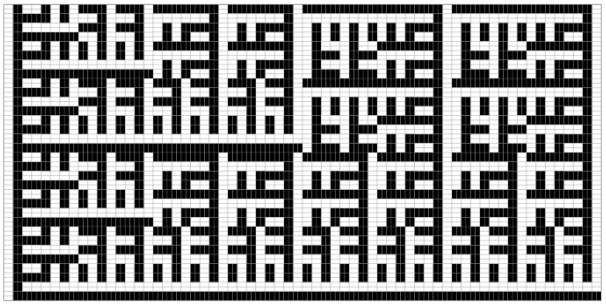
0300



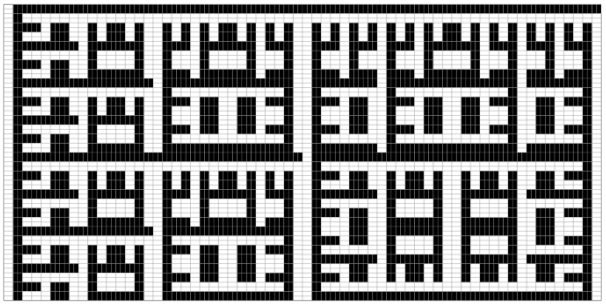
0301



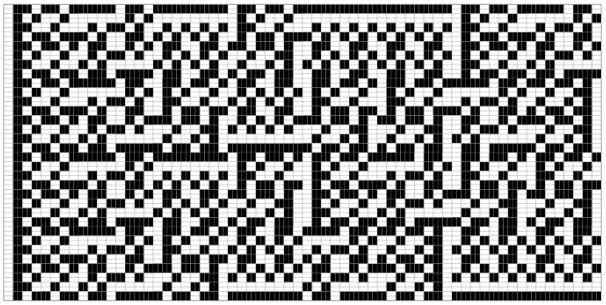
0302



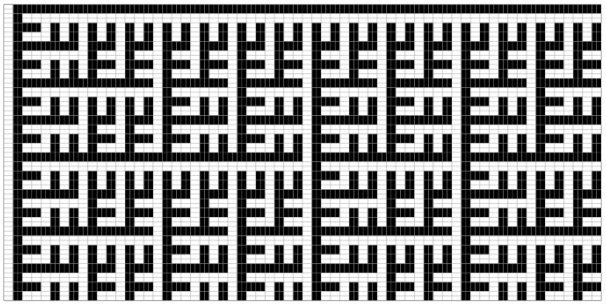
0303



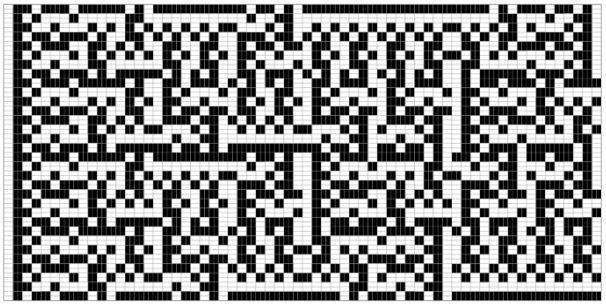
0310



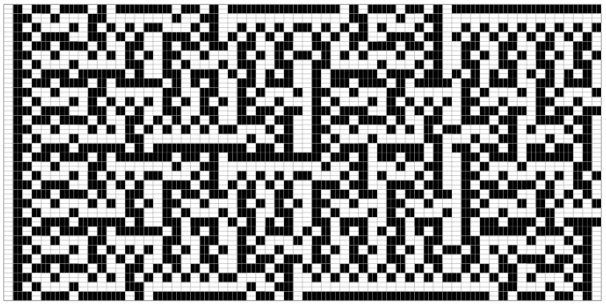
0311



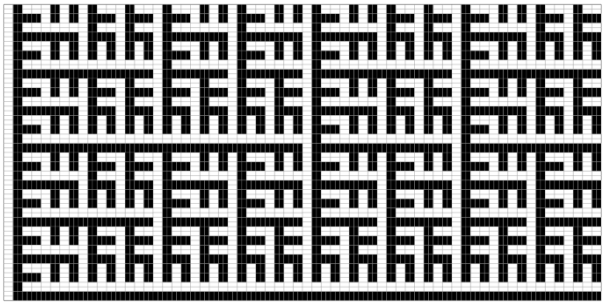
0312



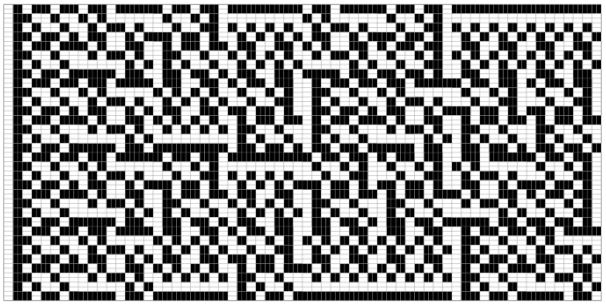
0313



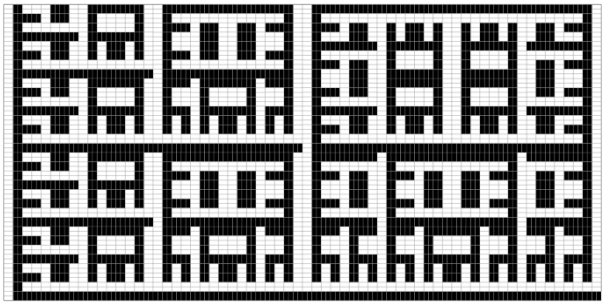
0320



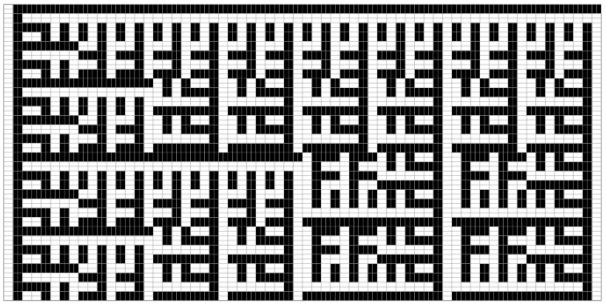
0321



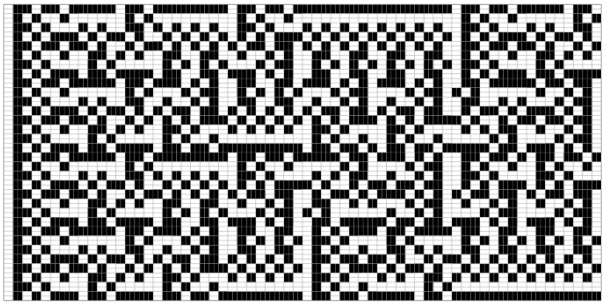
0322



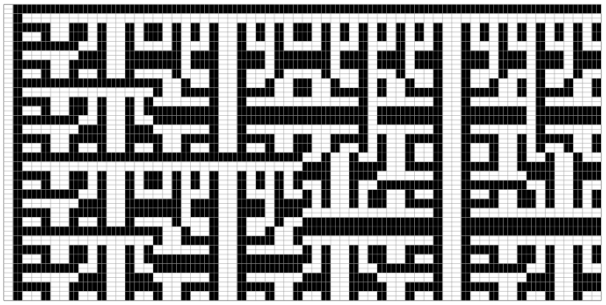
0323



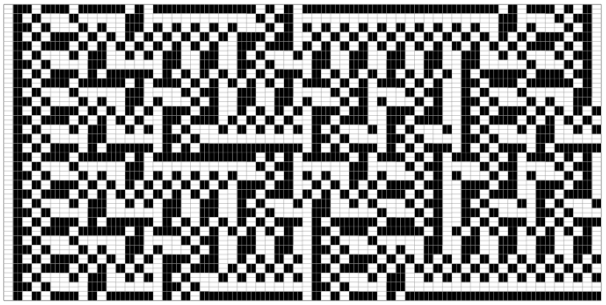
0330



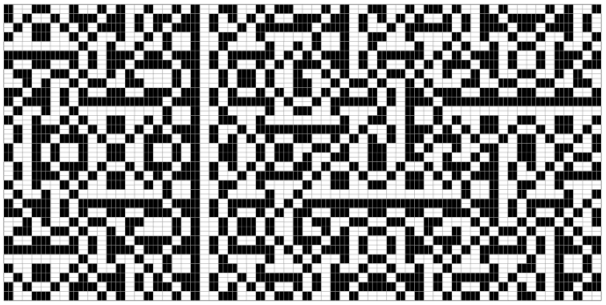
0331



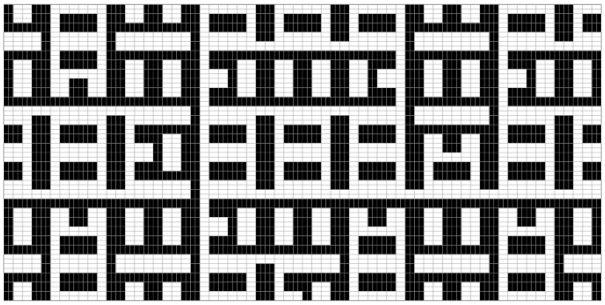
0332



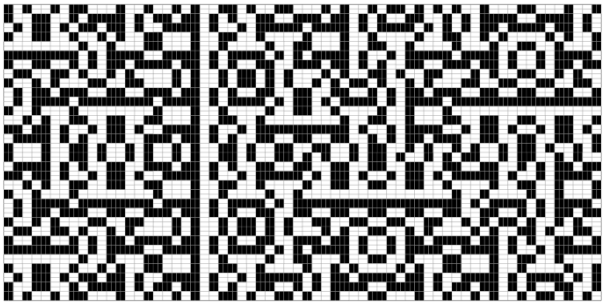
0333



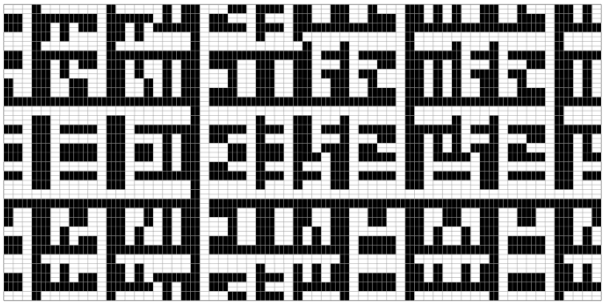
1000



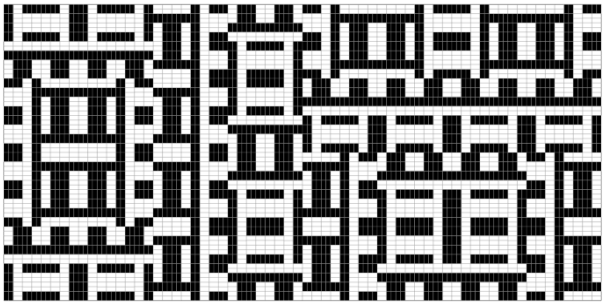
1001



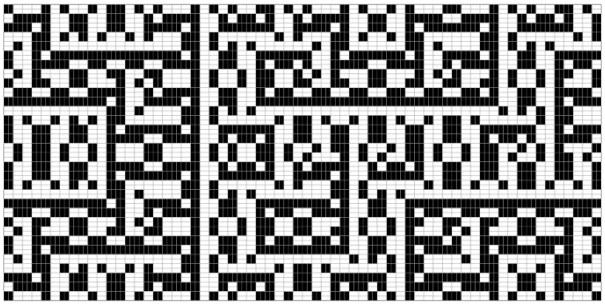
1002



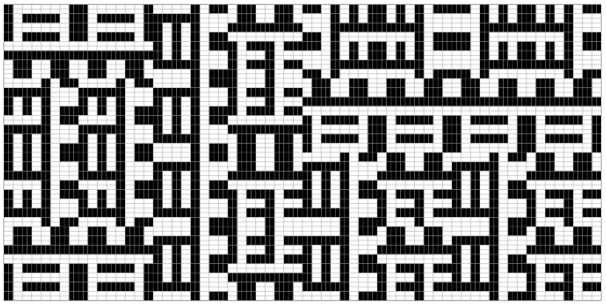
1003



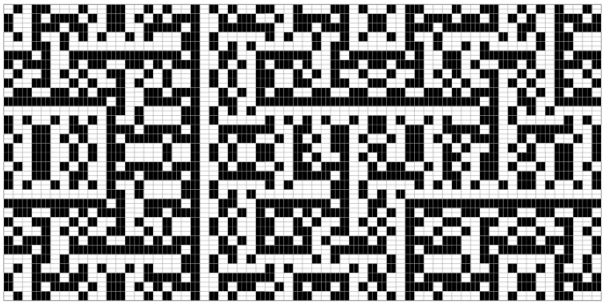
1010



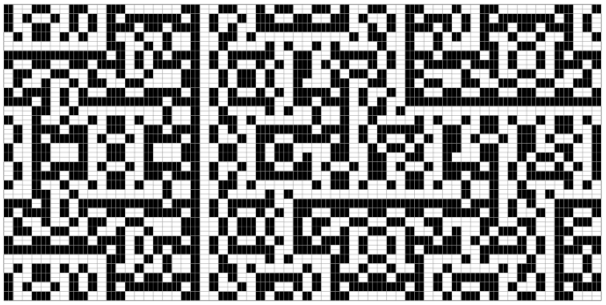
1011



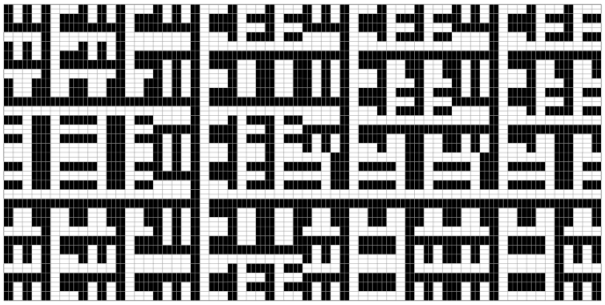
1012



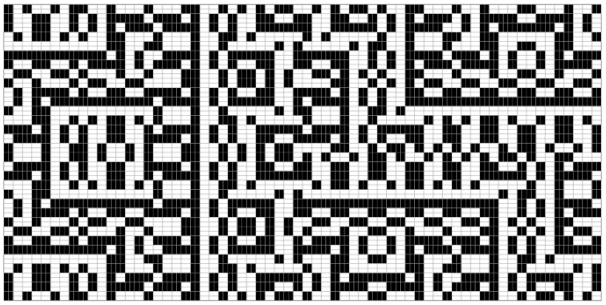
1013



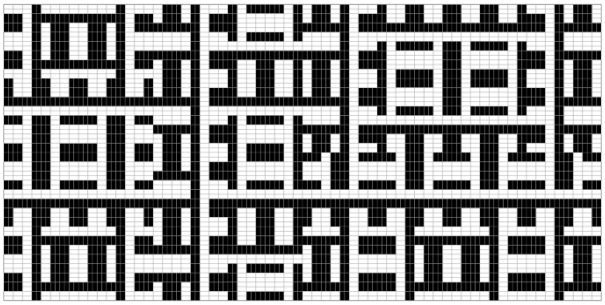
1020



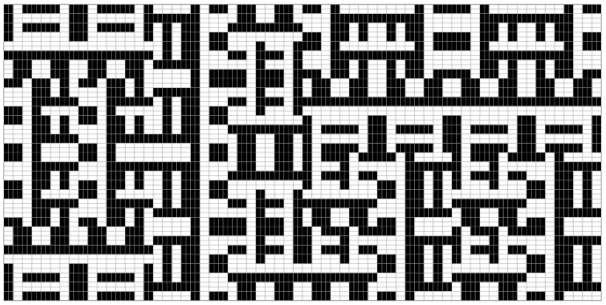
1021



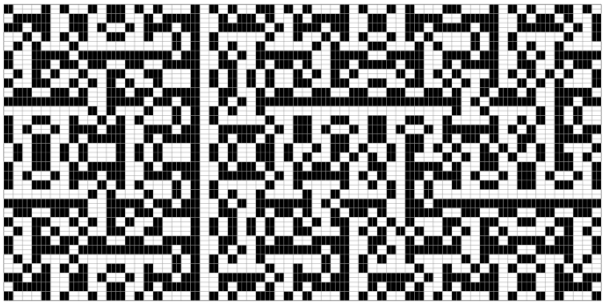
1022



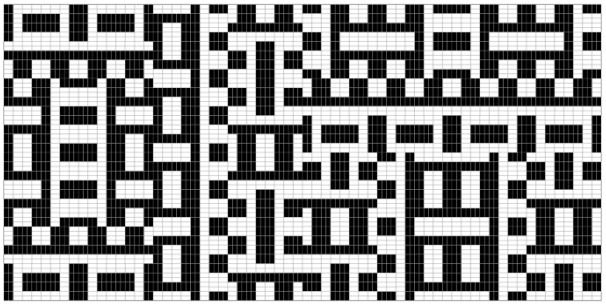
1023



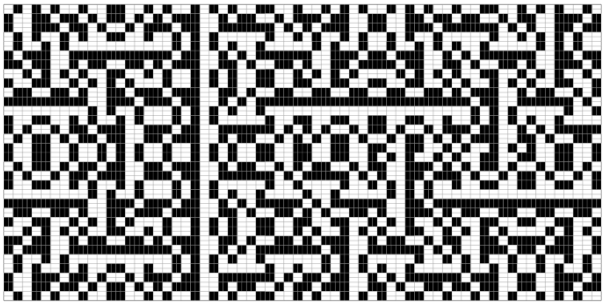
1030



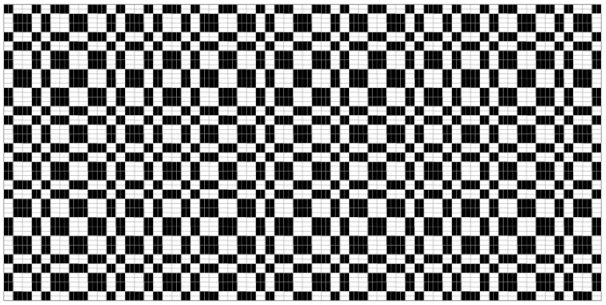
1031



1032

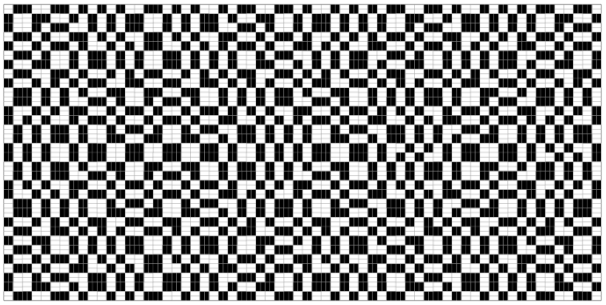


1033

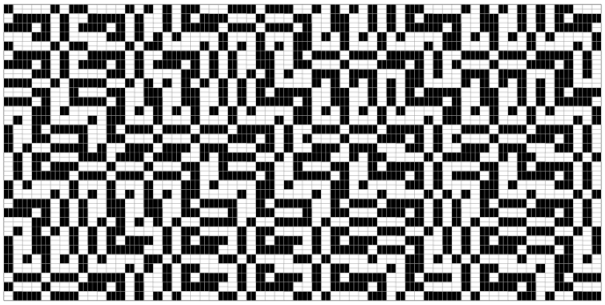


1100

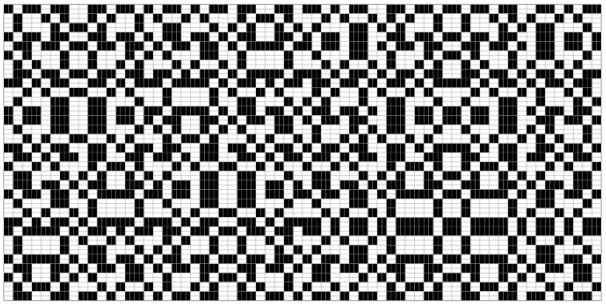
1101



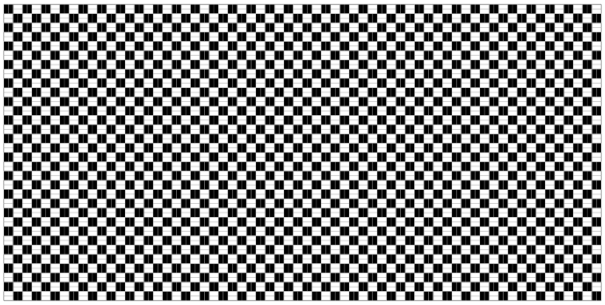
1102



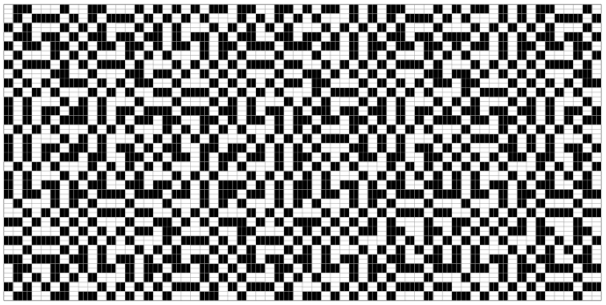
1103



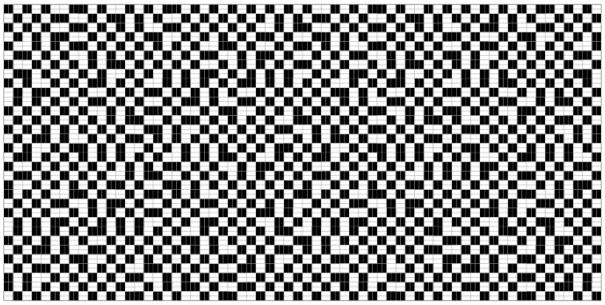
1110



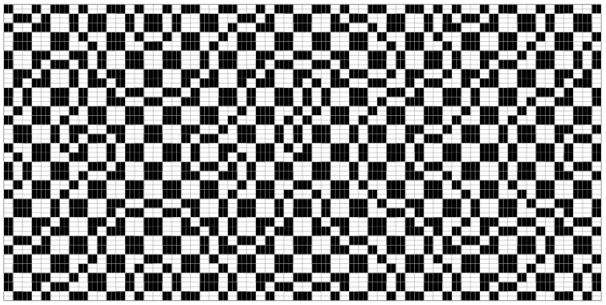
1111



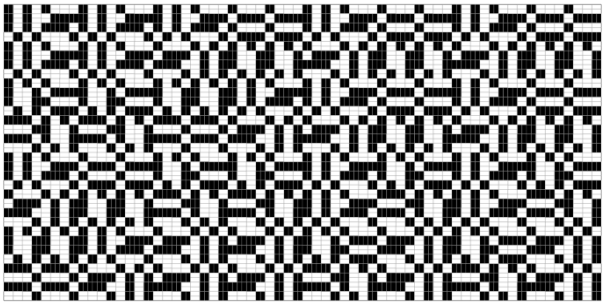
1112



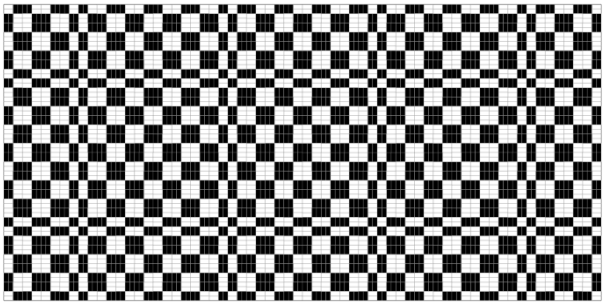
1113



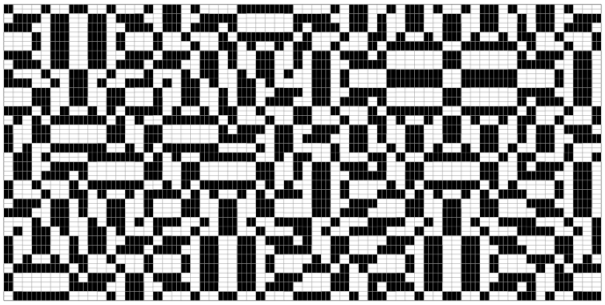
1120



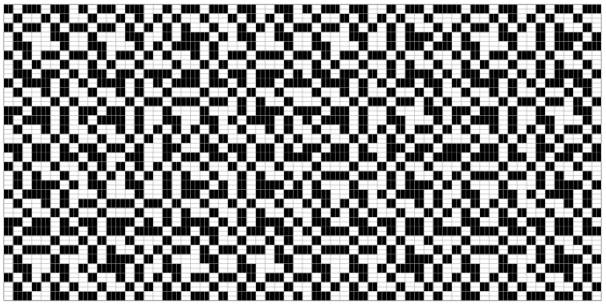
1121



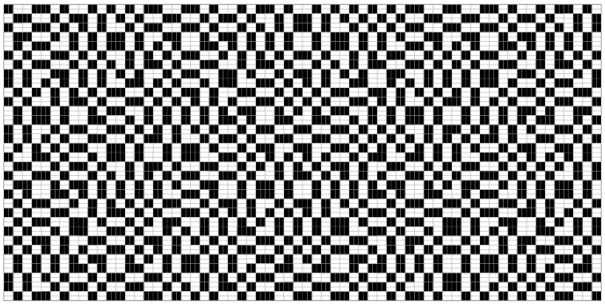
1122



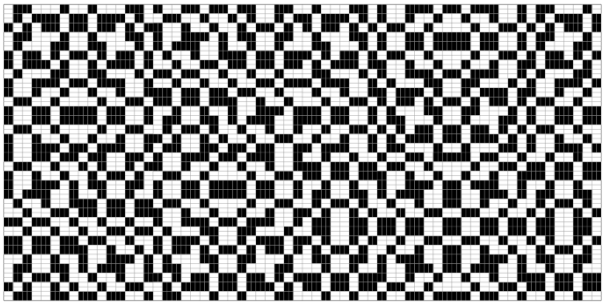
1123



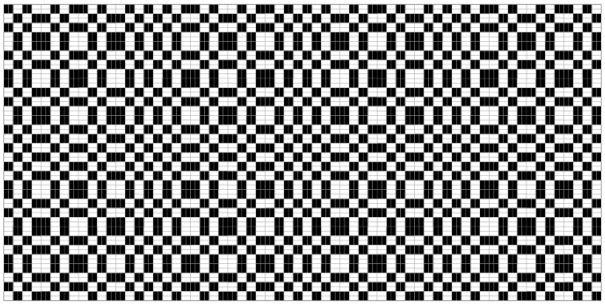
1130



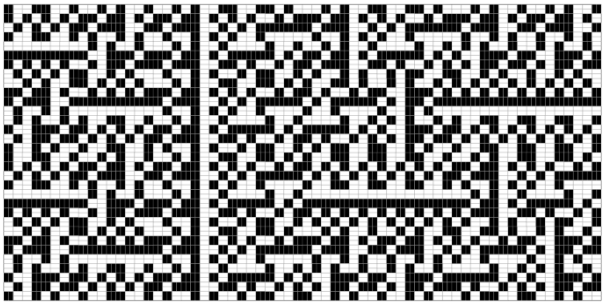
1131



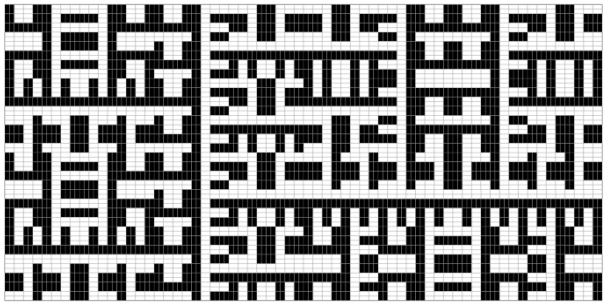
1132



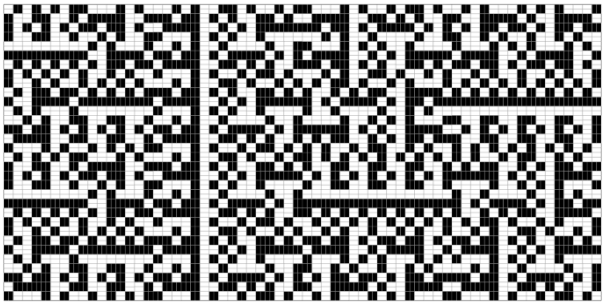
1133



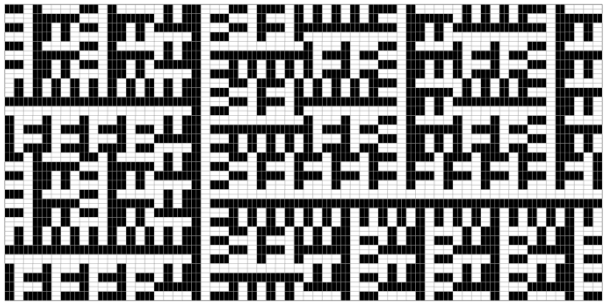
1200



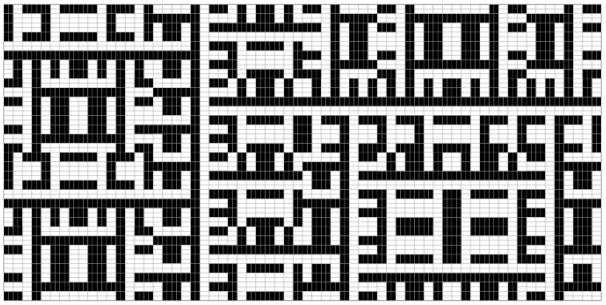
1201



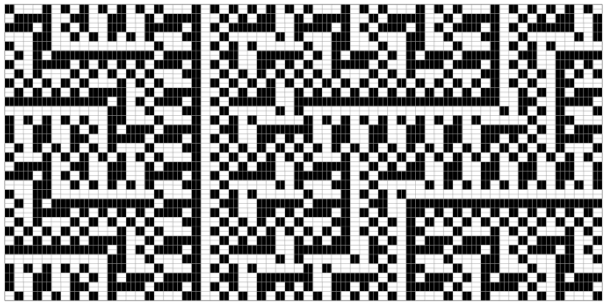
1202



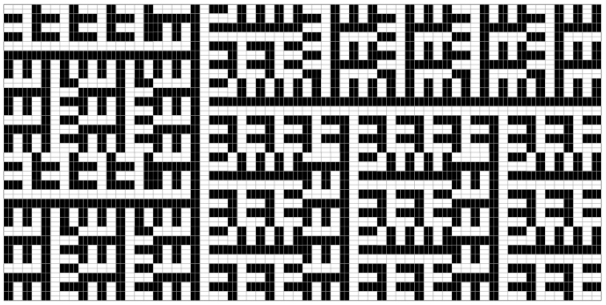
1203



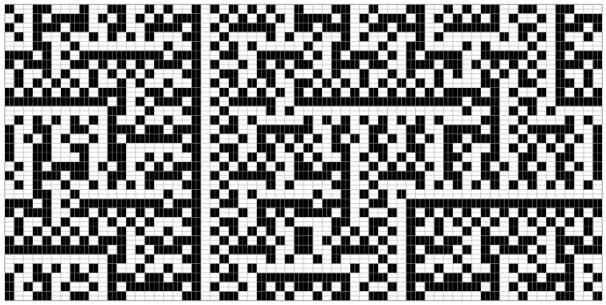
1210



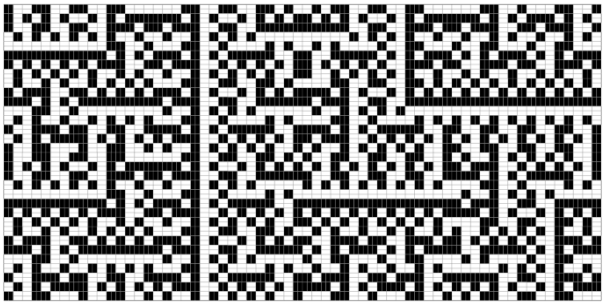
1211



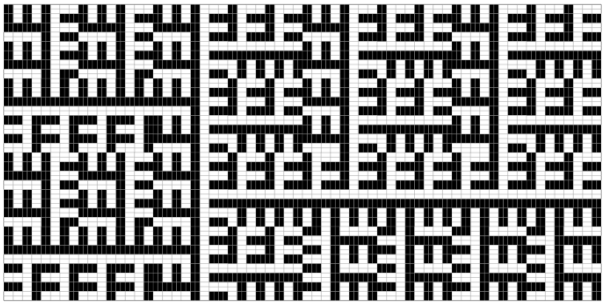
1212



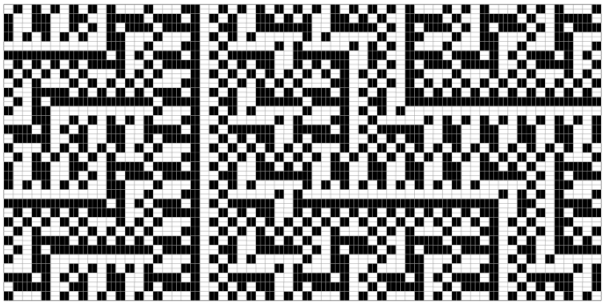
1213



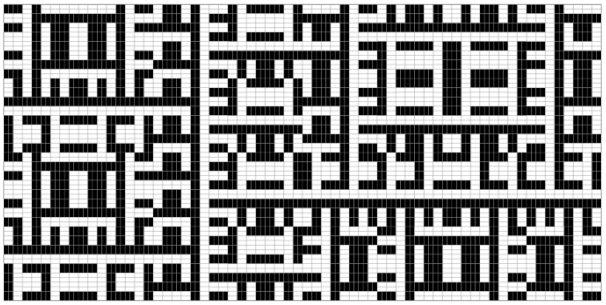
1220



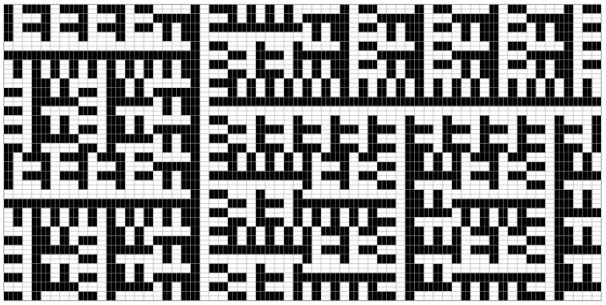
1221



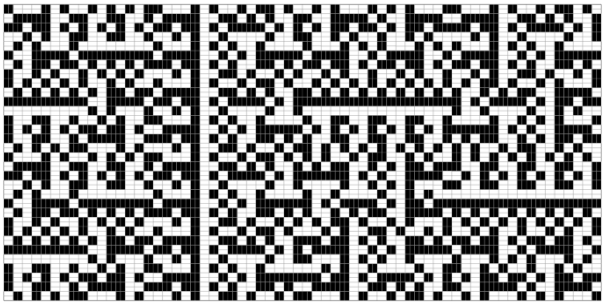
1222



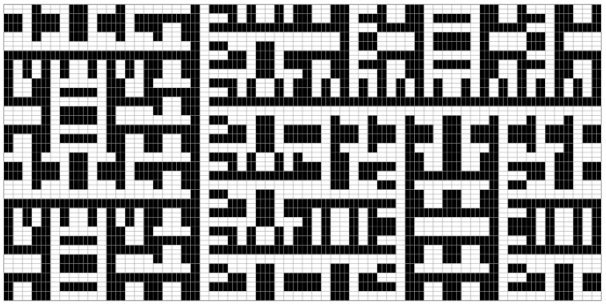
1223



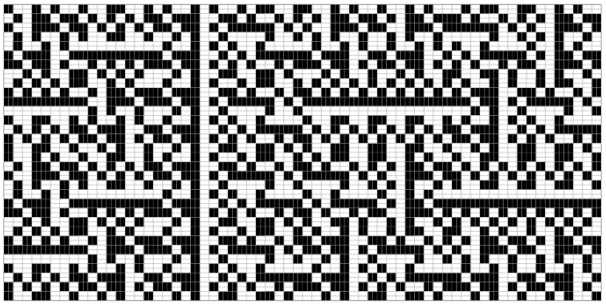
1230



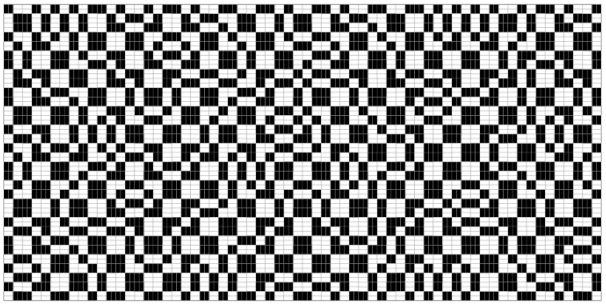
1231



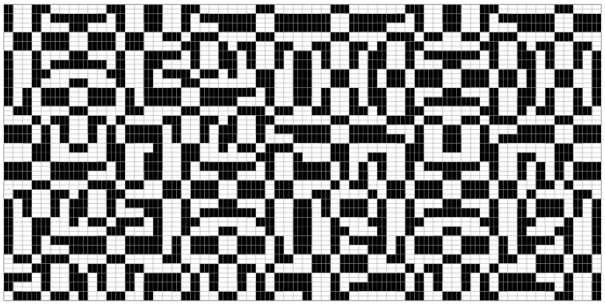
1232



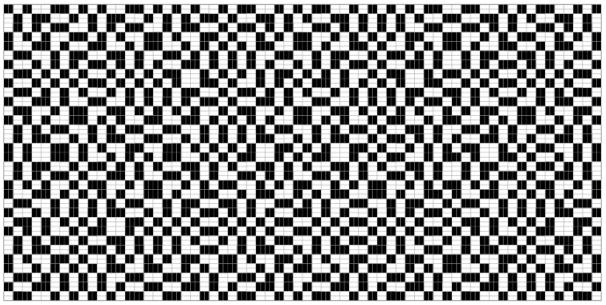
1233



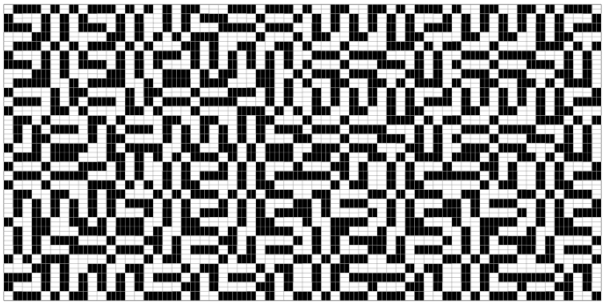
1300



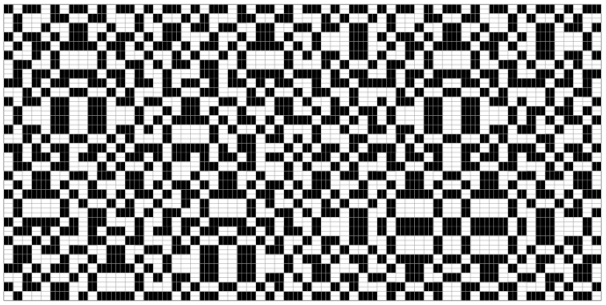
1301



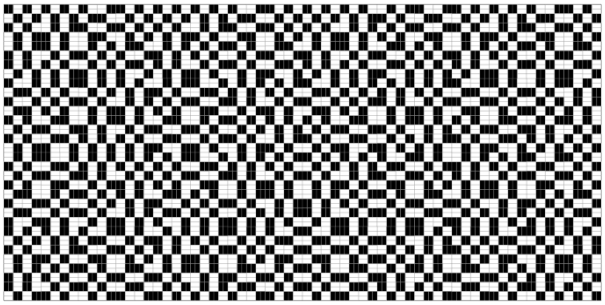
1302



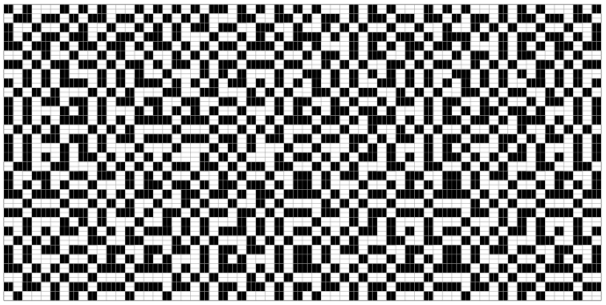
1303



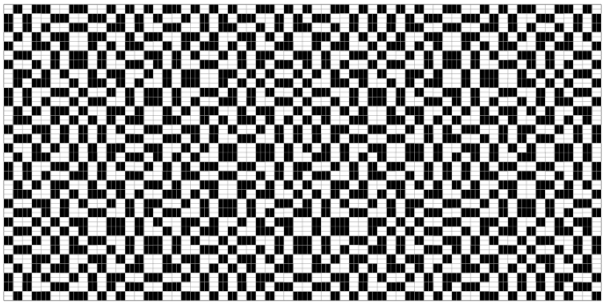
1310



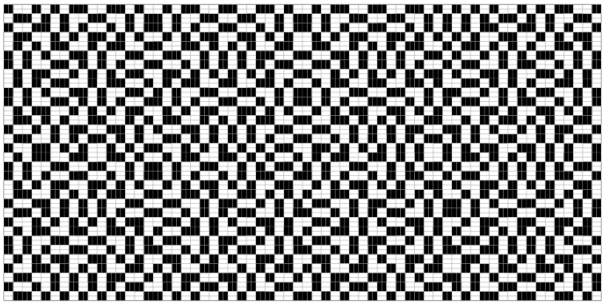
1311



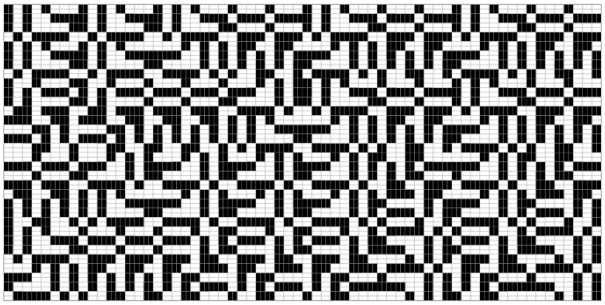
1312



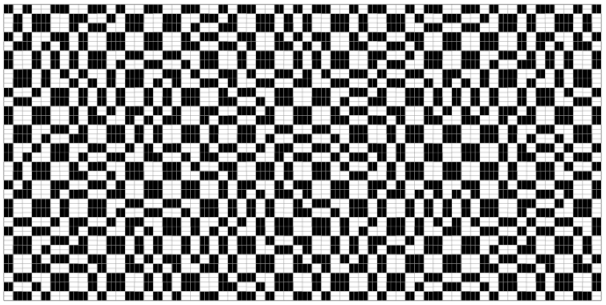
1313



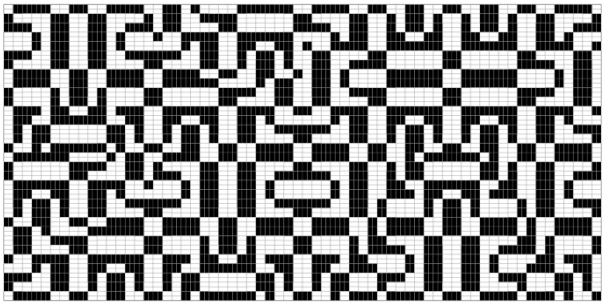
1320



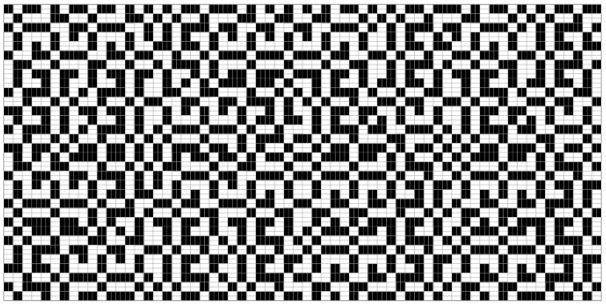
1321



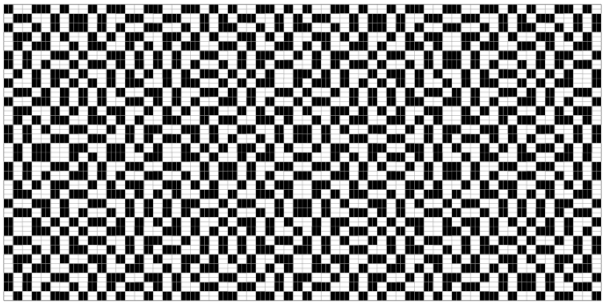
1322



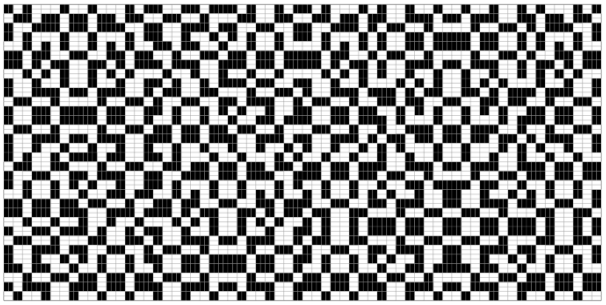
1323



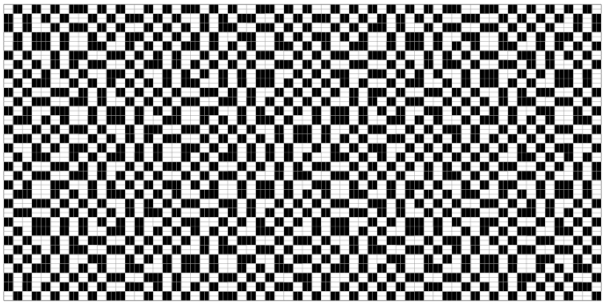
1330



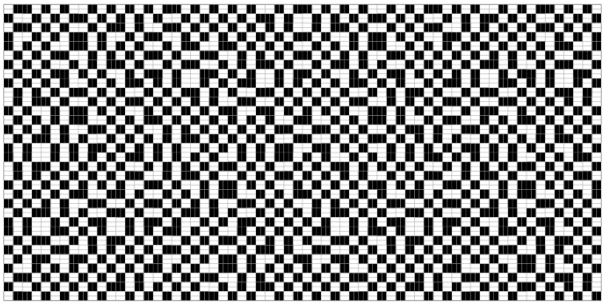
1331



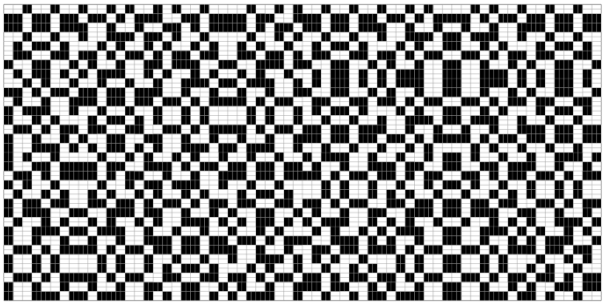
1332



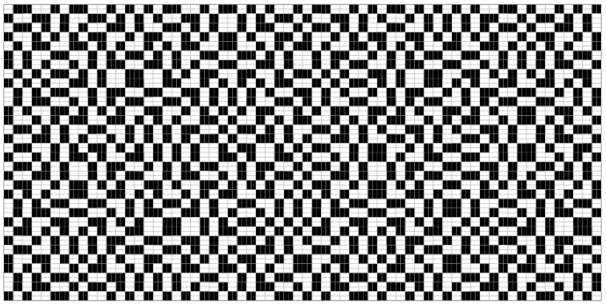
1333



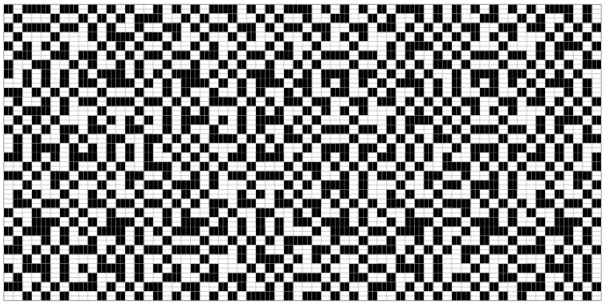
2000



2001

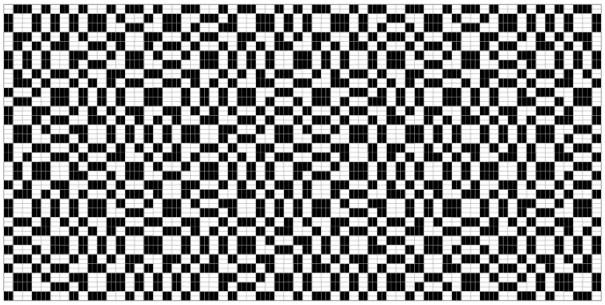


2002

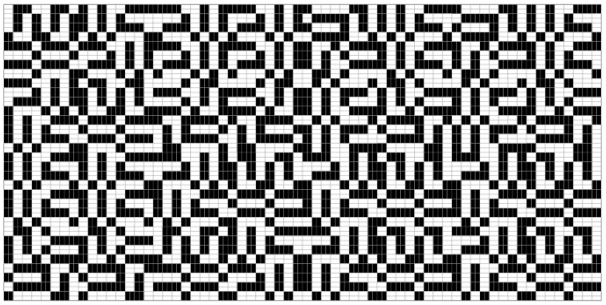


2003

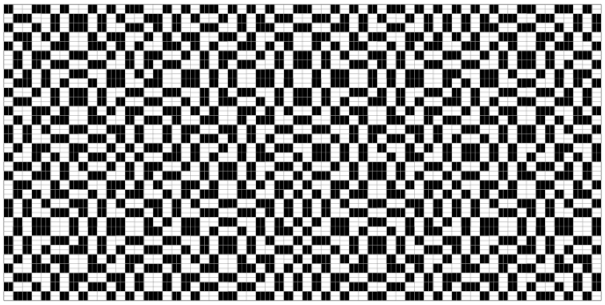
2010



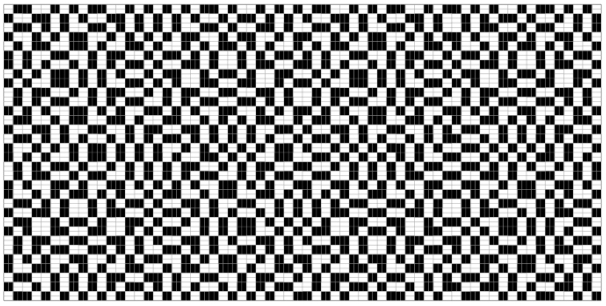
2011



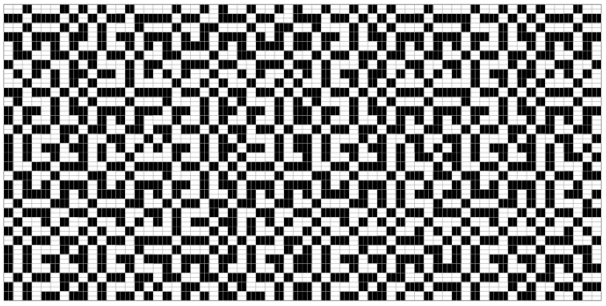
2012



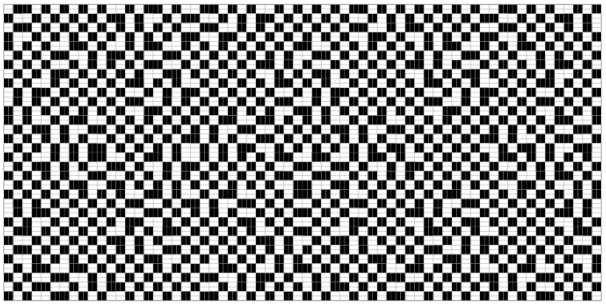
2013



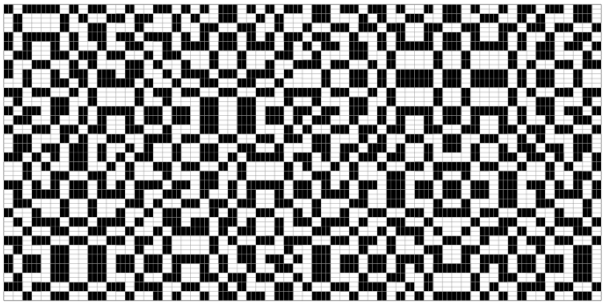
2020



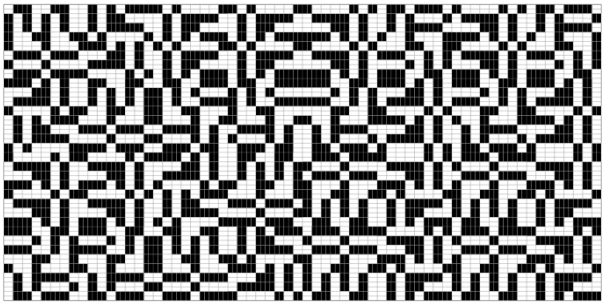
2021



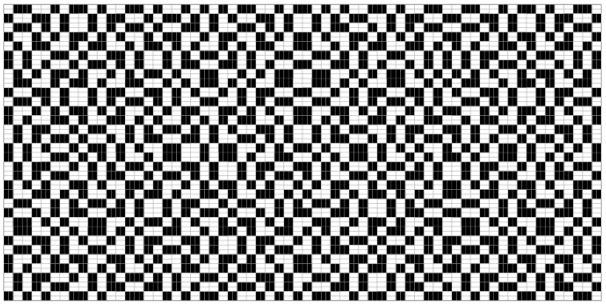
2022



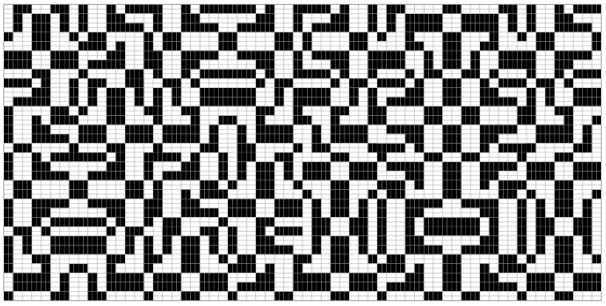
2023



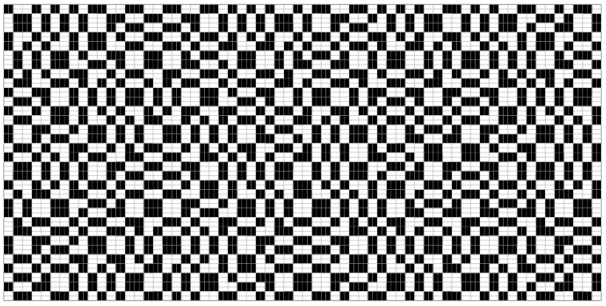
2030



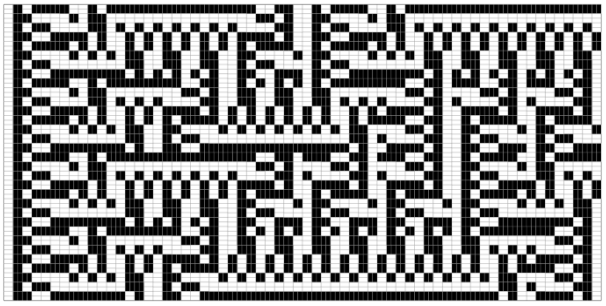
2031



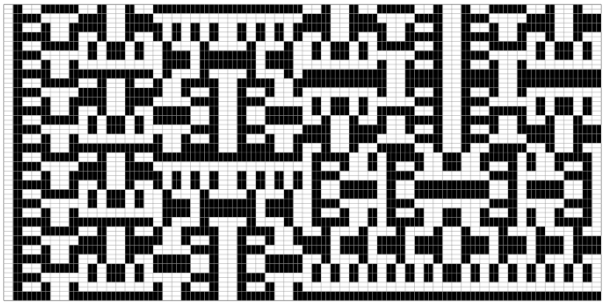
2032



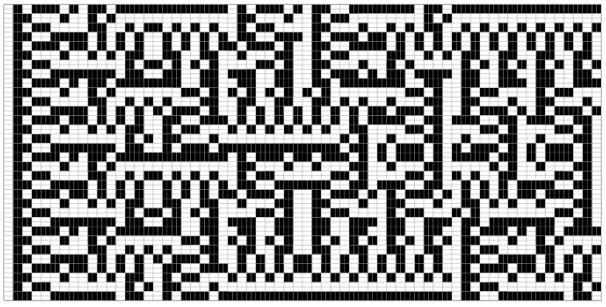
2033



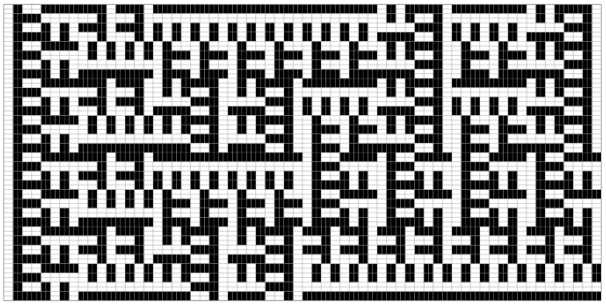
2100



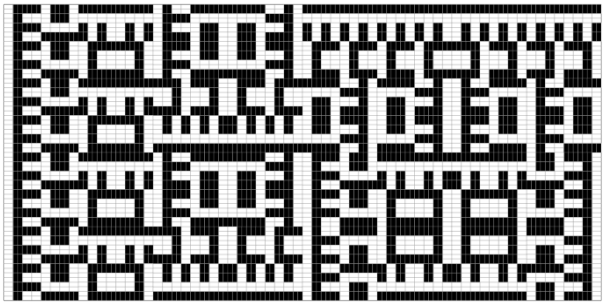
2101



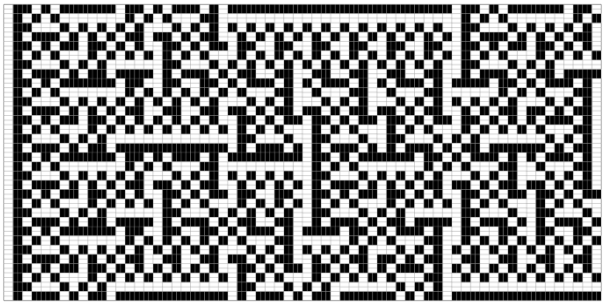
2102



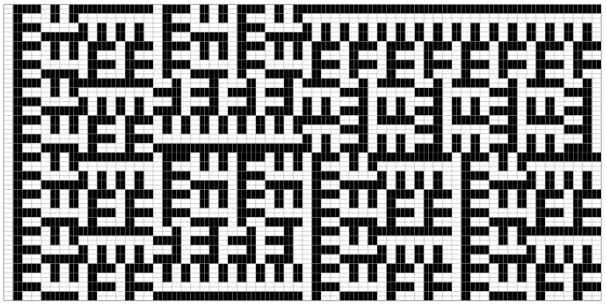
2103



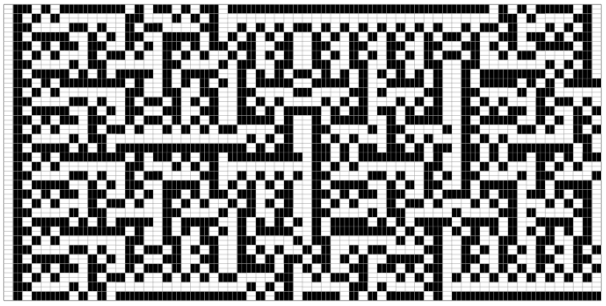
2110



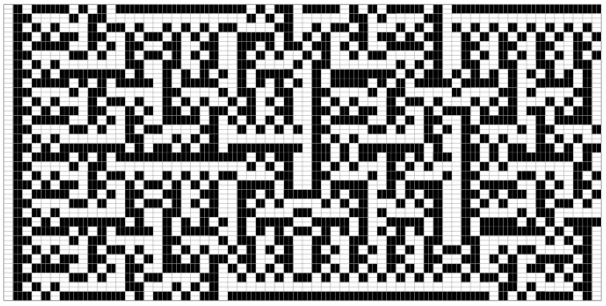
2111



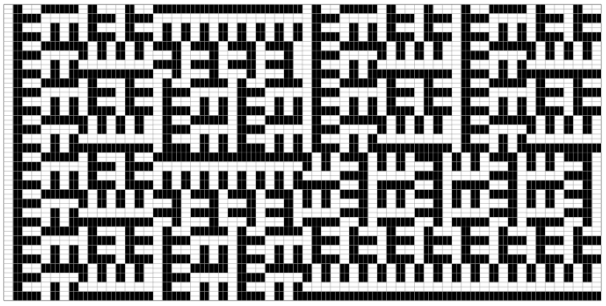
2112



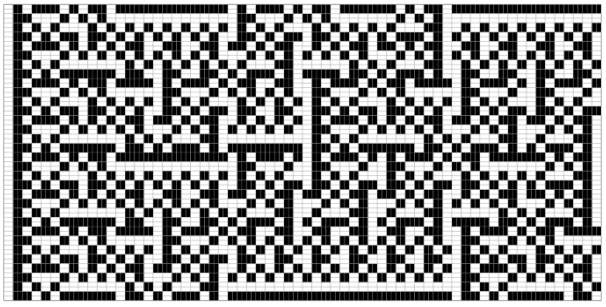
2113



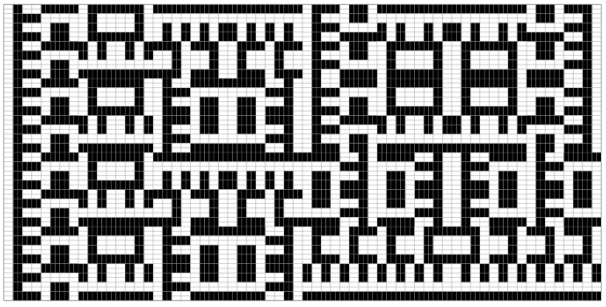
2120



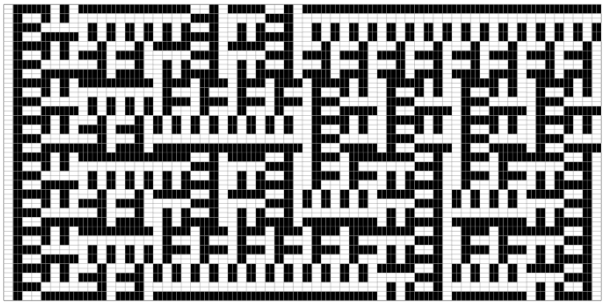
2121



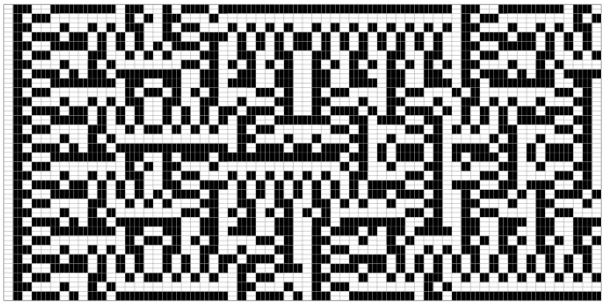
2122



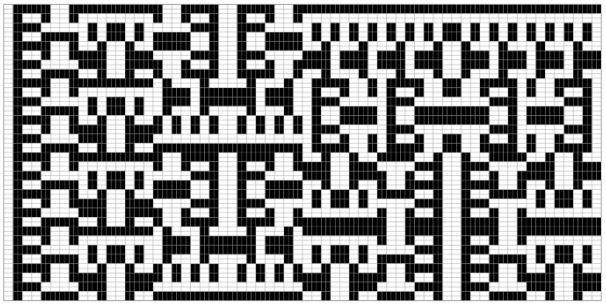
2123



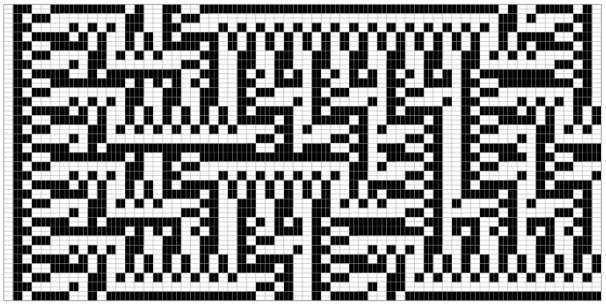
2130



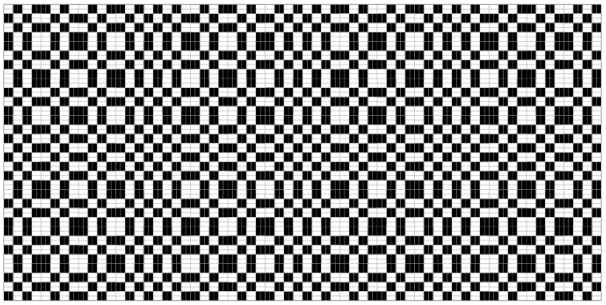
2131



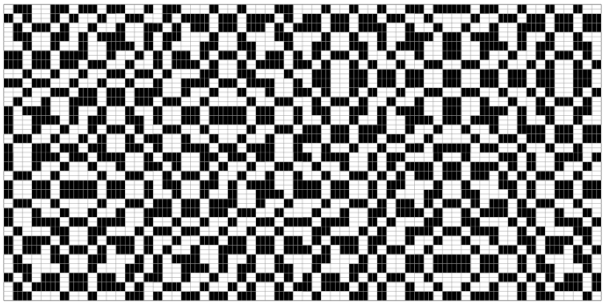
2132



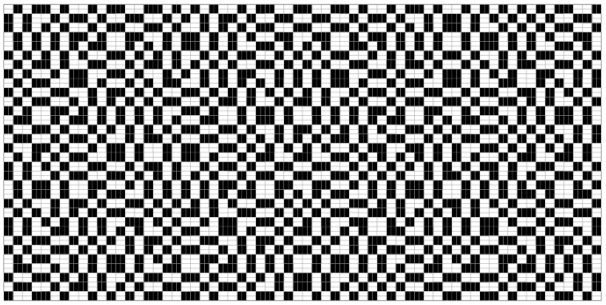
2133



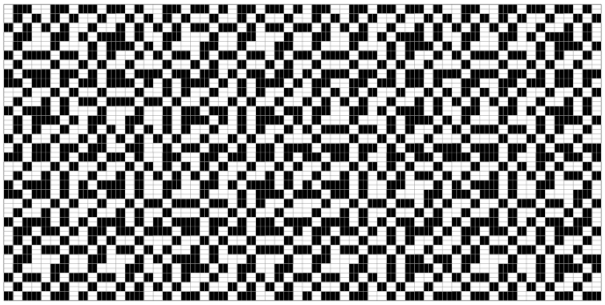
2200



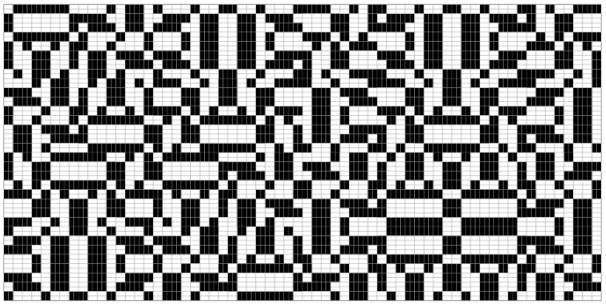
2201



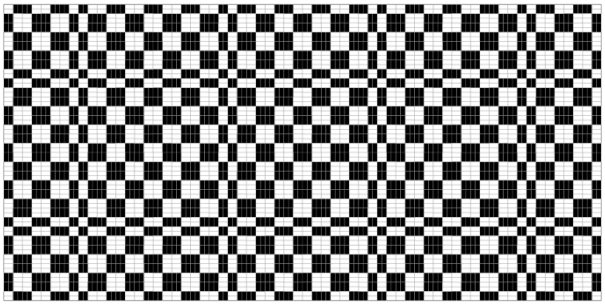
2202



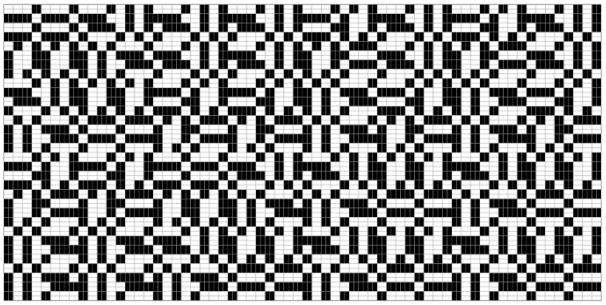
2203



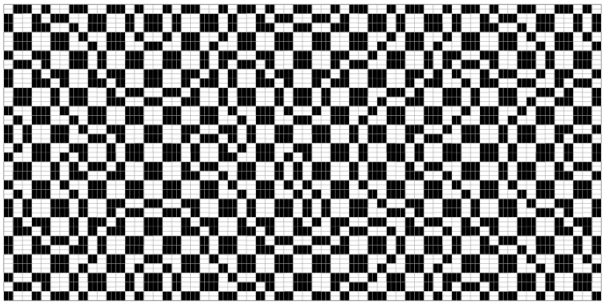
2210



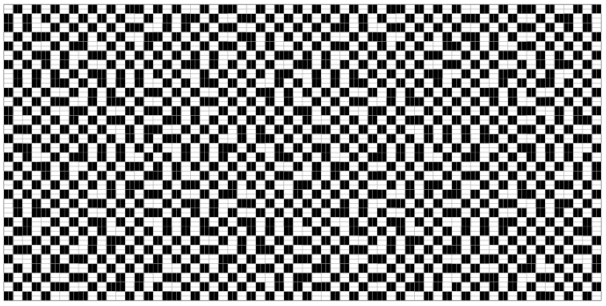
2211



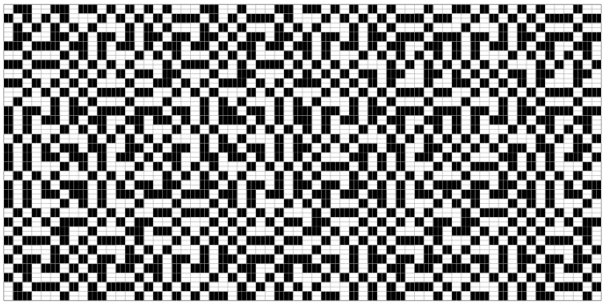
2212



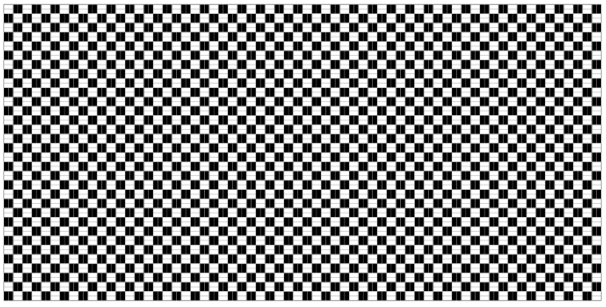
2213



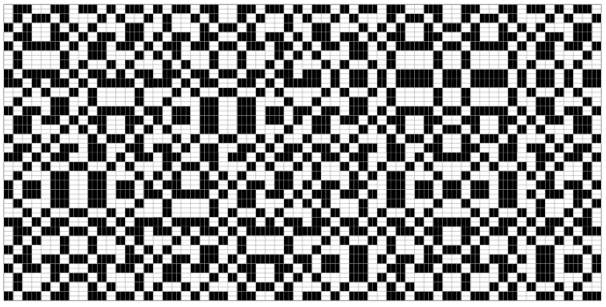
2220



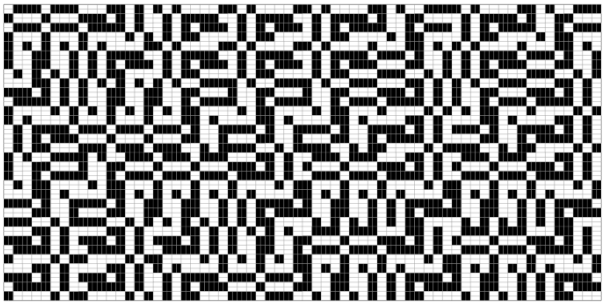
2221



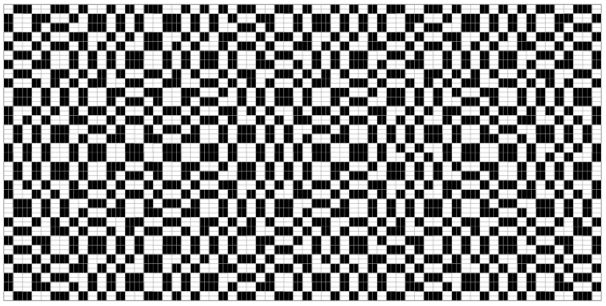
2222



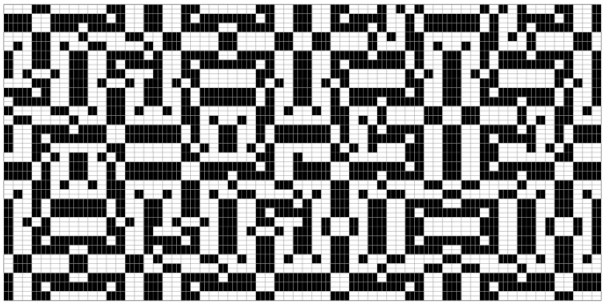
2223



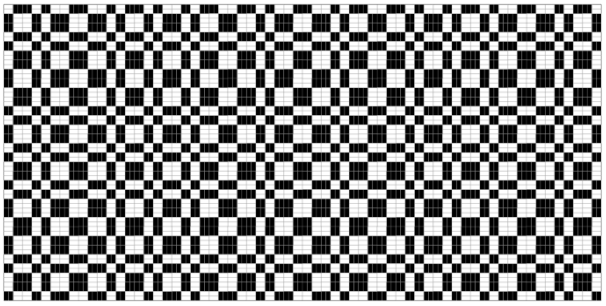
2230



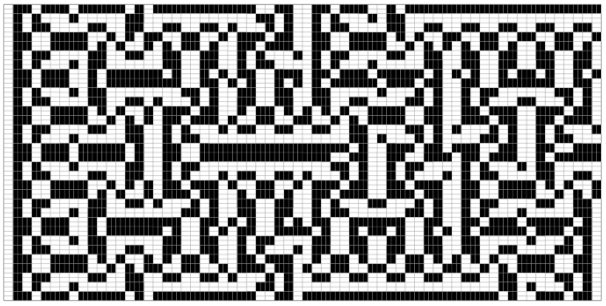
2231



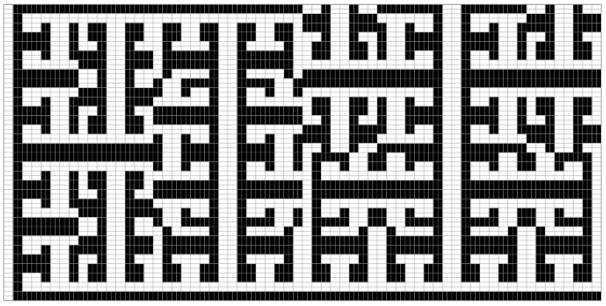
2232



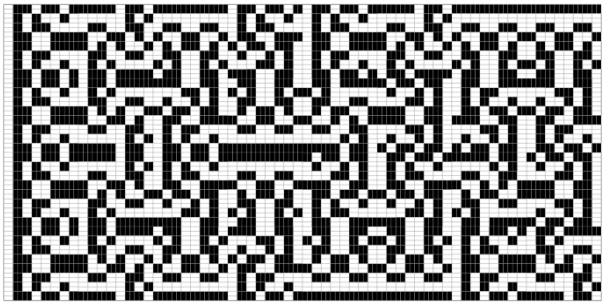
2233



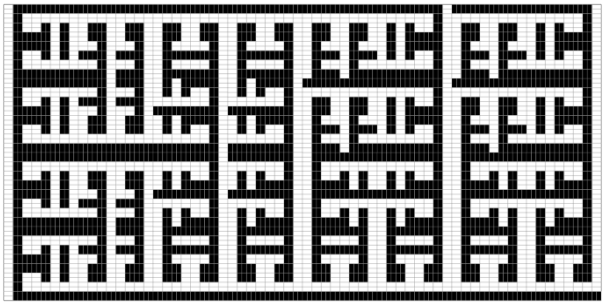
2300



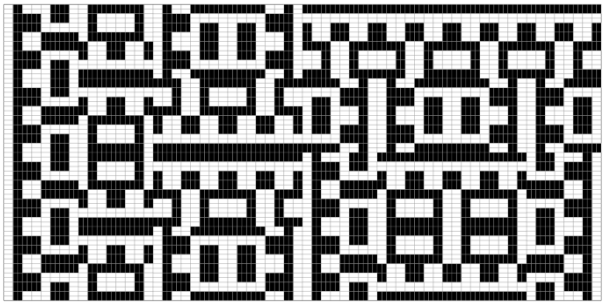
2301



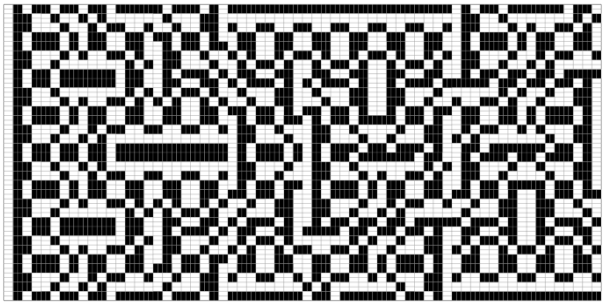
2302



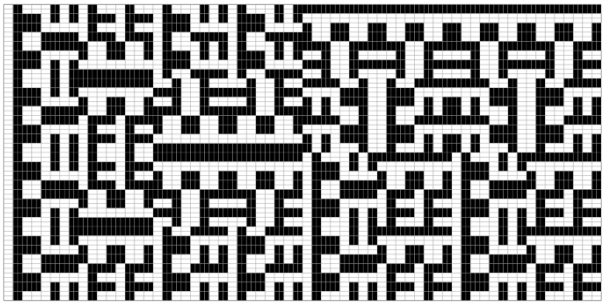
2303



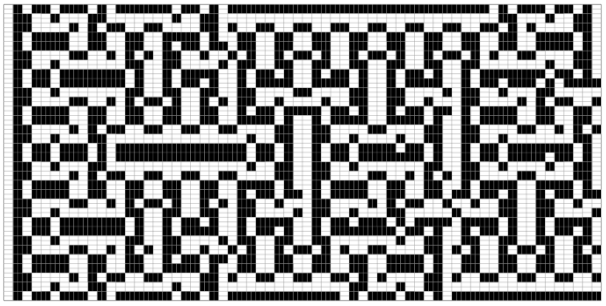
2310



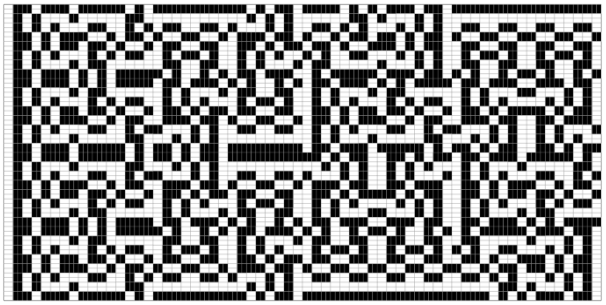
2311



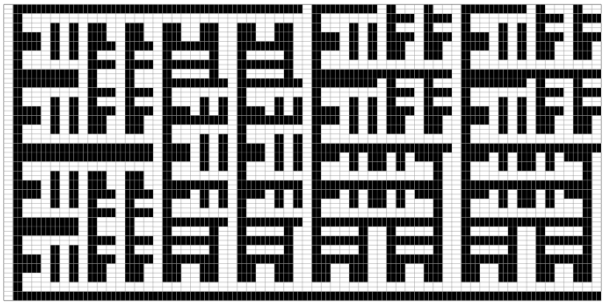
2312



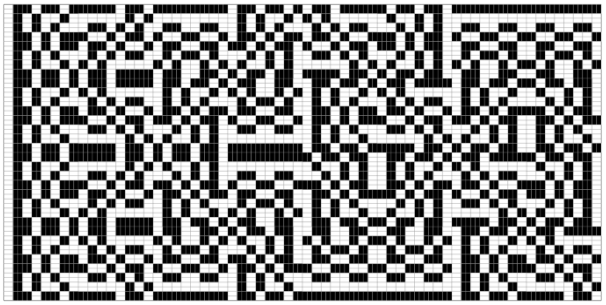
2313



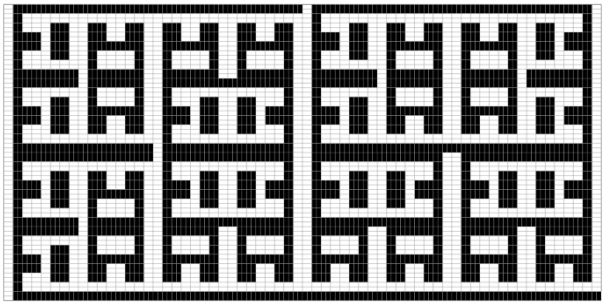
2320



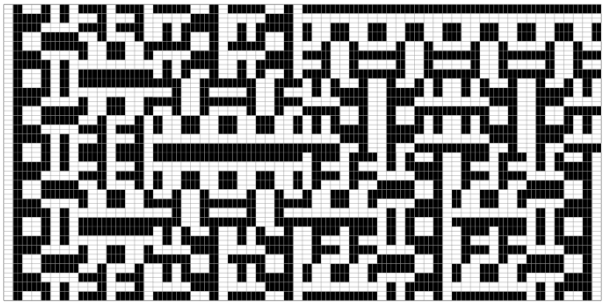
2321



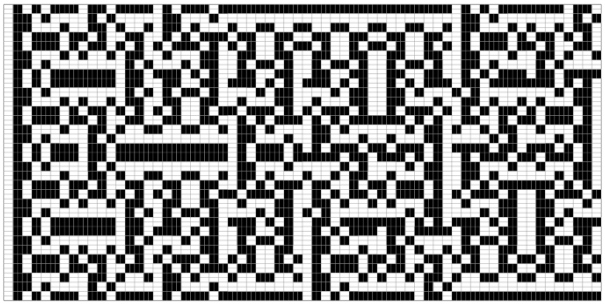
2322



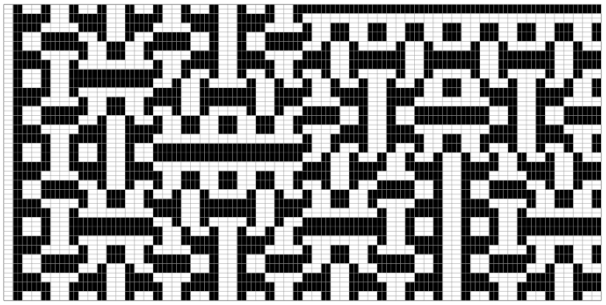
2323



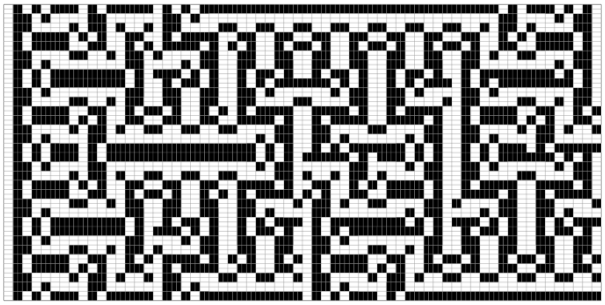
2330



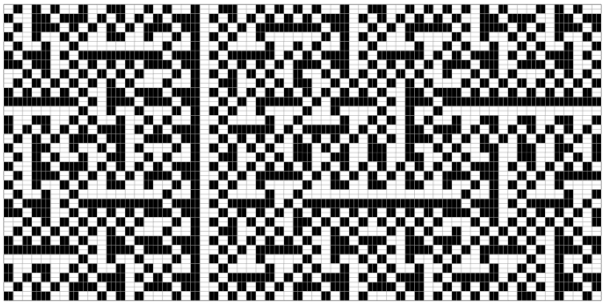
2331



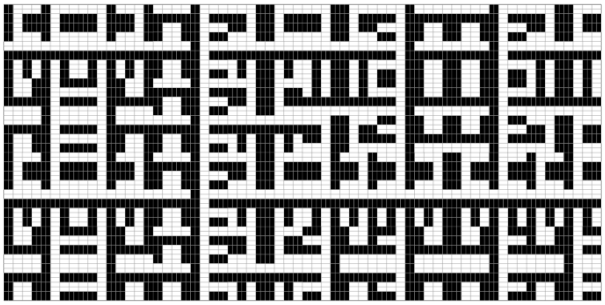
2332



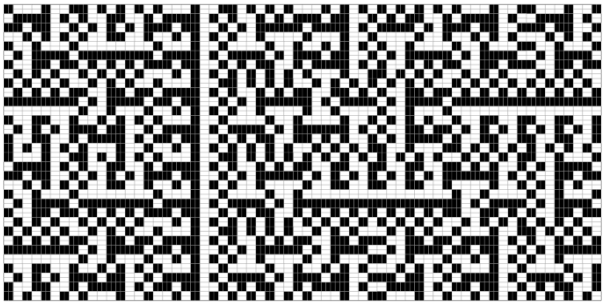
2333



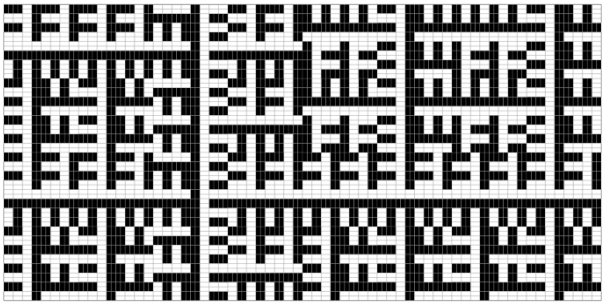
3000



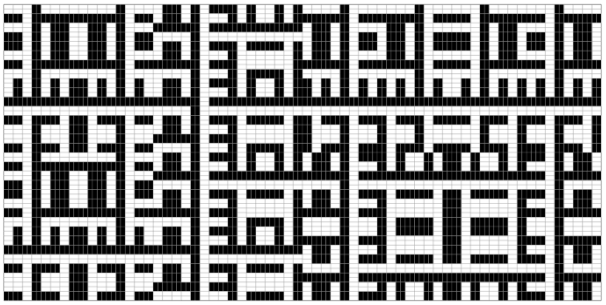
3001



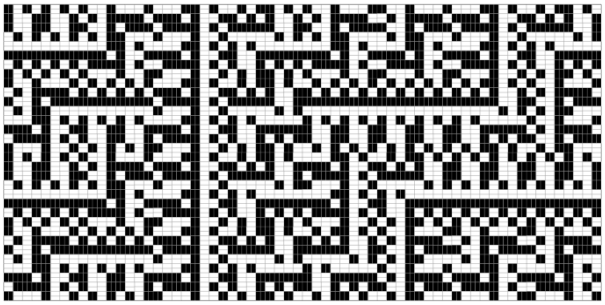
3002



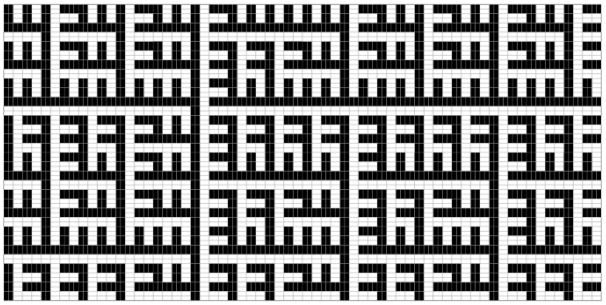
3003



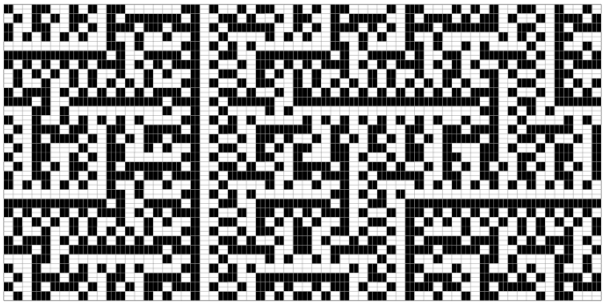
3010



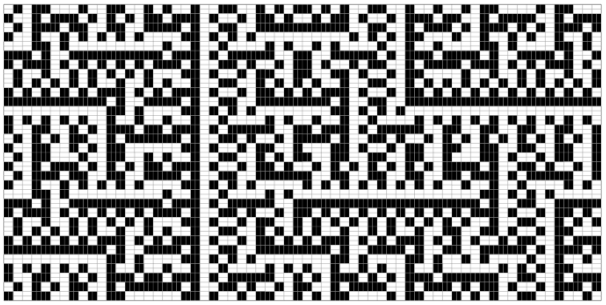
3011



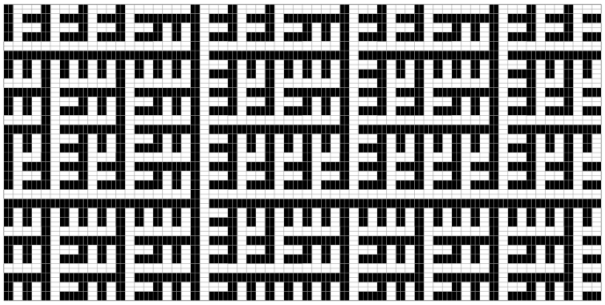
3012



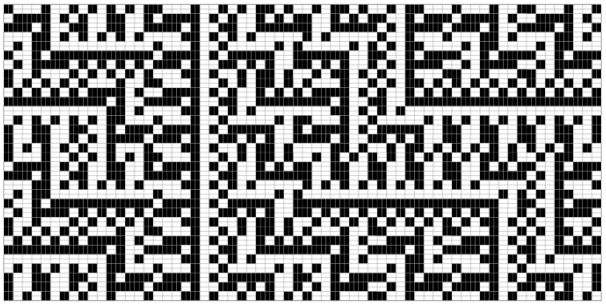
3013



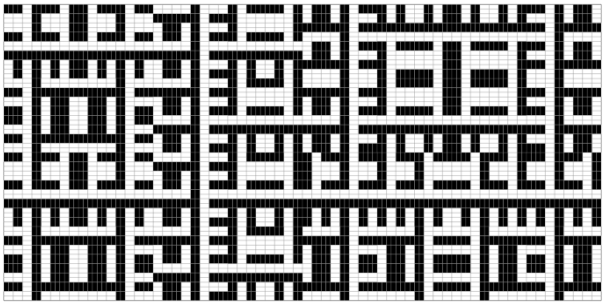
3020



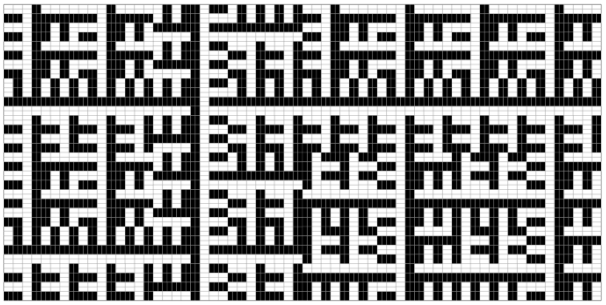
3021



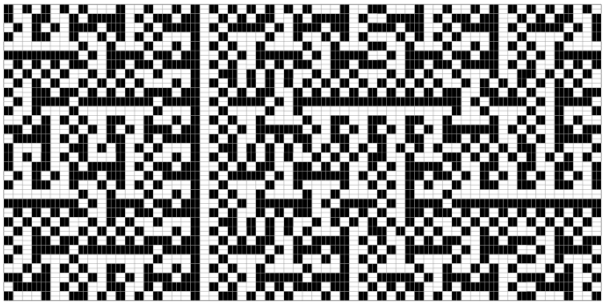
3022



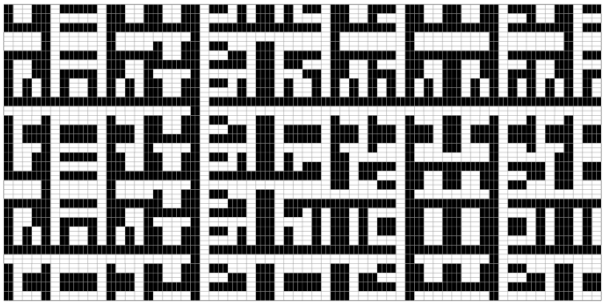
3023



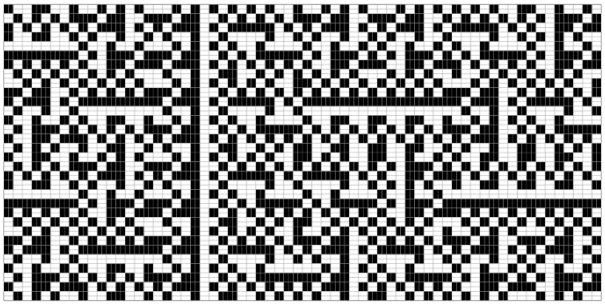
3030



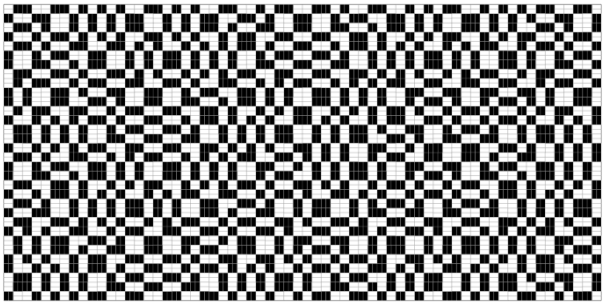
3031



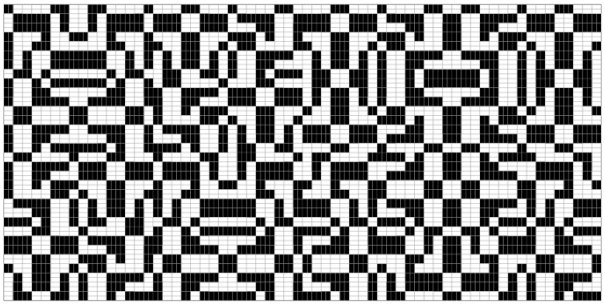
3032



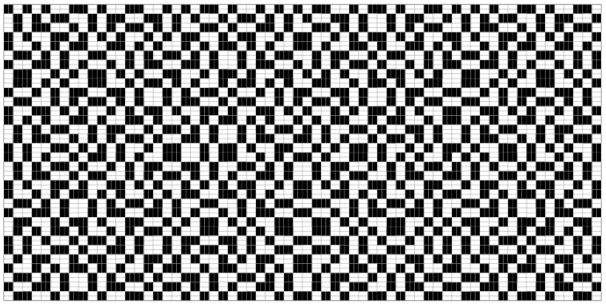
3033



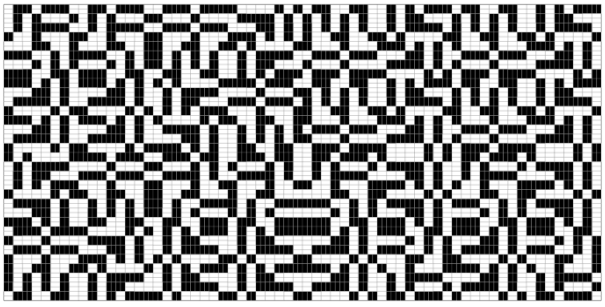
3100



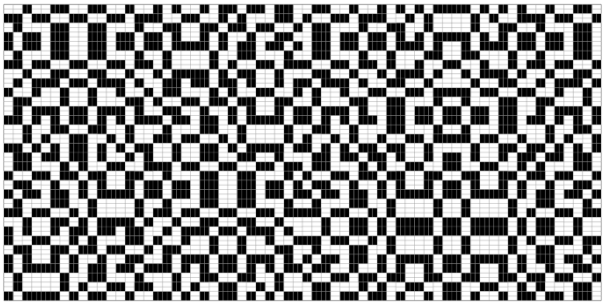
3101



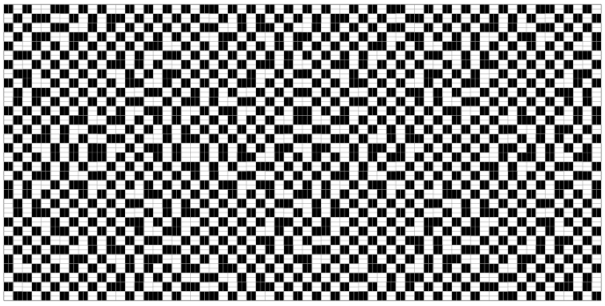
3102



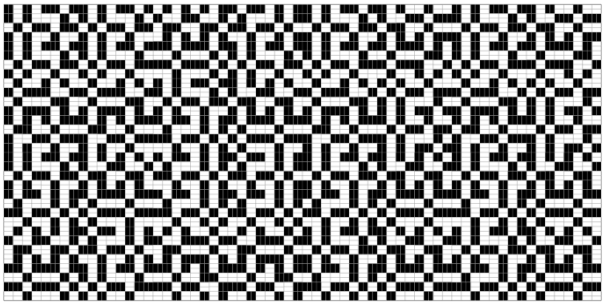
3103



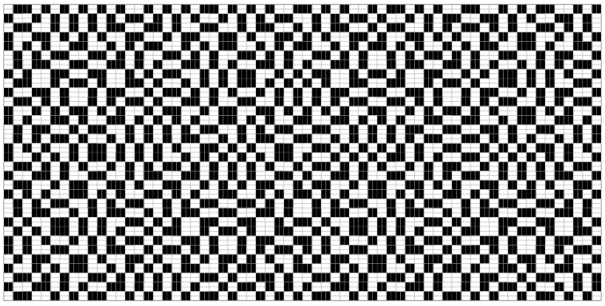
3110



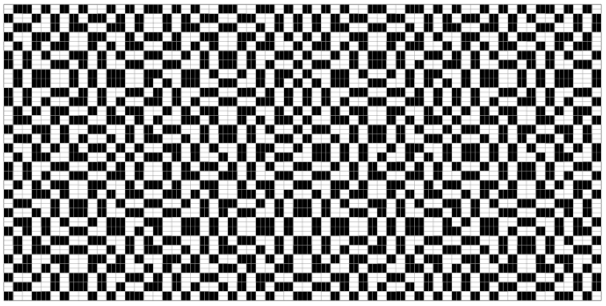
3111



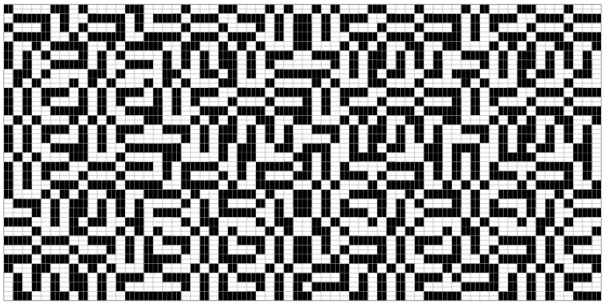
3112



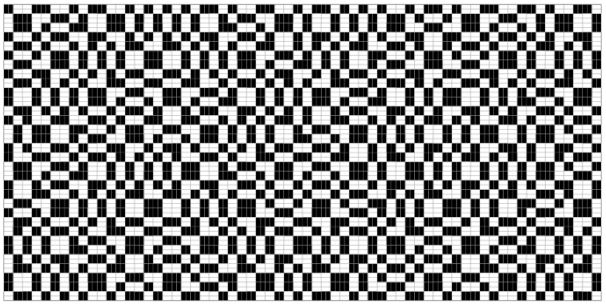
3113



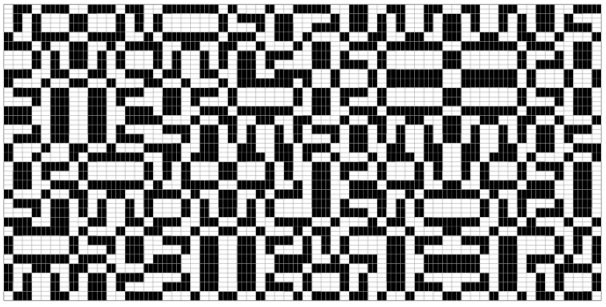
3120



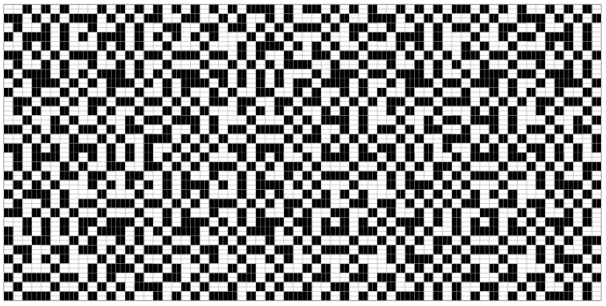
3121



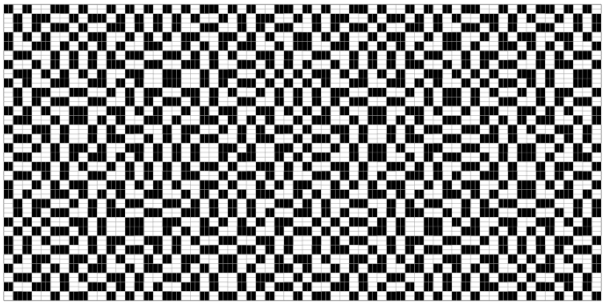
3122



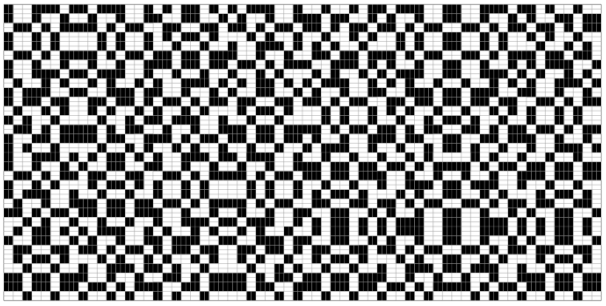
3123



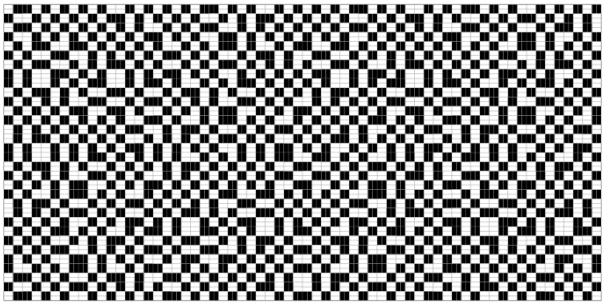
3130



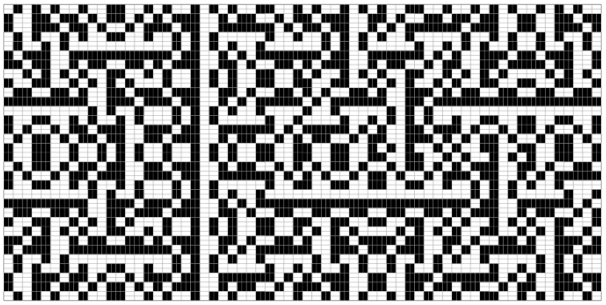
3131



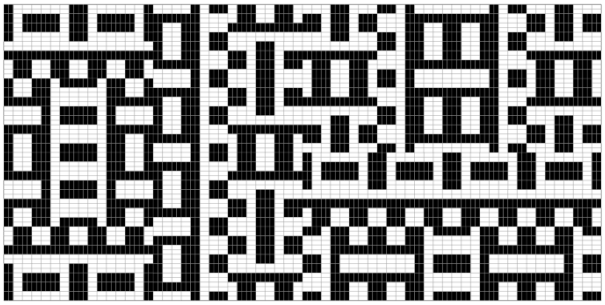
3132



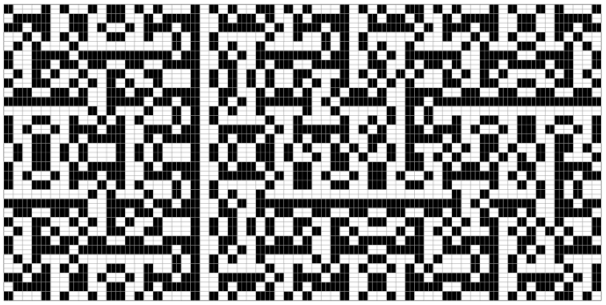
3133



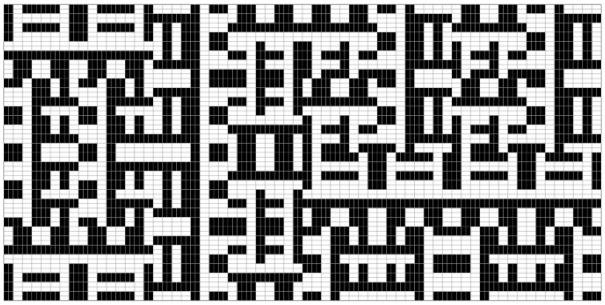
3200



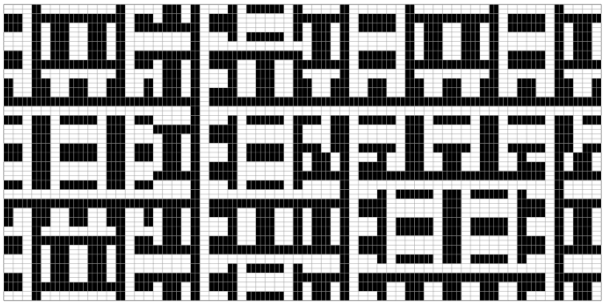
3201



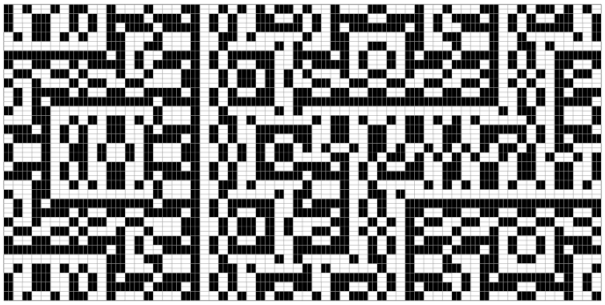
3202



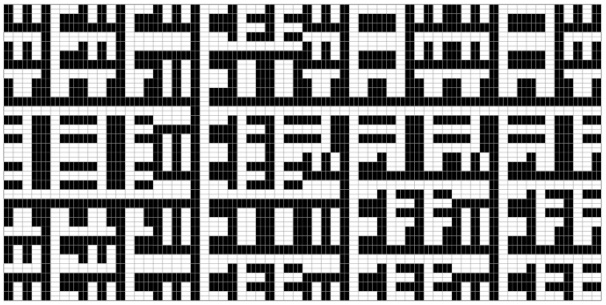
3203



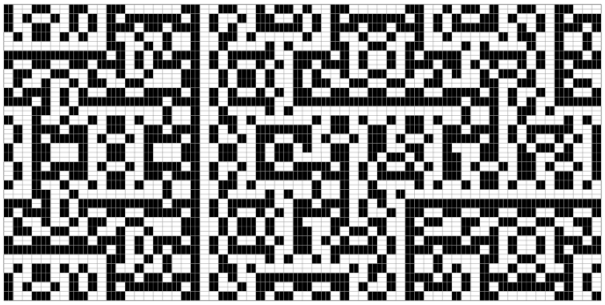
3210



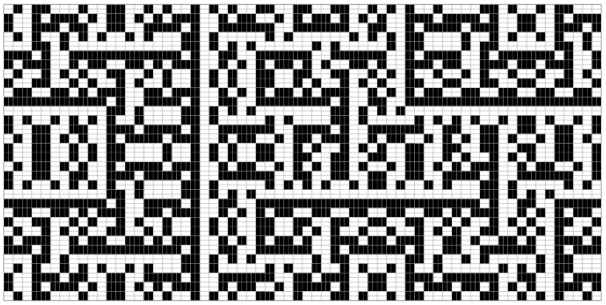
3211



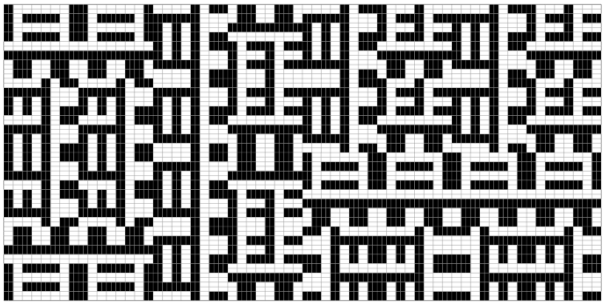
3212



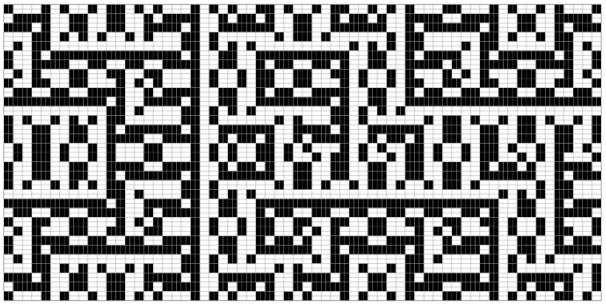
3213



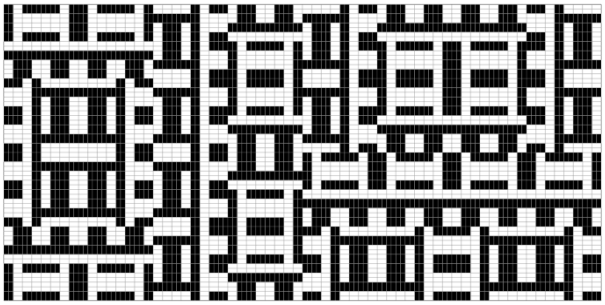
3220



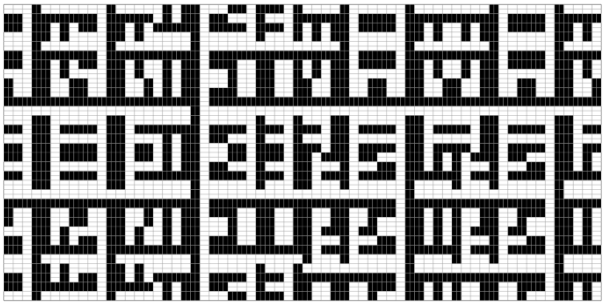
3221



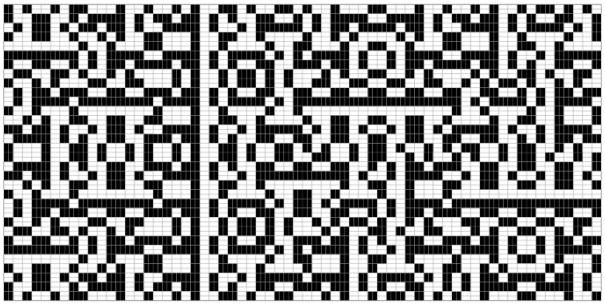
3222



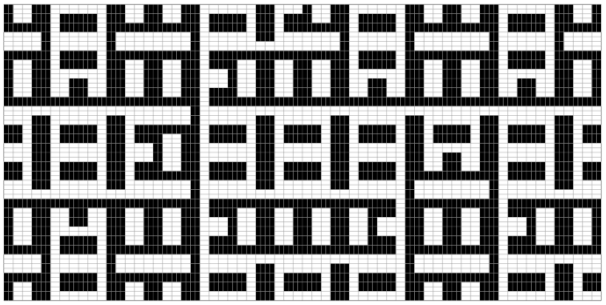
3223



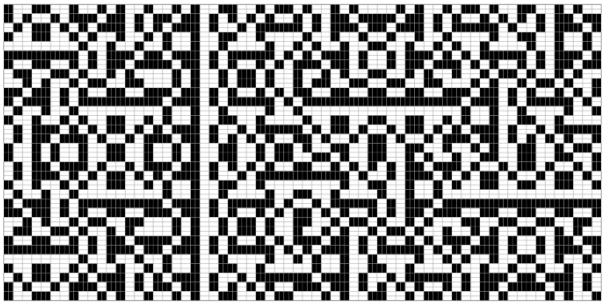
3230



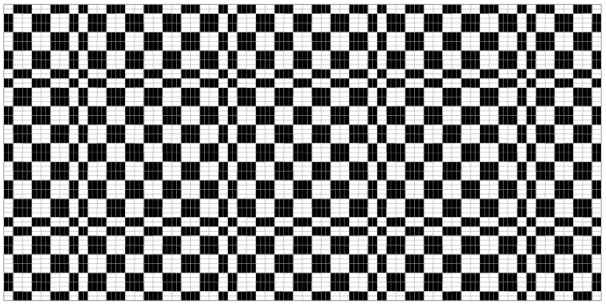
3231



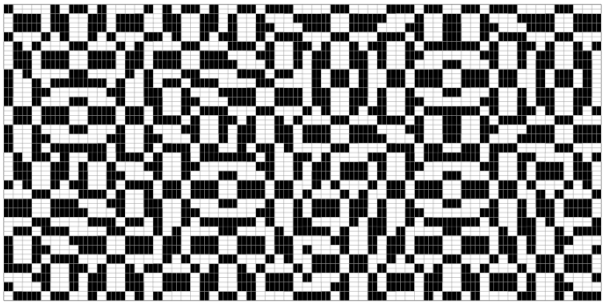
3232



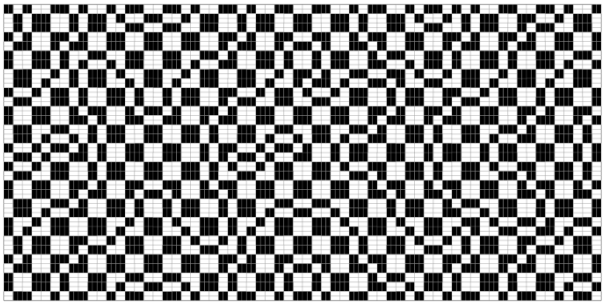
3233



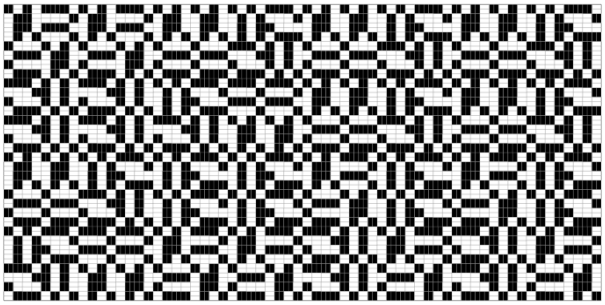
3300



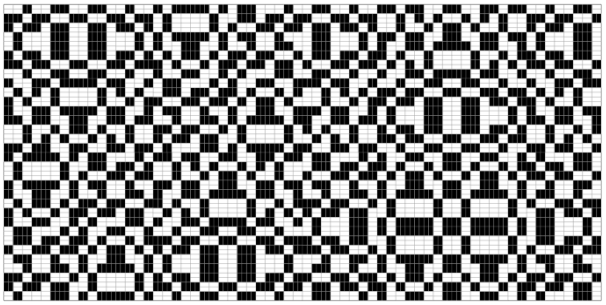
3301



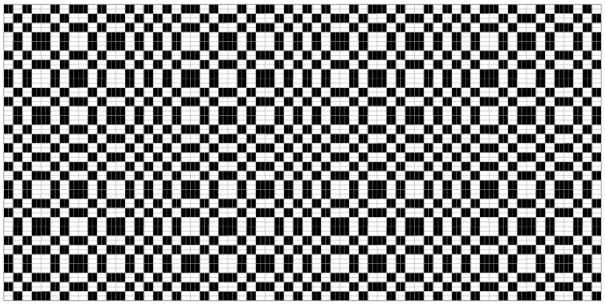
3302



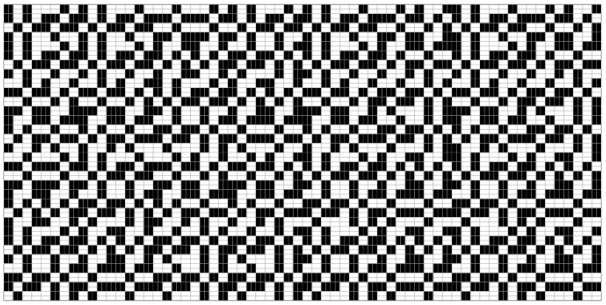
3303



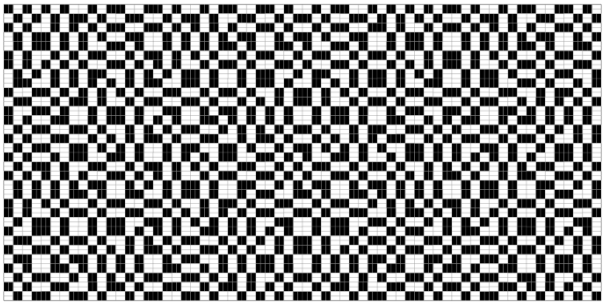
3310



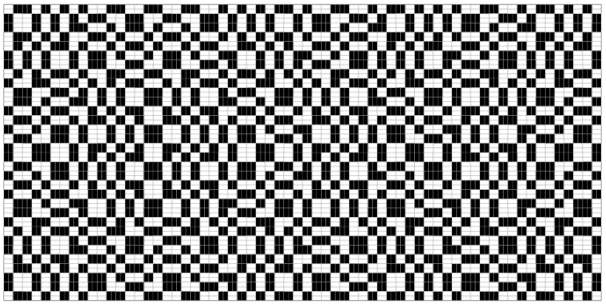
3311



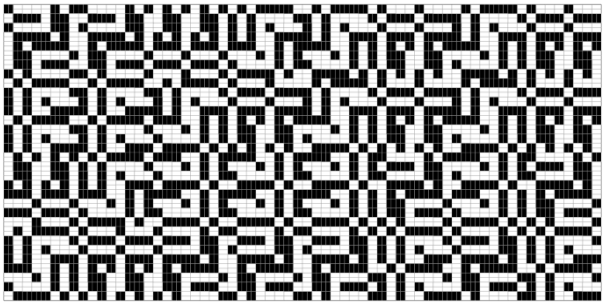
3312



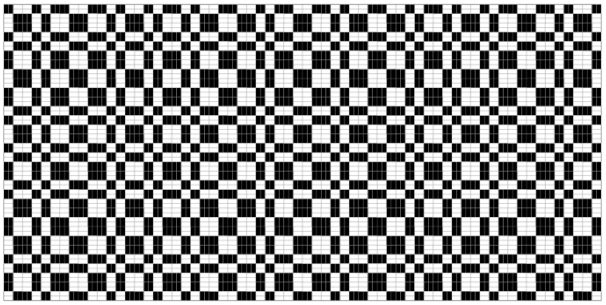
3313



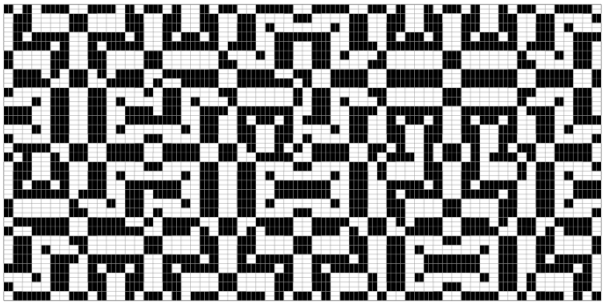
3320



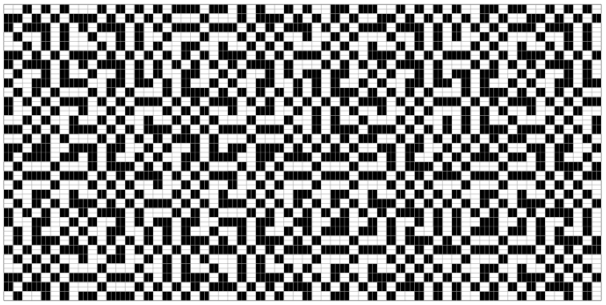
3321



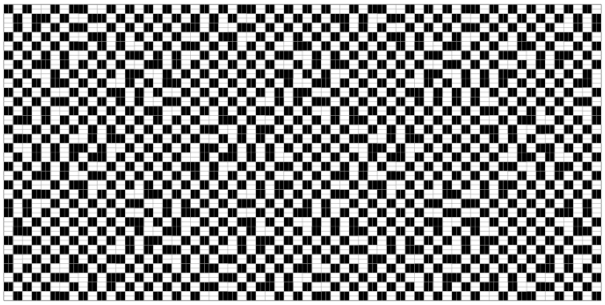
3322



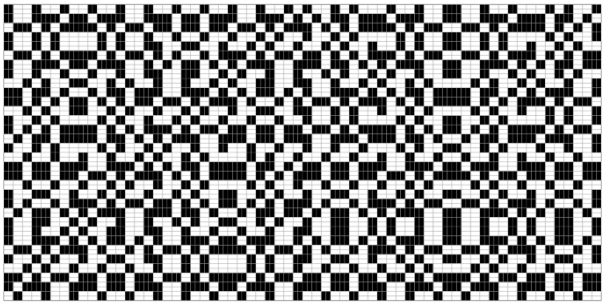
3323



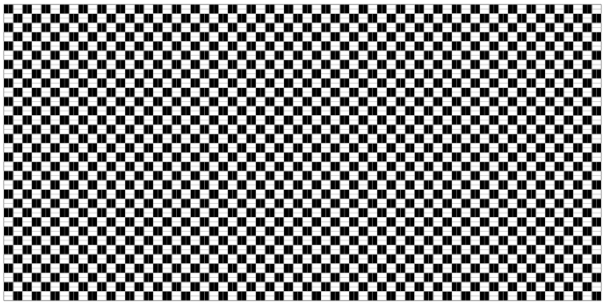
3330



3331



3332



3333