***ostinato and interrupt***
for guitar, unpitched noises, and sustained pitched tones

michael winter (mexico city, mx; 2017)
*para fatima*

## general remarks

The piece consists of a relatively slow, floating ostinato comprised of the following three note cells within which there is a descending bassline (given by the bottom note in each column):

| | | | | | |
|---|---|---|---|---|---|
| f♯ −14¢ | f♯ −14¢ | f♯ −14¢ | d | d | d |
| d | d | d | a | a | a |
| c | b | a♯ | f | e | d |

*Note that the f♯ is flat 14 cents, or hundredths of a tempered semitone, from the tempered f♯ in twelve-tone equal temperament as it is a 5/4 major third above the d.

The ostinato is occasionally interrupted by a strictly metered, more rapid sequence of sounds derived from and accompanying open and muted strings as well as natural harmonics played on the guitar.

The interruptions are at a tempo of precisely 75 beats per minute with four 16th notes to the beat (such that the 16th note is always 0.2 seconds). The piece is designed such that the interruptions always start at a second marking to enable coordination and the ability to set the correct tempo by, for example, counting 60 beats per minute (one beat per second) with five 16ths to the beat and then making the metric modulation to 75 beats per minute by simply changing the beat from five 16ths to four 16ths; i.e. changing the tempo while keeping the atomic unit of a 16th note the same at 0.2 seconds. A number below a stem indicates a change in the duration of the following notes: 1 equals 0.2 seconds, 2 equals 0.4 seconds (or two times the atomic duration), or 3 equals 0.6 seconds (or three times the atomic duration).

## guitar

The piece is essentially a guitar piece that can be accompanied by computer (with custom software written in SuperCollider) and / or live performers. In the following sections, each accompanying element is detailed with an explanation of what is occurring in the computer program and how it can be substituted or complemented by live performers.

The notation is given in tablature-form where each line represents one of the strings of the guitar from high to low. A filled-notehead indicates an open string. A number indicates a stopped string at the fret of the given number. A diamond-notehead indicates a natural harmonic (chosen arbitrarily / randomly). And an x-notehead indicates a muted string.

The guitar is tuned in an open d tuning as follows: VI) e down to d, V) a, IV) d, III) g down to f♯ −14¢, II) b up to c −31¢, and I) e down to d. Strings IV and III can be tuned from the 5th and 7th harmonic of string VI, respectively.

The ostinato sections should be interpreted freely. The guitar enters 12 seconds into the piece. The given notes, written proportionally with 12 seconds per system, should serve more as a general guideline. The guitarist is free to embellish ad lib such that initially, embellishments are rather infrequent resulting in a relatively sparse texture, and as time progresses, the general density of the ostinato increases by the aid of more and more embellishments. The embellishments should remain within the harmonic world of the ostinato.

The computer program allows the guitarist to use sampled recordings of a guitar which will accurately produce what is written. The guitarist may choose to play with a sample-based realization and / or with other guitarists; e.g. by dividing the ostinato and interrupt sections among multiple performers and / or the computer realization. Note that the electronic accompaniment makes decisions at every point indicated by a notehead in the ostinato sections (explained in detail below).

The interruption sections are clearly delimited from the ostinato by double bar lines and because they are strictly metered. For both the ostinato and interrupt sections, tones should be allowed to decay as long as possible.

## unpitched noises (noise fields and percussion)

The ostinato and interrupt sections are further distinguished by noise fields. The computer program simply oscillates between brown noise for the ostinato sections and white noise for the interruptions. These sounds can be replaced and / or complemented by live performers choosing two distinct noises for the ostinato and interrupt sections, respectively. The level of the noise fields should be relatively unobtrusive and situated at a level well below the guitar. The noise fields fade out over 10 seconds after the final note of the guitar.

In the interrupt sections, unpitched short percussion sounds double the muted strings of the guitar (indicated by x-noteheads) using a distinct instrument for every line of the staff. While a live performer is highly preferred, the computer program allows this part to be realized using sampled recordings. The score includes a percussion part with all non x-noteheads grayed out.

**sustained tones (harmonic flickering, sine beating, and interrupt highlights)**

The bassline of the ostinato can be optionally highlighted by what can be described as a 'harmonic flickering' where harmonics of the bassline flit in and out with various degrees of pitch definition. This is produced in the computer program by a simple waveshaping technique where a buffer is intermittently filled with bursts of noise and played back such that position of the player resets to the beginning of the buffer at a rate equivalent to the frequency of the current bass note in the ostinato. These sounds can be replaced and / or complemented by live performers playing swelled tones and occasionally mimicking the synthesis process on harmonics (including the fundamental) of the current bass note. Note that at each ostinato note, the computer can either turn on or off the flickering. The flickering nature of the sound itself is created by randomly gating the buffer player at very short intervals and the random intervals in which the buffer is refilled. If the flickering is realized by live performers, a similar decision making process should be coordinated. In the notation, brackets around the notes indicate when the bassline progresses. The harmonic flickering should be relatively unobtrusive and situated at a level well below the guitar.

The ostinato and interrupt sections are also distinguished by the beating of two low tones between 0.5 and 3 hertz apart centered around d for the ostinato sections and a fourth below, a, for the interrupt sections. This is achieved using sine tones in the computer program, but may be replaced by live performers playing low instruments. Note that the sine tones, and thus preferably any instruments that replace the sine tones, are actually centered an octave below the VI string of the guitar in the ostinato sections and 2 octaves below the V string of the guitar in the interrupt sections. In the ostinato sections, the computer determines whether or not to change the rate of beating on the onset of every note. This can easily be reproduced by two players such that one remains constant while the other changes occasionally. The rate of beating remains constant throughout each interrupt section. The beating should be at a level such that it provides a clear and present foundation while not overwhelming any of the other sounds.

In the interrupt sections, every open string and natural harmonic played on guitar is accompanied / highlighted by sustained tones that are octave equivalents of the fundamental or harmonics above the fundamental, respectively. For the latter, any harmonic greater than 2 can be chosen randomly / arbitrarily. The computer uses sine tones with amplitudes equivalent to 1 divided by the harmonic number. It is encouraged that live performers complement instead of replace the computer generated tones such that each player interprets one line of the score realizing filled-noteheads as octave equivalents of the given open string of the guitar and daimond-noteheads as octave equivalents of natural harmonics of the given string. If possible, as with the synthesized tones, performers should try to play these tones at a level indirectly proportional to the harmonic number. If a filled-notehead is repeated, the tone may be rearticulated or played in a different octave. A different harmonic can be chosen every time a diamond-notehead occurs. An x-notehead indicates to stop the currently sounding tone. If a sound has not been stopped before a transition back into the ostinato section, the tone fades out over a few seconds slightly overlapping into the following ostinato section. The score includes 6 ensemble parts where all but the relevant stafflines and noteheads are grayed out. The same performers can realize both the harmonic flickering of the ostinato and the interrupt highlights. To facilitate performances with less than 6 performers realizing the highlights, the parts may be divided among performers and the computer program (which allows muting individual parts).

Note that where octave equivalents of harmonics are played (such as in the harmonic flickering and the interrupt highlights), several pitches deviate from the nearest pitch in twelve tone equal-temperament. Below, the first 6 unique pitch classes (based on primes) of the harmonic series on d are listed with a cents deviation from the nearest pitch in twelve-tone equal-temperament. These pitches can be transposed accordingly for all other cases.

| | | | | | |
|---|---|---|---|---|---|
| d | a +2¢ | f$^\sharp$ −14¢ | c −31¢ | g$^\sharp$ −49¢ | a$^\sharp$ +40¢ |
| 2 | 3 | 5 | 7 | 11 | 13 |

**SuperCollider program structure**

The structure of the application is hopefully straightforward and does not warrant much explanation. The application launches a graphical user interface (gui) that controls each element explained in an environment similar to a digital audio workstation (daw). Each element is played back from a multichannel soundfile. A timer in minutes:seconds and a visual metronome at 75 beats per minute (for the interrupt sections) is provided for coordination. Images of each tab of the gui and the directory structure of the application resources are provided below. The channels of the soundfile (24 in total) correspond to the faders from left to right.

To launch the application, execute `ostinato_and_interrupt_main.scd` in SuperCollider after booting the server (on linux, this is achieved by pressing cmd+enter with the cursor anywhere within the code block).

The “generate” button regenerates the piece. By default it will generate the original version included with this score, however the random seed can be changed if someone is so bold as to try to create a new version. Note that the application was written to create the given version, but should hopefully function properly when generating new version even though it has not been extensively tested. Regenerating the piece creates / replaces several files: most importantly `ostinato_and_interrupt.wav` which is needed for playback and should open in most daws (tested with Audacity). It also regenerates the Lilypond files which can be rendered and engraved using Lilypond.

For the generation function to work properly, the application requires that single samples be placed in the `samples/` folder within the directory tree given on the following page. Ideally, there should be several samples for each sound. While the application does not adhere to any naming conventions for the sample files, changing the names of the folders will break the application. The number prefix of each folder applies to the string / part number. With exception of the `strings_harmonics/` folder, the samples within a folder should be different versions of the same sound.

**`ostinato_bass/`:** each folder contains samples of the given bass note of the ostinato from highest (1) to lowest (6) (the descending bassline is described in the beginning of the instructions).

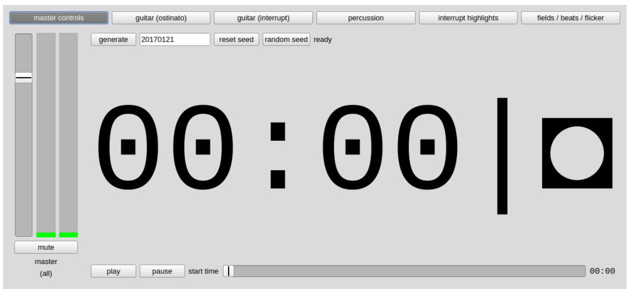
**`strings_open/`:** each folder contains samples of the given open string from highest (1) to lowest (6).

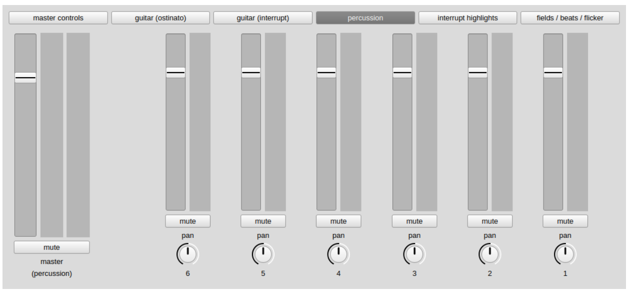
**`strings_open/`:** each folder contains samples of different natural harmonics of the given string from highest (1) to lowest (6). These are the only folders of the guitar samples that will contain sounds of different pitches. Note that if several samples of each harmonic are recorded, then there must be the same number of each harmonic in each folder; e.g. two samples of the 2nd harmonic, two samples of the 3rd harmonic, etc.

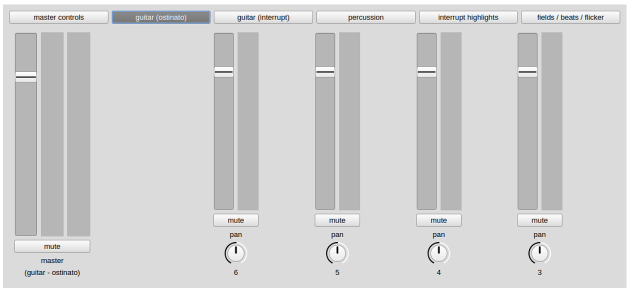
**`strings_muted/`:** each folder contains samples of the given string muted from highest (1) to lowest (6).

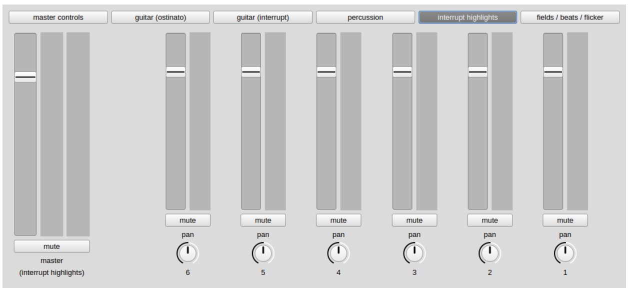
**`percussion/`:** each folder contains samples of one of the six different percussion instruments.

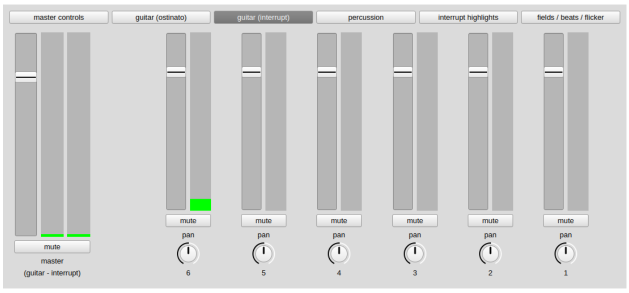
The primary source code for the application is appended at the end of this score and can be can be downloaded from a git repository at `https://gitea.unboundedpress.org/mwinter/ostinato_and_interrupt`. The whole package including all audio file resources is available upon request or can be downloaded from `https://gitea.unboundedpress.org/mwinter/ostinato_and_interrupt/releases`. Note that the package comes with a pregenerated version of the piece (the multichannel soundfile and all the Lilypond files) using an included sample set. This original sample set (recorded in August of 2017) was not recorded in ideal conditions and was used simply to audition the piece. The generation of this document (using LaTex) and the musical parts (in Lilypond) contain version dates in order to help track changes and the git repository will also detail commit changes. The piece was written using SuperCollider version 3.8.0 and Lilypond version 2.18.2.

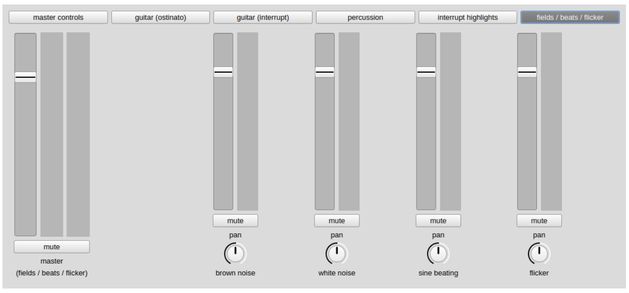












```
ostinato_and_interrupt_source/
├── supercollider/
│   ├── ostinato_and_interrupt_main.scd
│   ├── ostinato_and_interrupt_generator_synthdef.scd
│   ├── ostinato_and_interrupt_nrt_generator_function.scd
│   ├── ostinato_and_interrupt_lilypond_generator_function.scd
│   ├── ostinato_and_interrupt_player_synthdef.scd
│   ├── ostinato_and_interrupt_gui_generator_function.scd
│   └── gen_data_resources/
│       ├── ostinato_and_interrupt_osc
│       └── ostinato_and_interrupt_data.wav
├── audio/
│   └── ostinato_and_interrupt.wav
├── lilypond/
│   ├── ostinato_and_interrupt_lilypond_score_template.ly
│   ├── ostinato_and_interrupt_lilypond_guitar_part.ly
│   ├── ostinato_and_interrupt_lilypond_percussion_part.ly
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_1.ly
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_2.ly
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_3.ly
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_4.ly
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_5.ly
│   └── ostinato_and_interrupt_lilypond_ensemble_part_6.ly
├── score_pdfs/
│   ├── ostinato_and_interrupt_lilypond_guitar_part.pdf
│   ├── ostinato_and_interrupt_lilypond_percussion_part.pdf
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_1.pdf
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_2.pdf
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_3.pdf
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_5.pdf
│   ├── ostinato_and_interrupt_lilypond_ensemble_part_4.pdf
│   └── ostinato_and_interrupt_lilypond_ensemble_part_6.pdf
└── samples/
```

```
ostinato_and_interrupt_source/
└── samples/
    ├── ostinato_bass/
    │   ├── 1_bass/
    │   ├── 2_bass/
    │   ├── 3_bass/
    │   ├── 4_bass/
    │   ├── 5_bass/
    │   └── 6_bass/
    ├── strings_open/
    │   ├── 1_open/
    │   ├── 2_open/
    │   ├── 3_open/
    │   ├── 4_open/
    │   ├── 5_open/
    │   └── 6_open/
    ├── strings_harmonics/
    │   ├── 1_harmonic/
    │   ├── 2_harmonic/
    │   ├── 3_harmonic/
    │   ├── 4_harmonic/
    │   ├── 5_harmonic/
    │   └── 6_harmonic/
    ├── strings_muted/
    │   ├── 1_muted/
    │   ├── 2_muted/
    │   ├── 3_muted/
    │   ├── 4_muted/
    │   ├── 5_muted/
    │   └── 6_muted/
    └── percussion/
        ├── 1_percussion/
        ├── 2_percussion/
        ├── 3_percussion/
        ├── 4_percussion/
        ├── 5_percussion/
        └── 6_percussion/
```

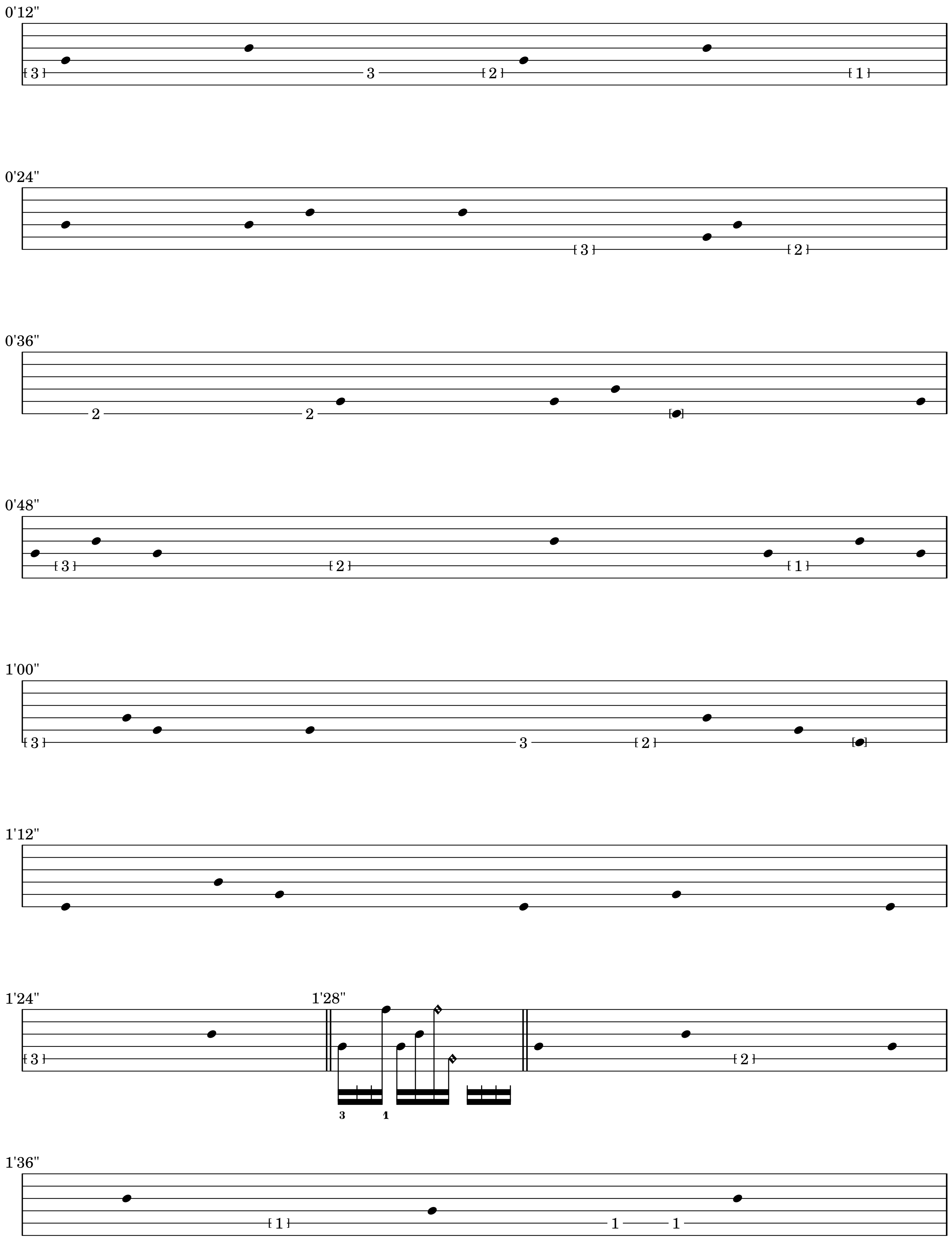
*I would like to extend a special thanks to Jose Manuel Alcantara and Alex Bruck for their help during this piece: Jose Manuel for his encouragement and for lending me a guitar; and Alex for his generosity in answering many, many questions and his patience when I just needed to someone to listen as I talked through ideas... this piece was very much shaped by our friendship...*

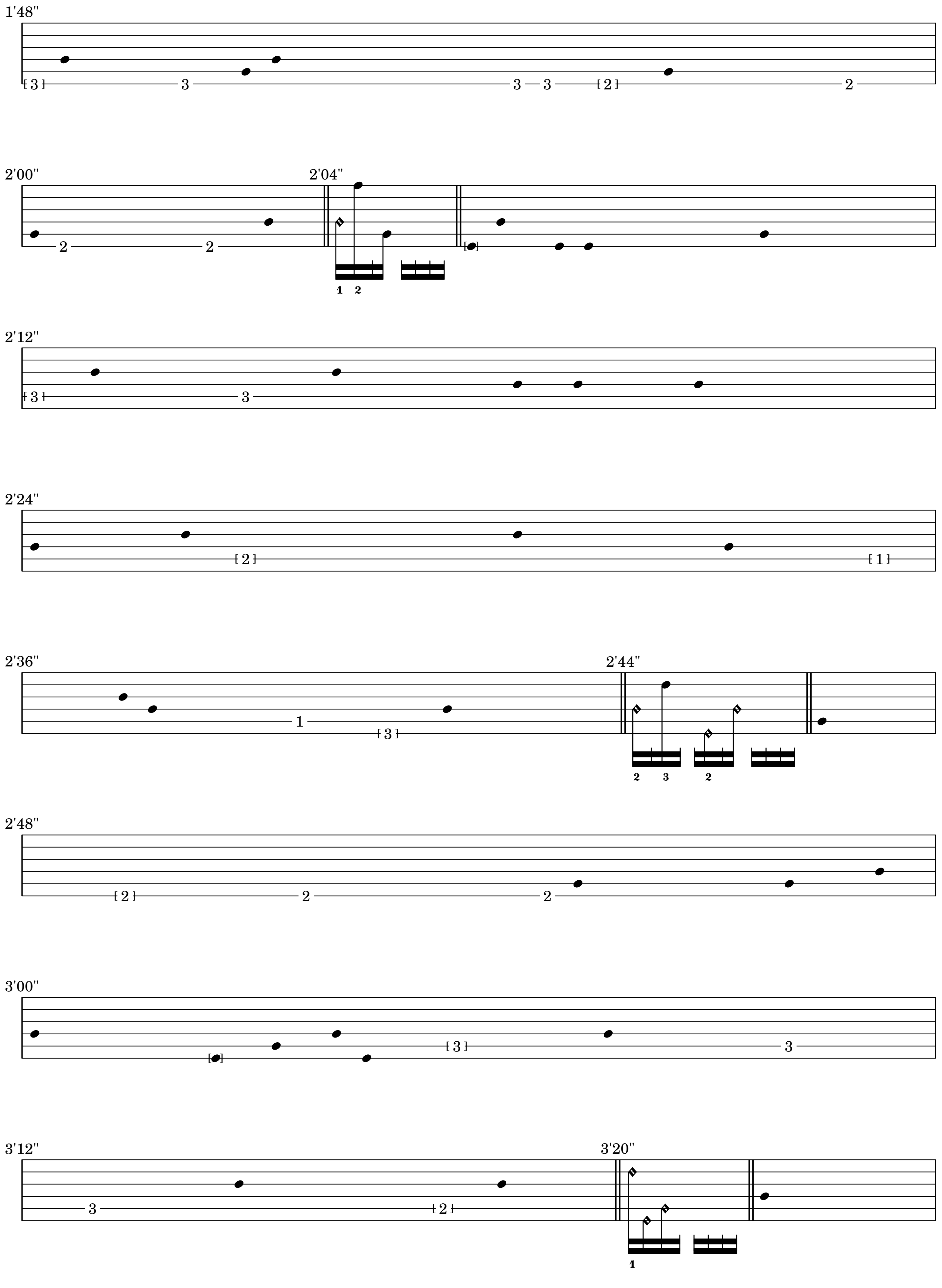
version generated: 2020.11.20

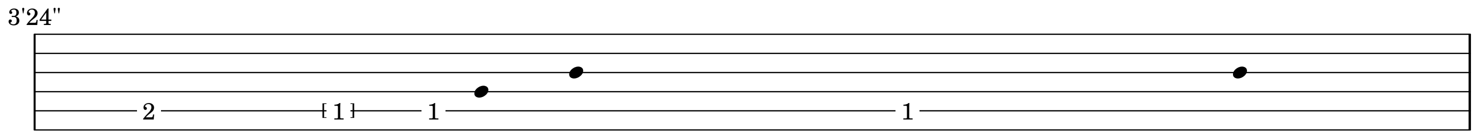
# *ostinato and interrupt*

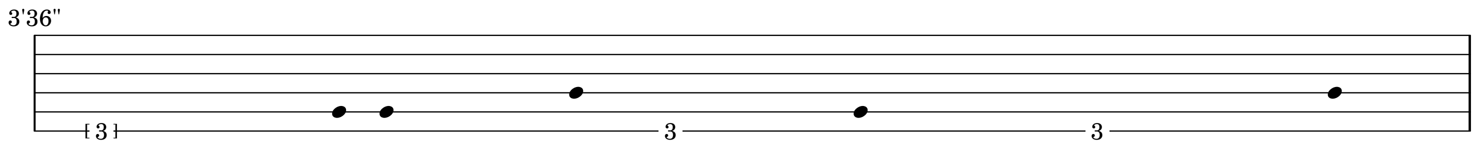
michael winter (mexico city, mx; 2017)
version generated: 2017.08.23

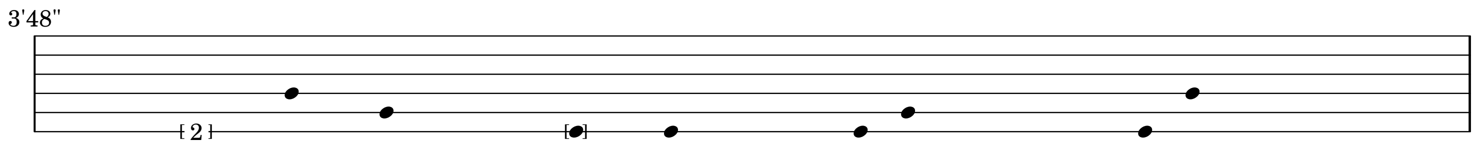
guitar/all

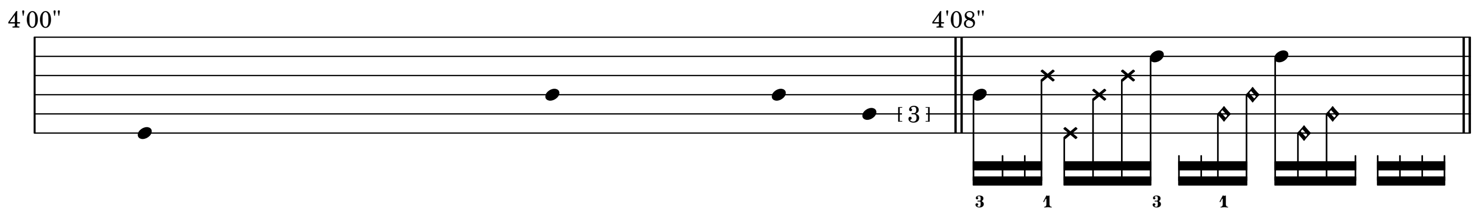


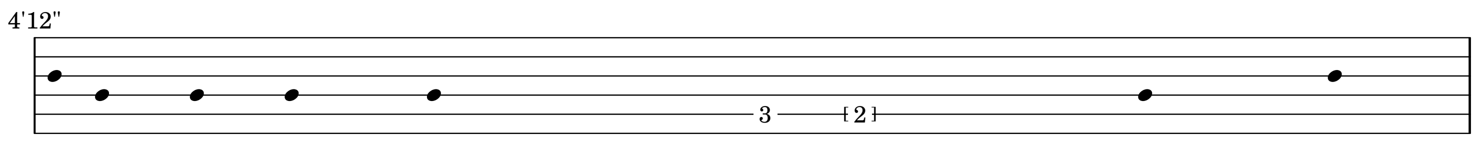

1'48"
2'00"
2'04"
2'12"
2'24"
2'36"
2'44"
2'48"
3'00"
3'12"
3'20"

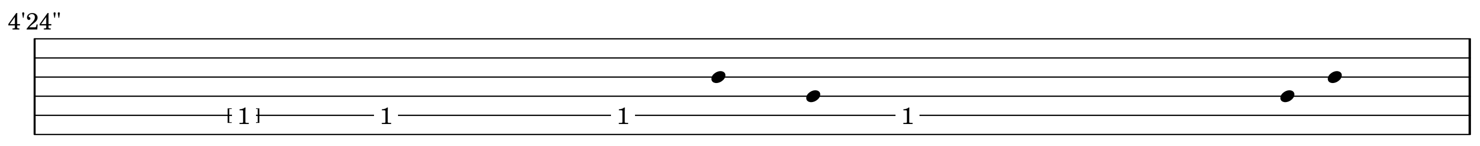

3'24"
2
{1}
1
1

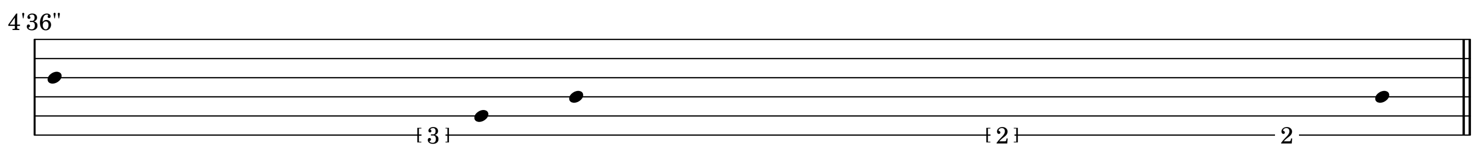

3'36"
{3}
3
3

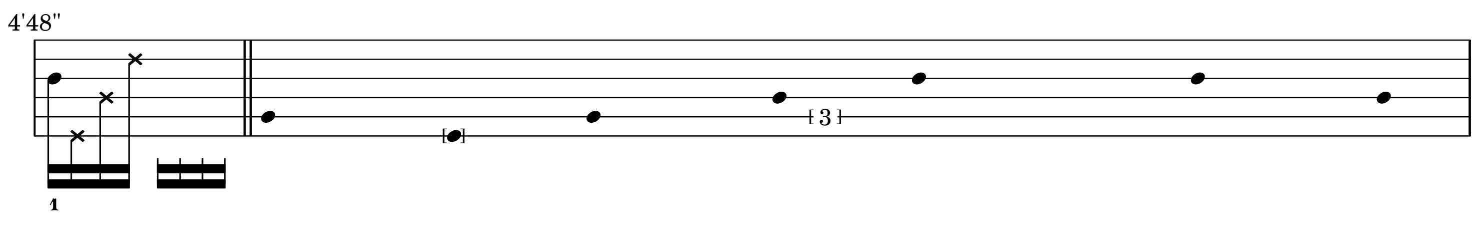

3'48"
{2}

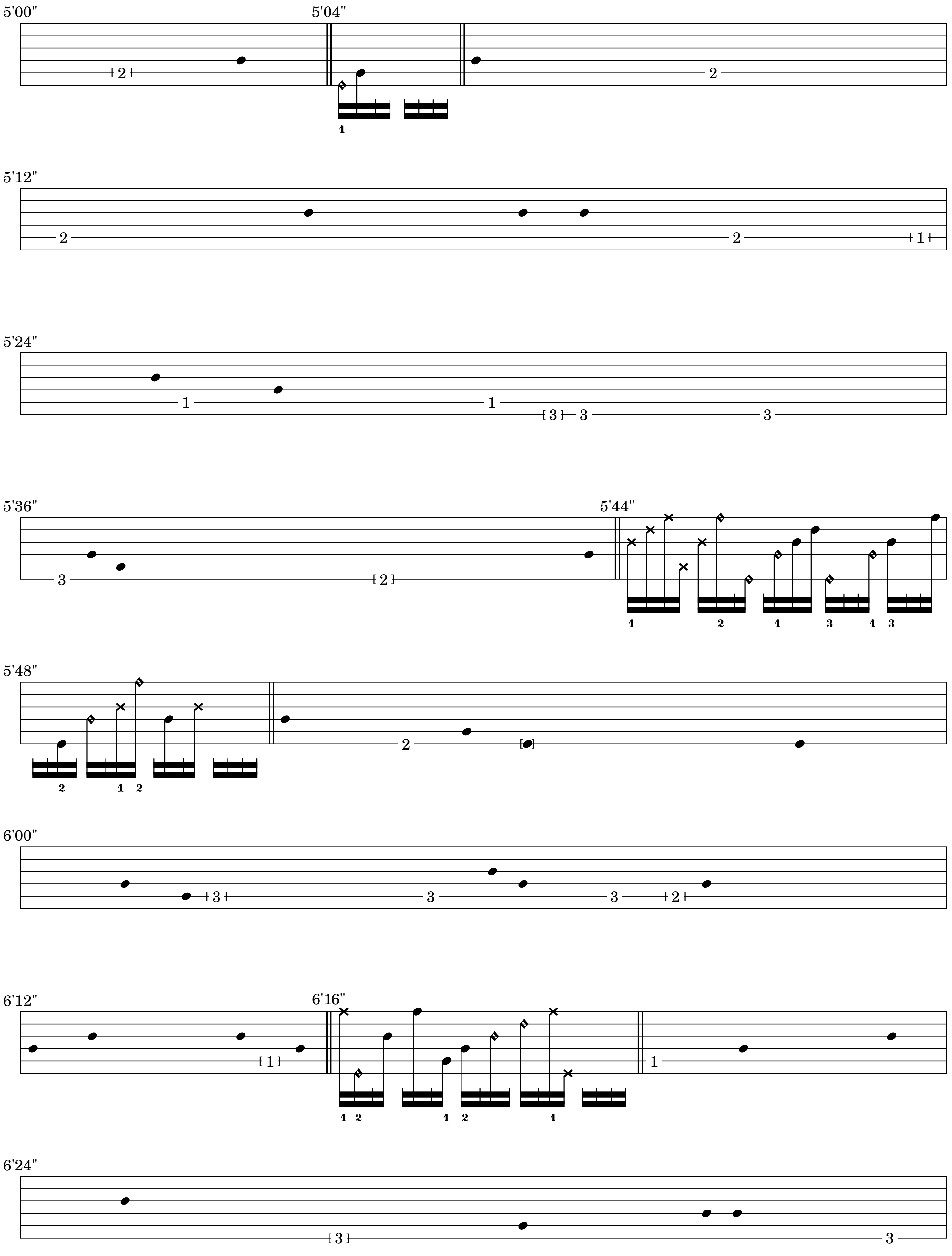

4'00"
{3}
4'08"
3
1
3
1


4'12"
3
{2}

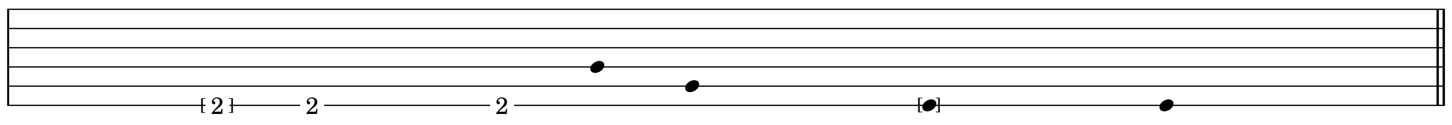

4'24"
{1}
1
1
1


4'36"
{3}
{2}
2

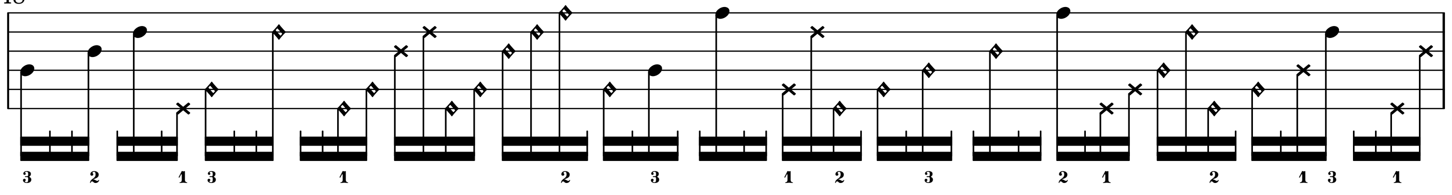

4'48"
1
{3}


5'00"
5'04"
5'12"
5'24"
5'36"
5'44"
5'48"
6'00"
6'12"
6'16"
6'24"

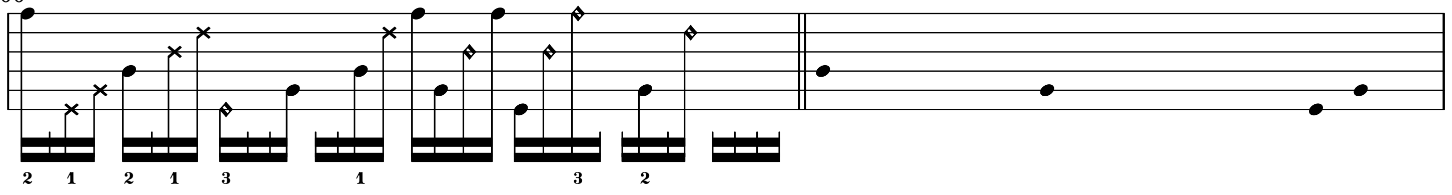
6'36"



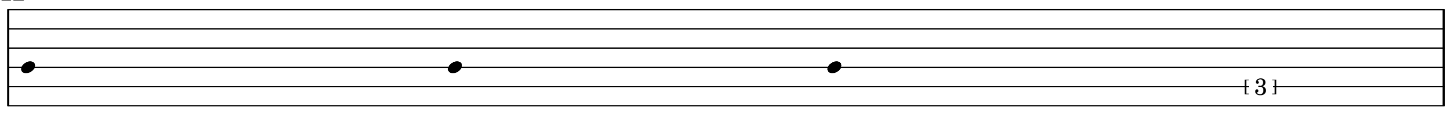
6'48"



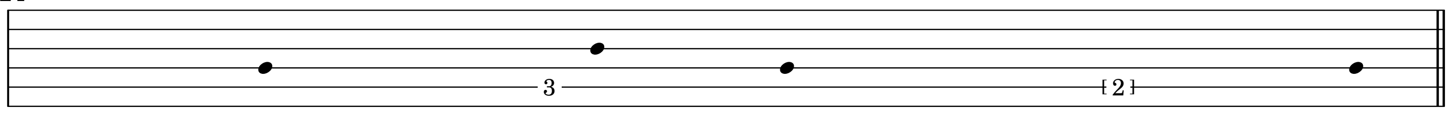
7'00"



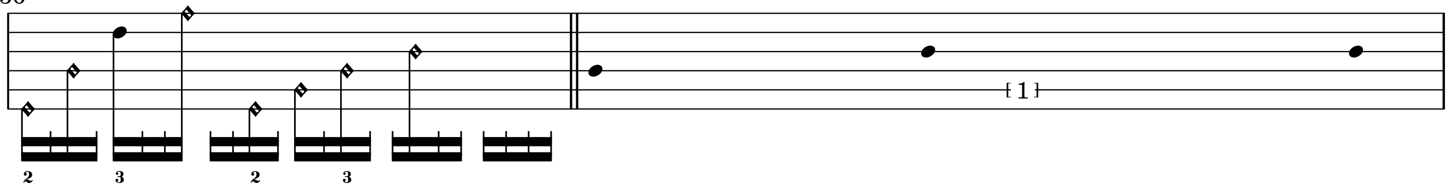
7'12"



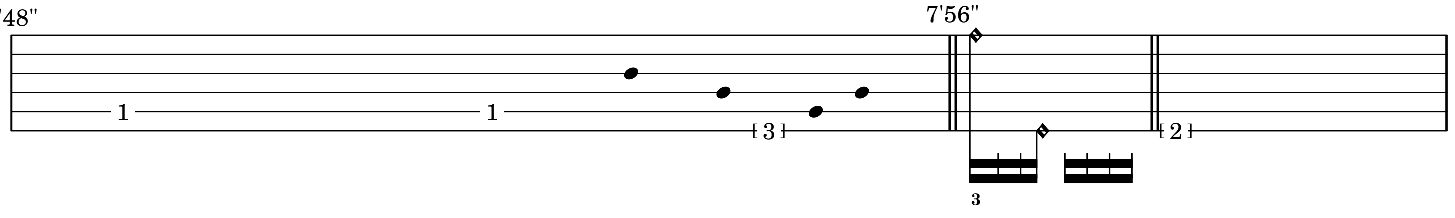
7'24"



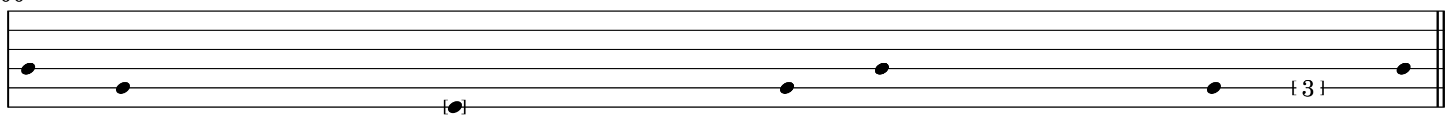
7'36"



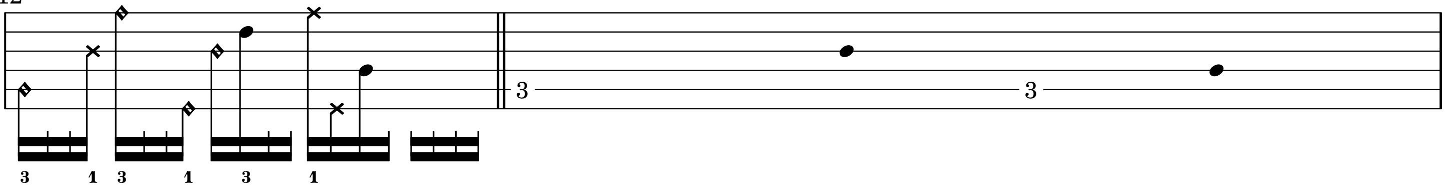
7'48"



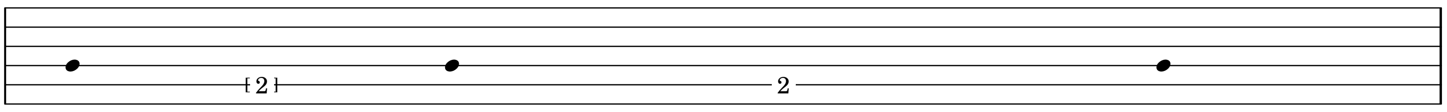
8'00"



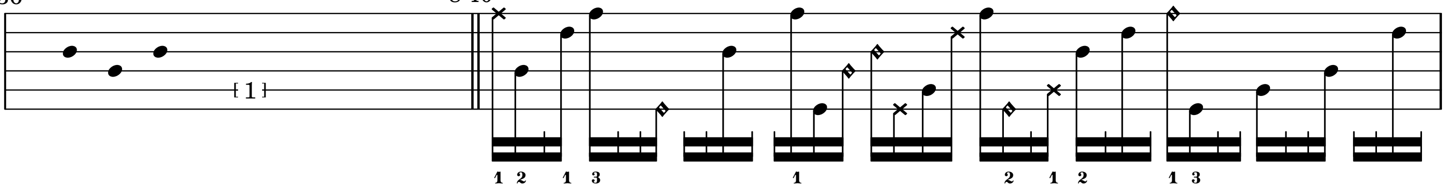
8'12"



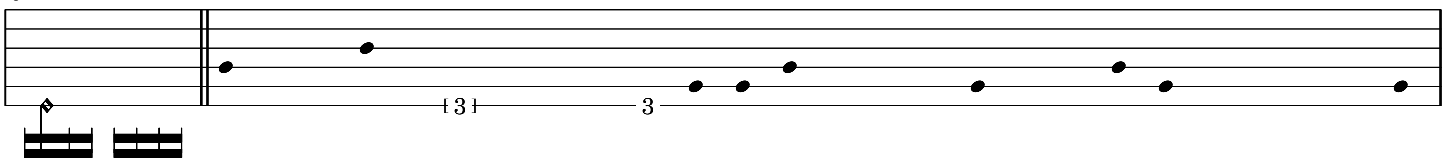
8'24"



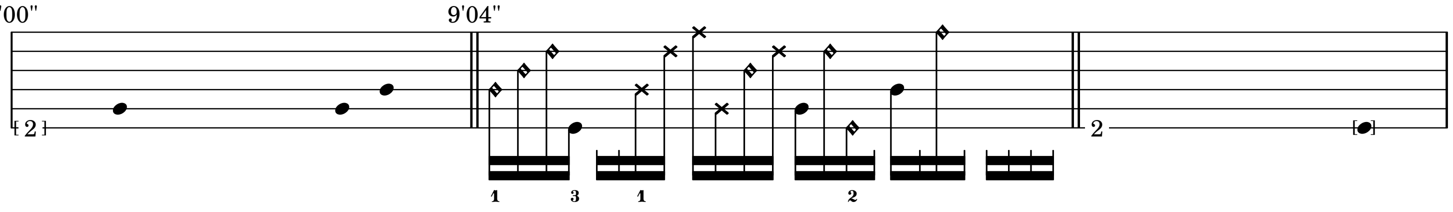
8'36" 8'40"



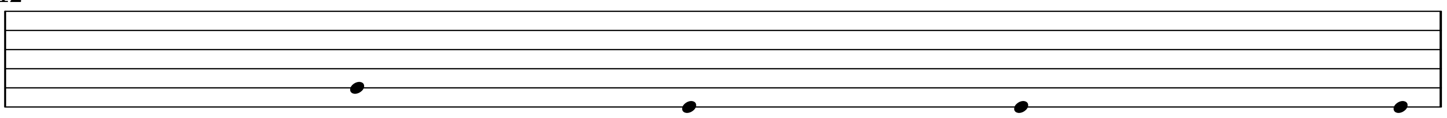
8'48"



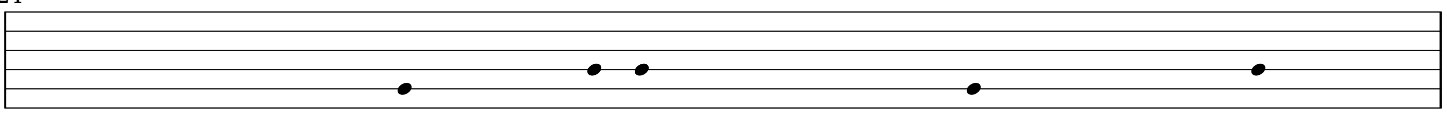
9'00" 9'04"



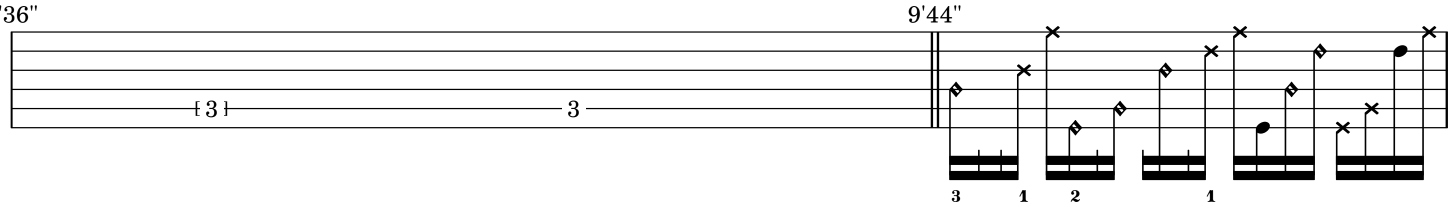
9'12"

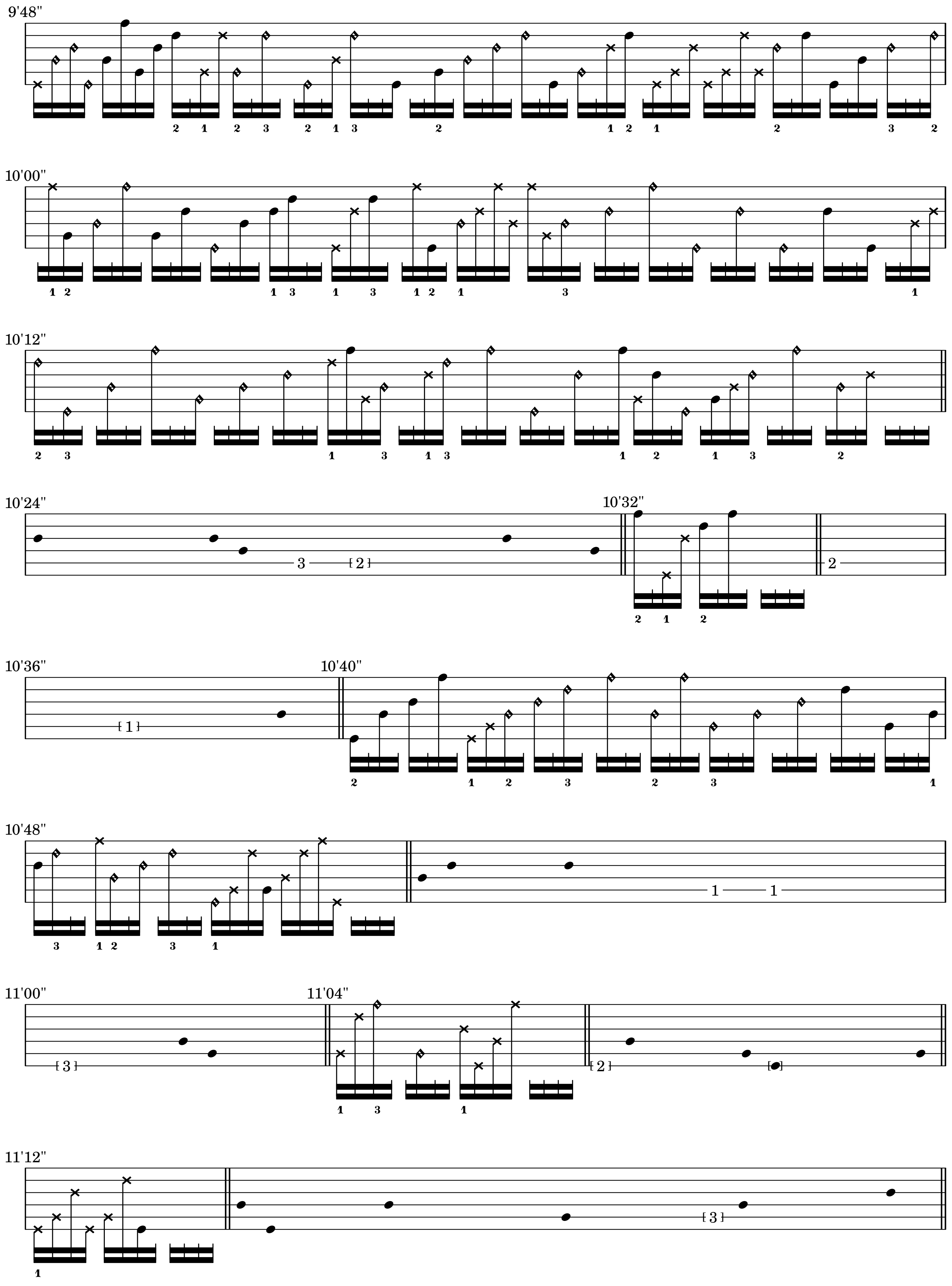


9'24"

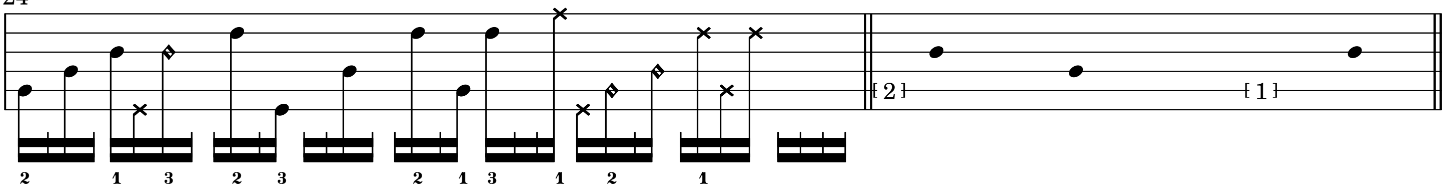


9'36" 9'44"

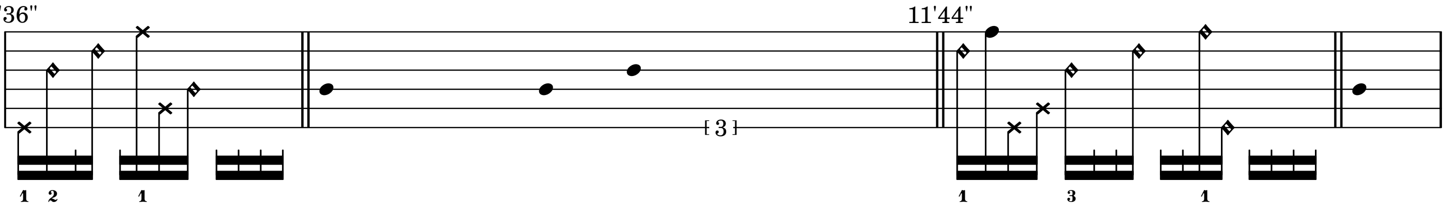



9'48"
10'00"
10'12"
10'24"
10'32"
10'36"
10'40"
10'48"
11'00"
11'04"
11'12"

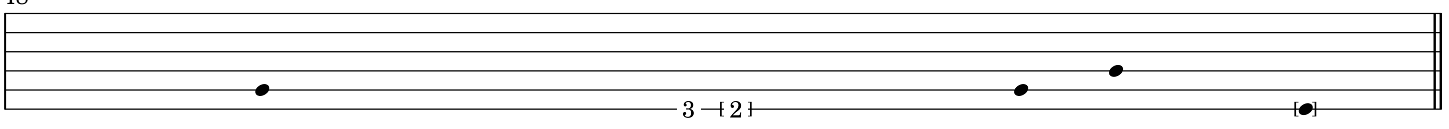
11'24"



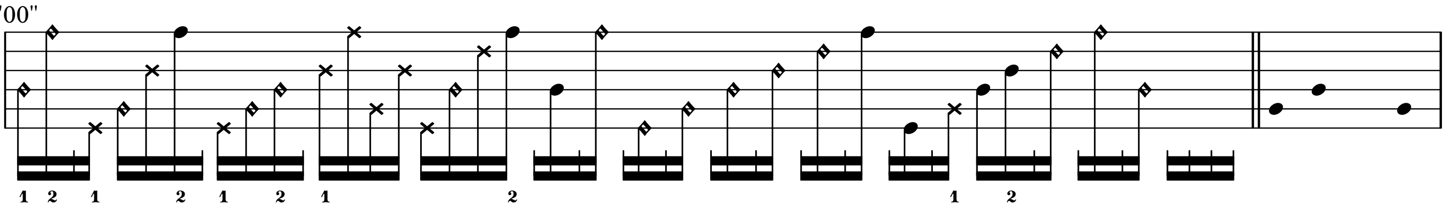
11'36"



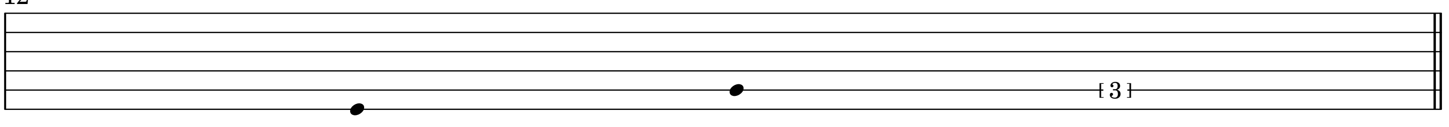
11'48"



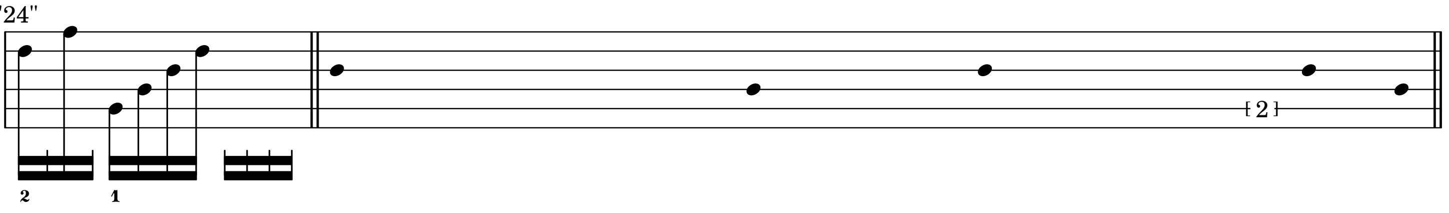
12'00"



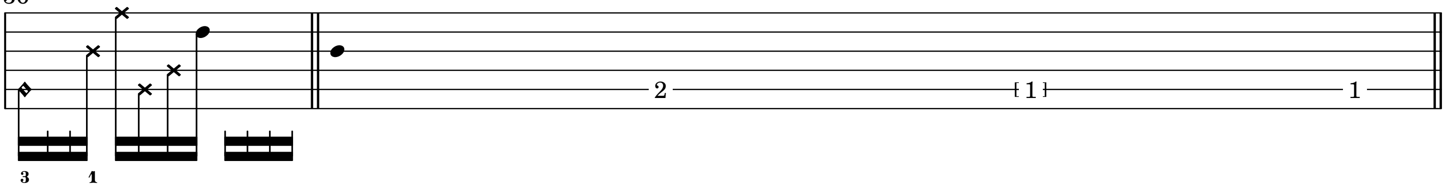
12'12"



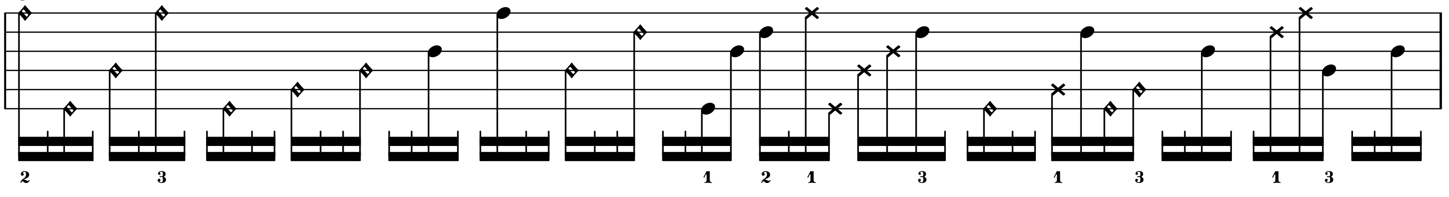
12'24"



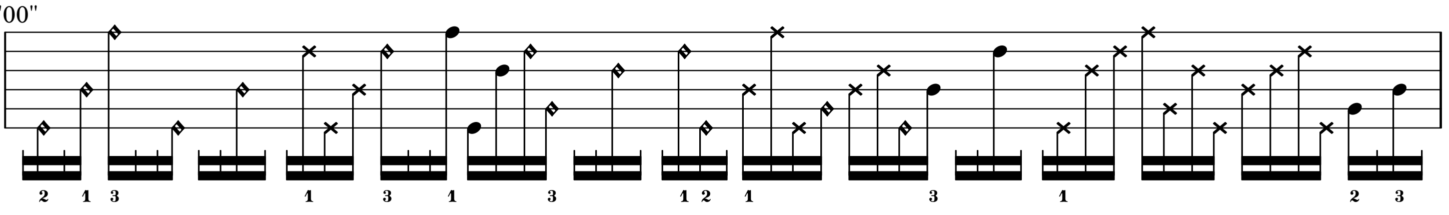
12'36"



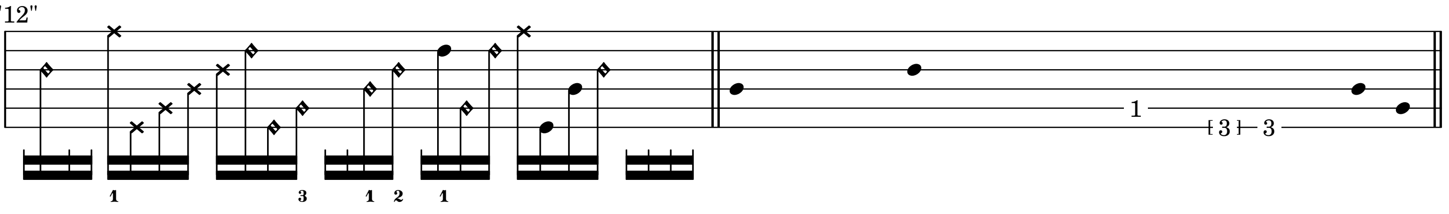
12'48"



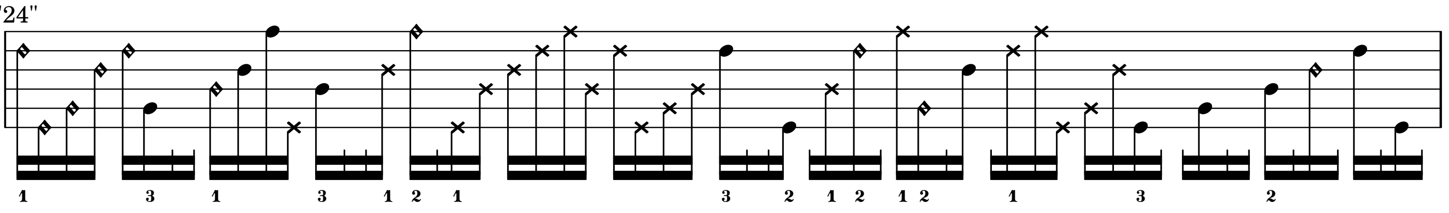
13'00"



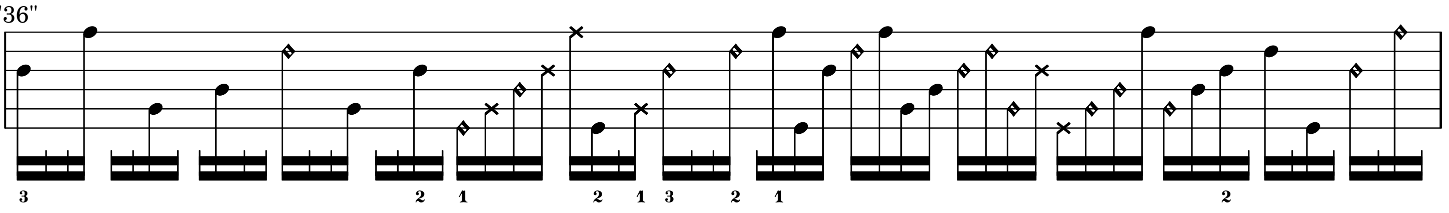
13'12"



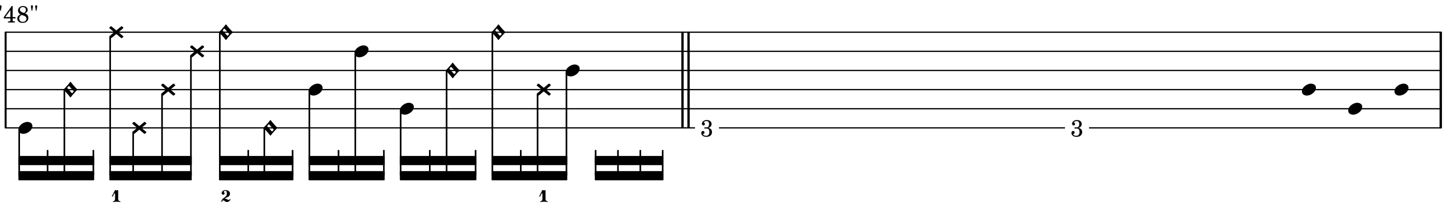
13'24"



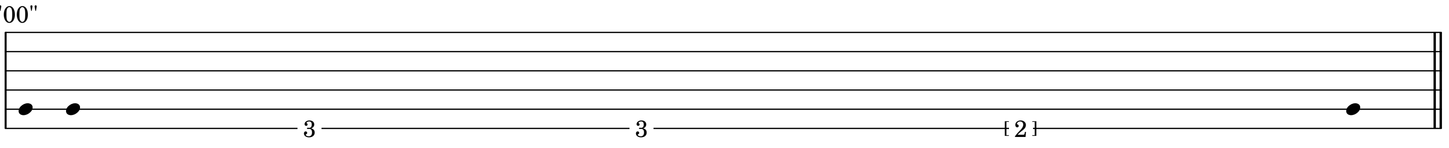
13'36"



13'48"

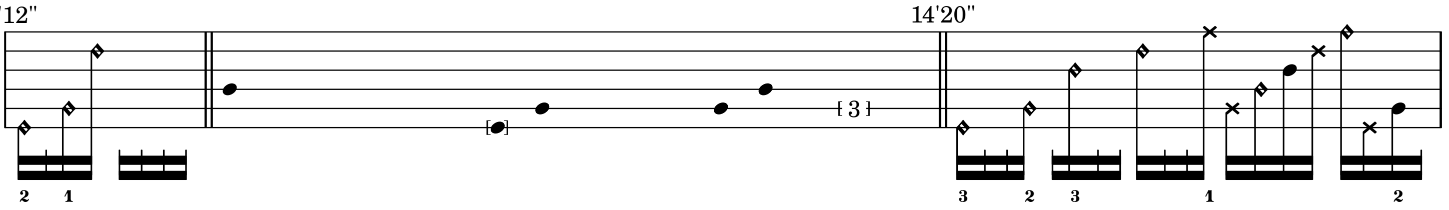


14'00"

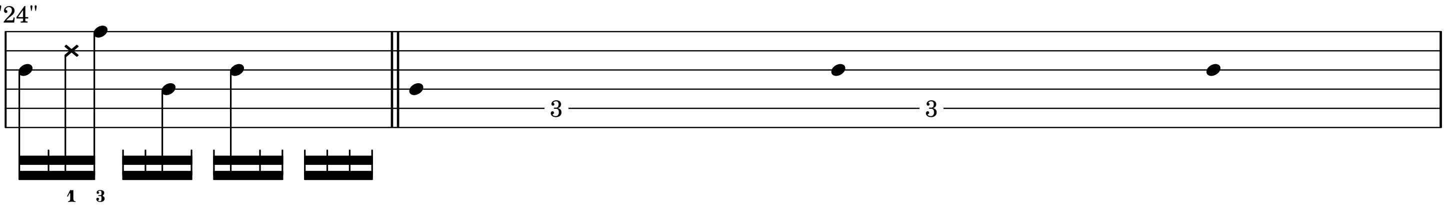


14'12"

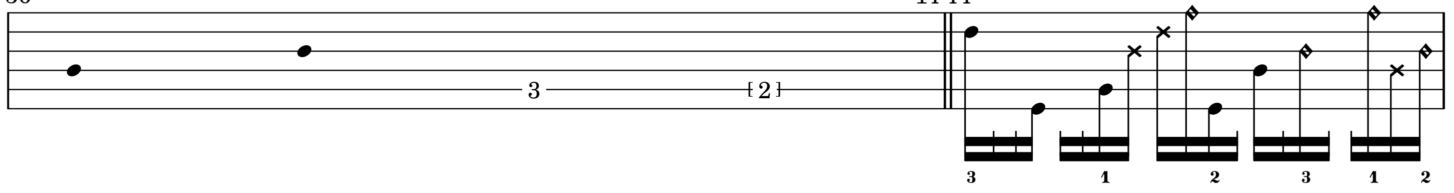
14'20"



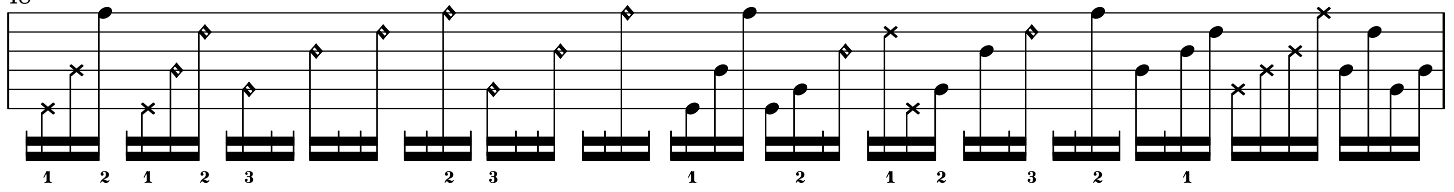
14'24"



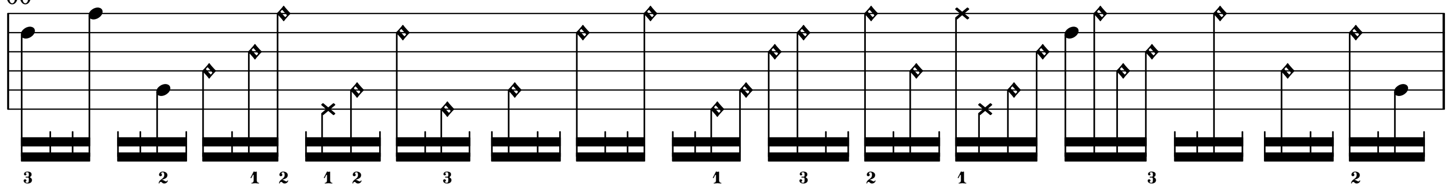
14'36"



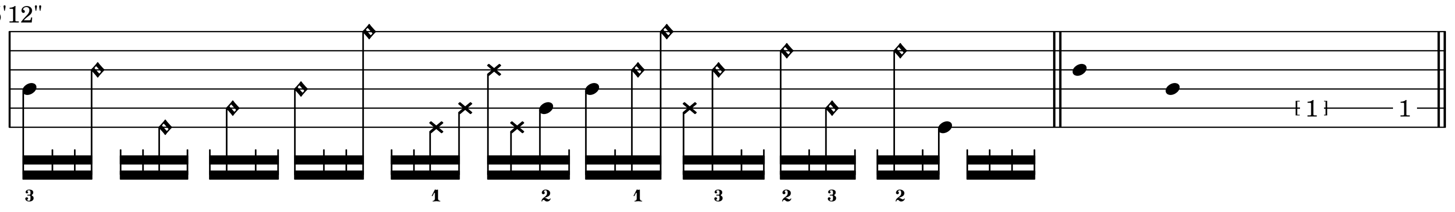
14'48"



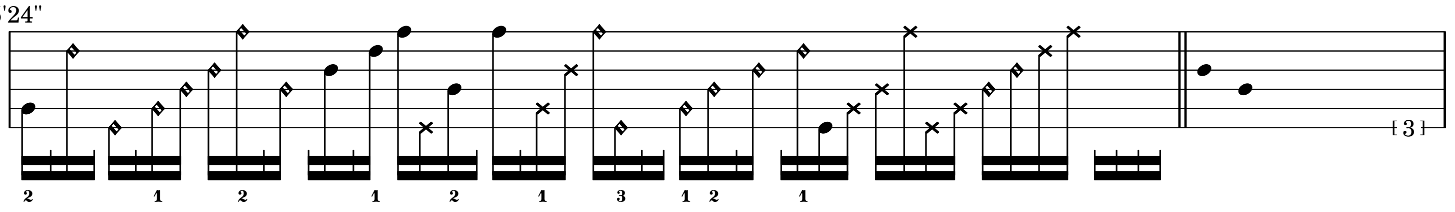
15'00"



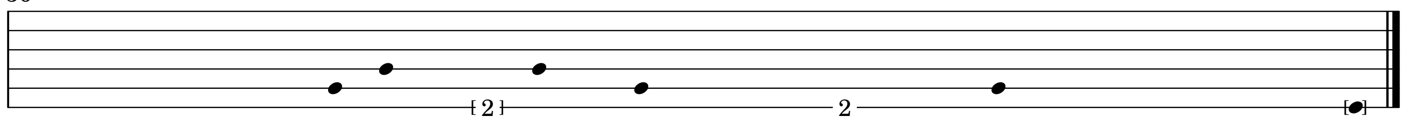
15'12"



15'24"



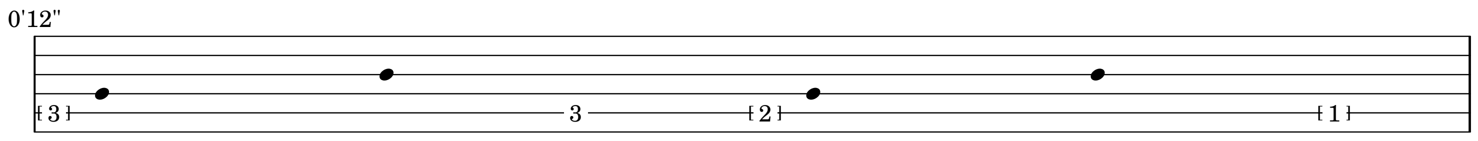
15'36"

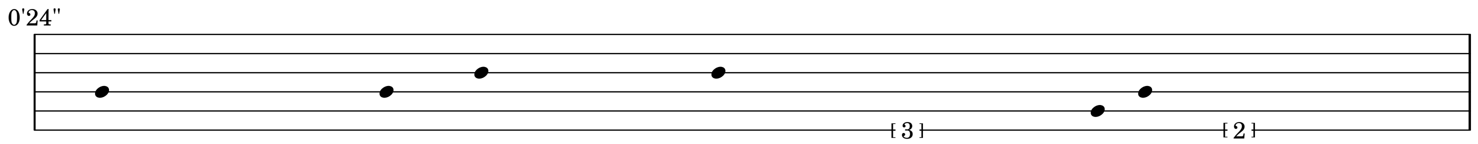


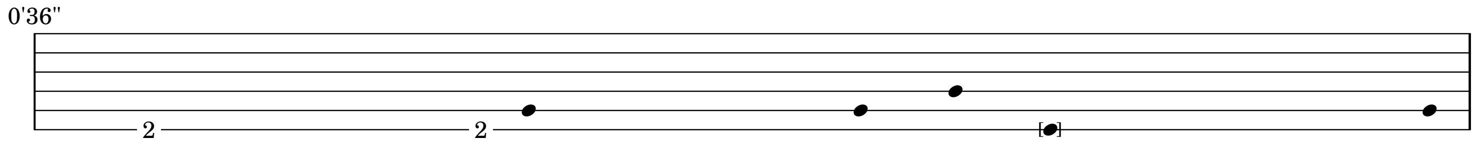
# *ostinato and interrupt*

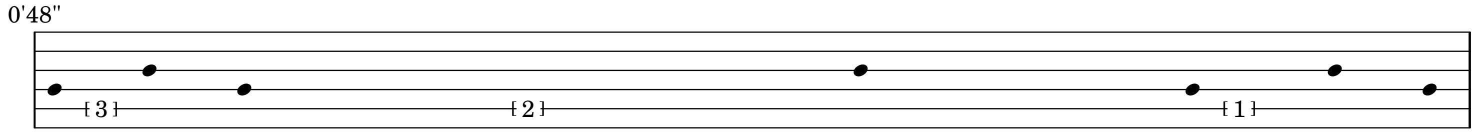
michael winter (mexico city, mx; 2017)
version generated: 2017.08.23

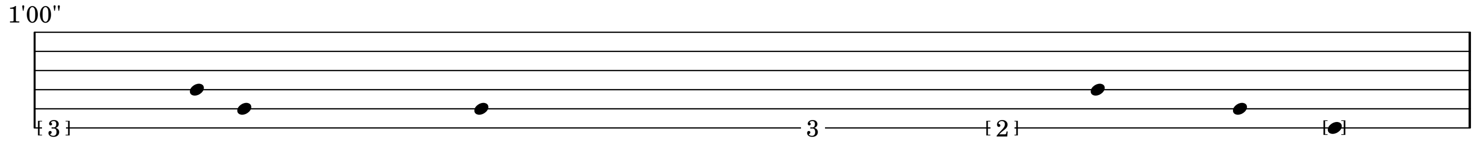
percussion

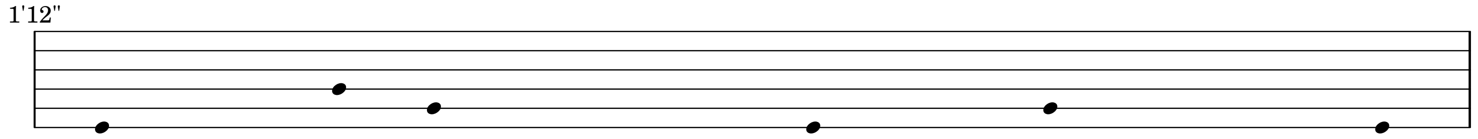


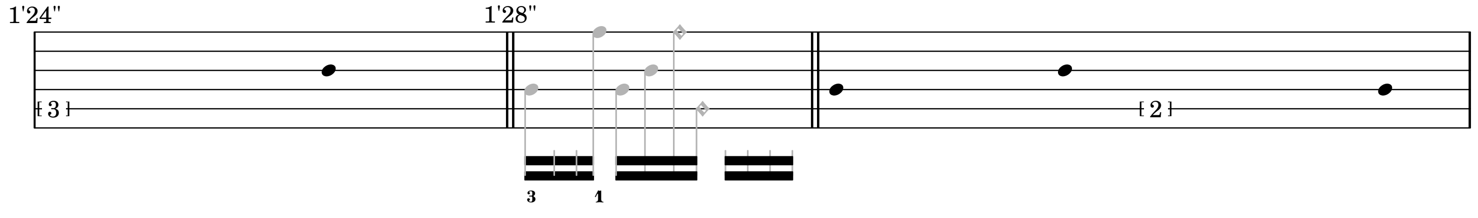


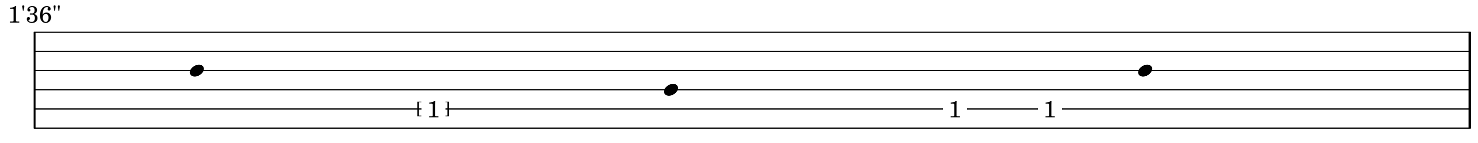


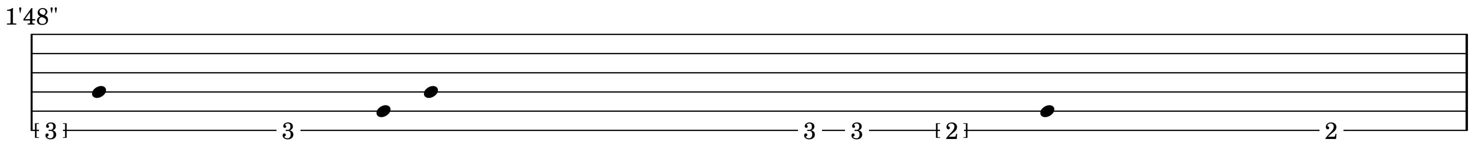


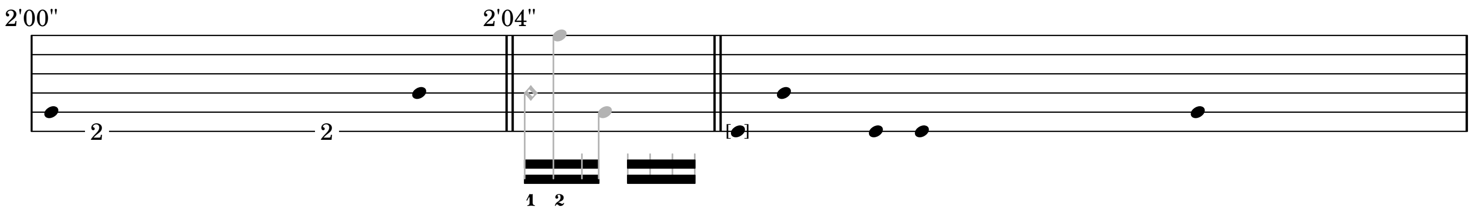


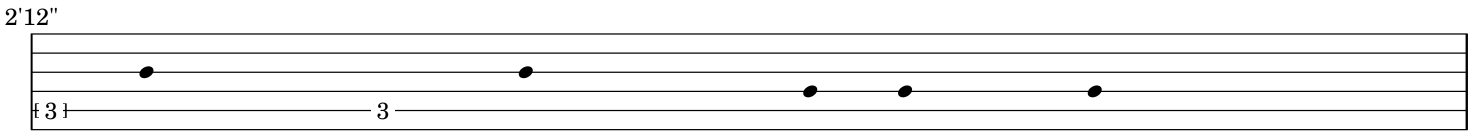


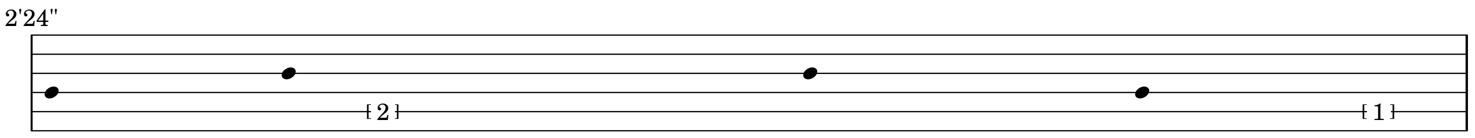


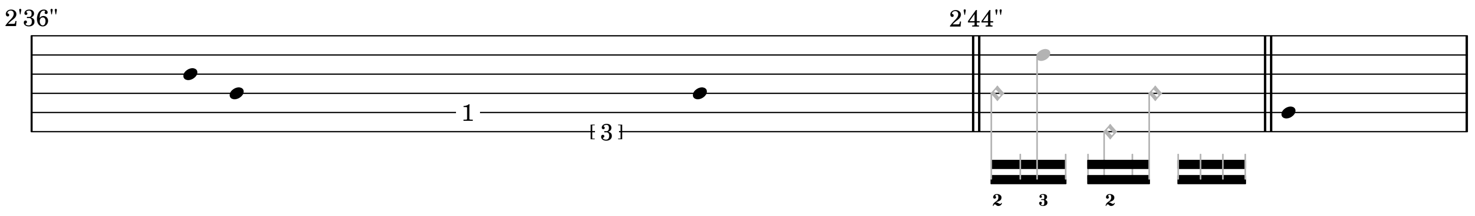


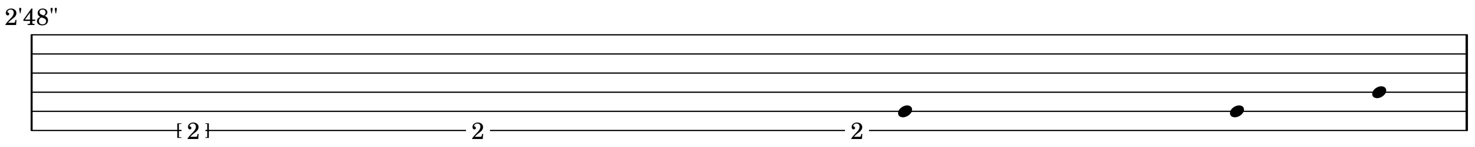

1'48"

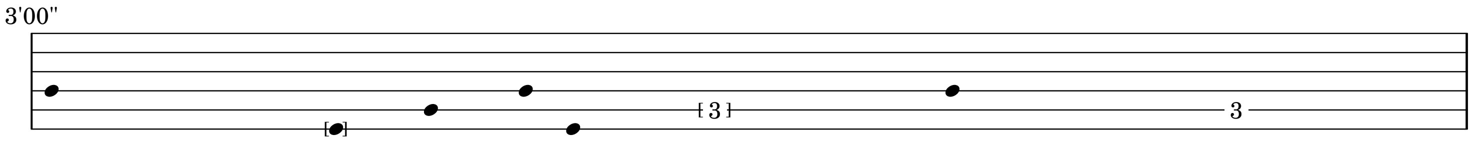

2'00"
2'04"

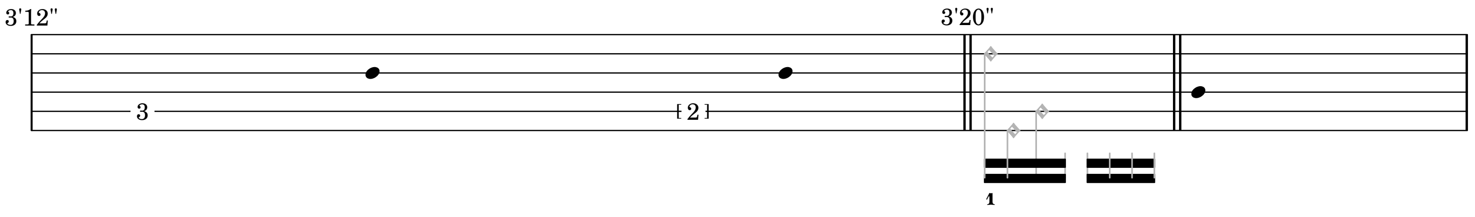

2'12"

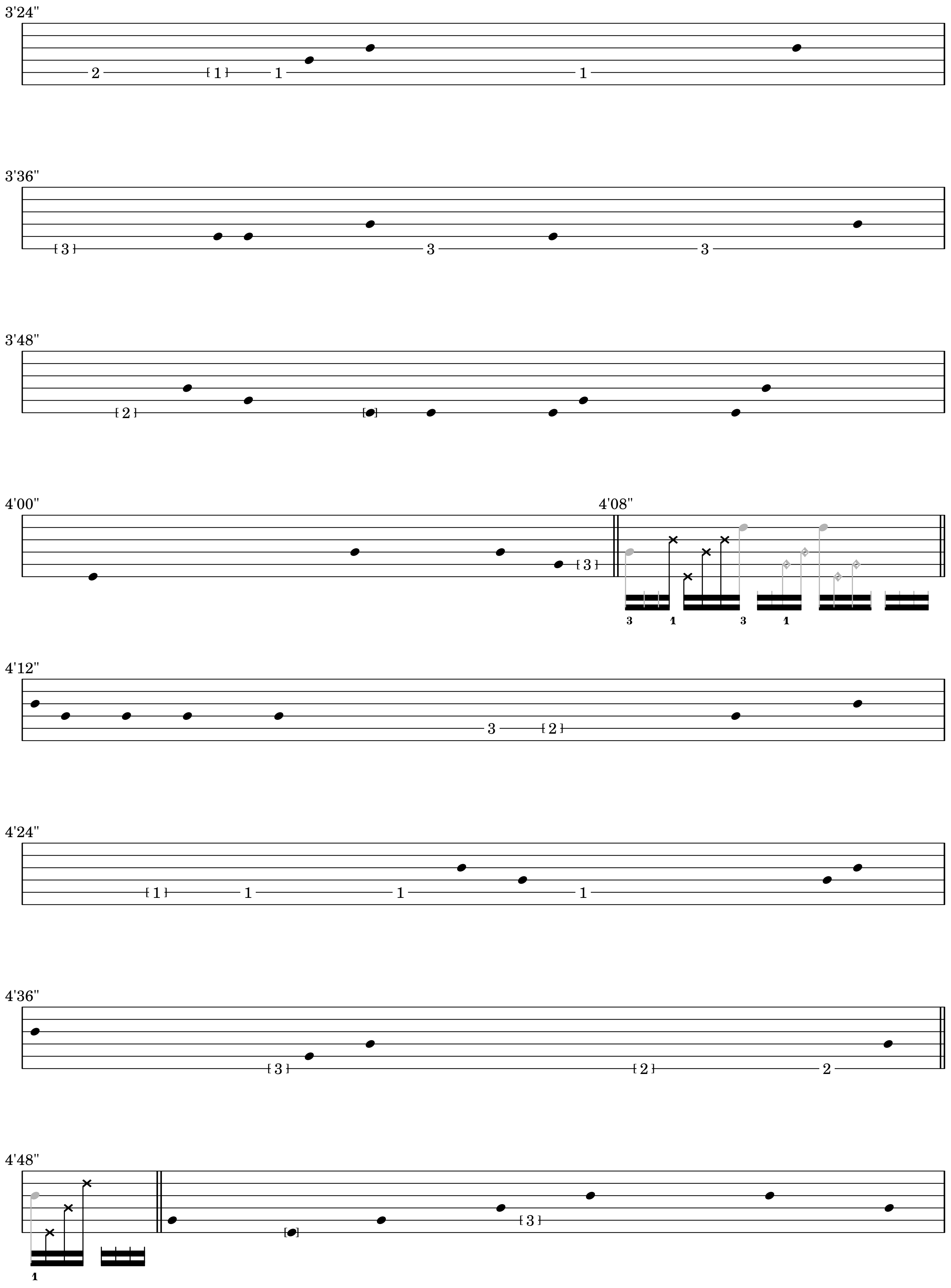

2'24"

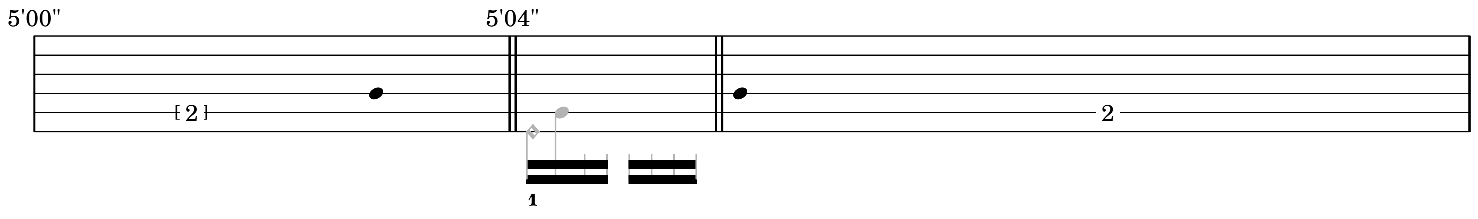

2'36"
2'44"

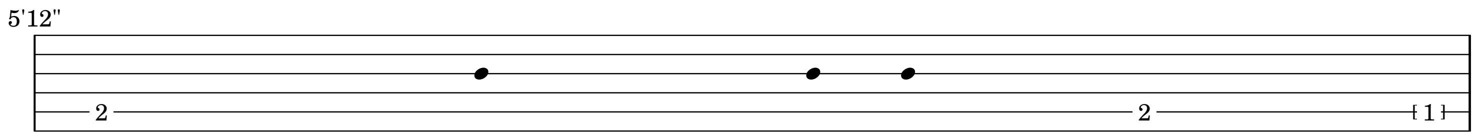

2'48"

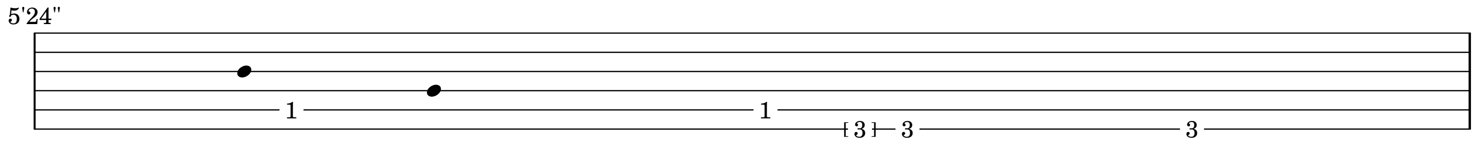

3'00"

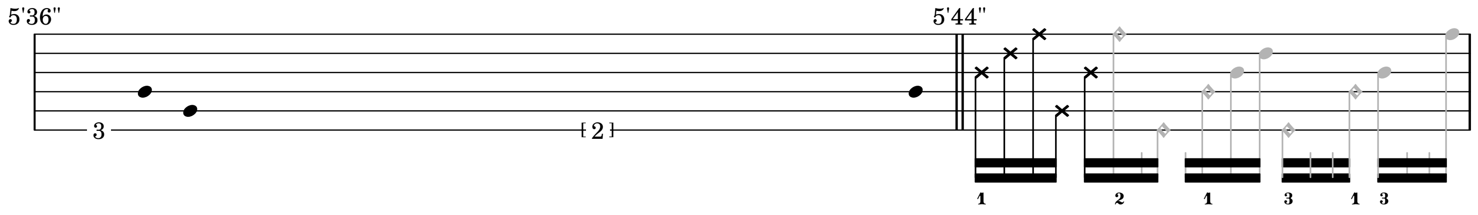

3'12"
3'20"

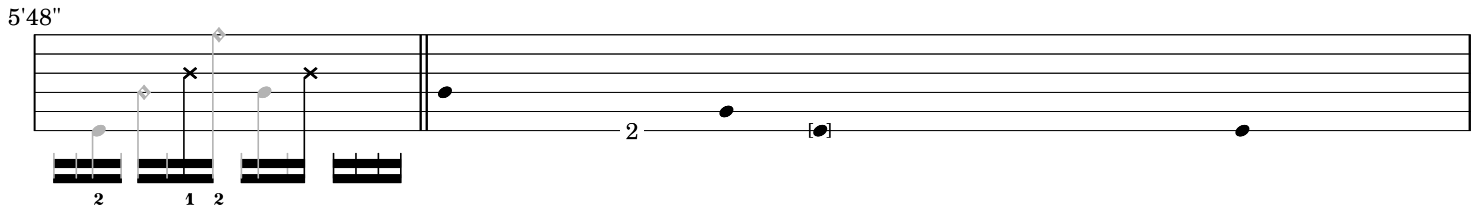

3'24"
3'36"
3'48"
4'00"
4'08"
4'12"
4'24"
4'36"
4'48"

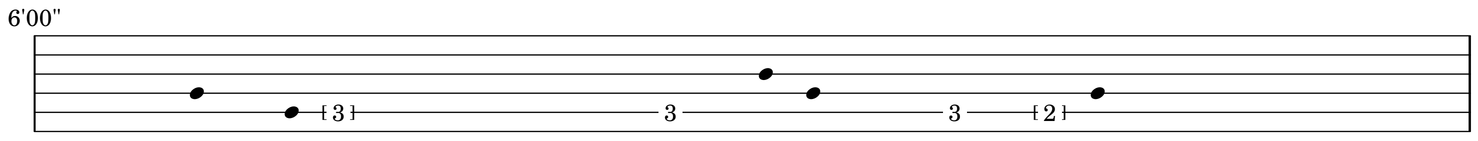

5'00"
5'04"
2
2
1

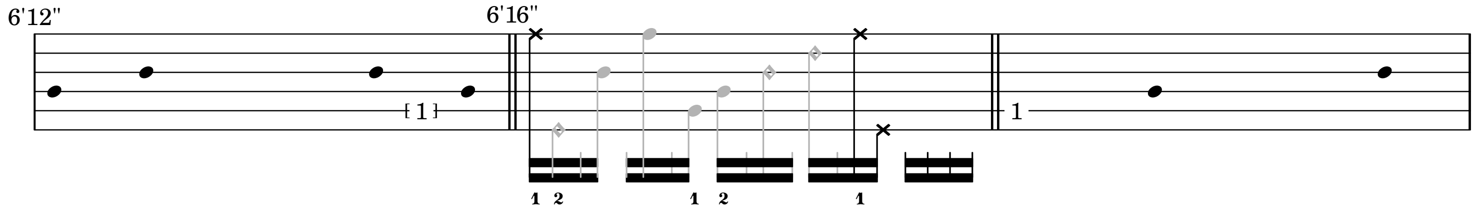

5'12"
2
2
1

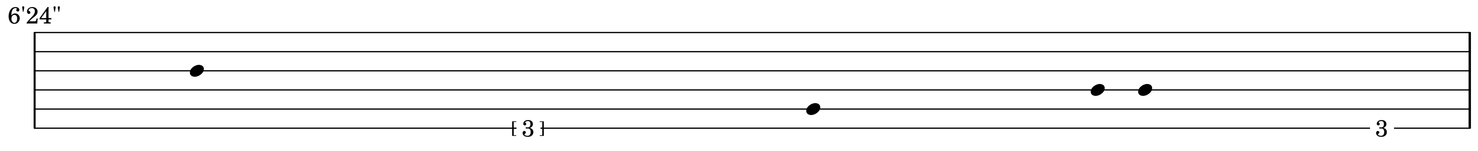

5'24"
1
1
3
3
3


5'36"
5'44"
3
2
1
2
1
3
1
3

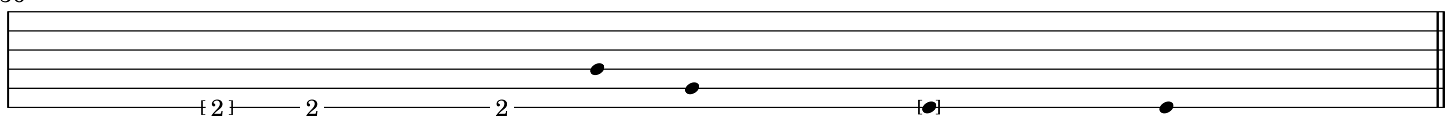

5'48"
2
1
2
2


6'00"
3
3
3
2

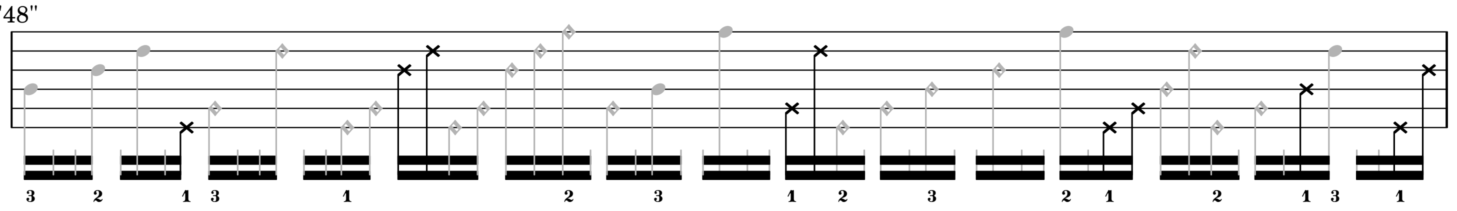

6'12"
6'16"
1
1
2
1
2
1
1


6'24"
3
3

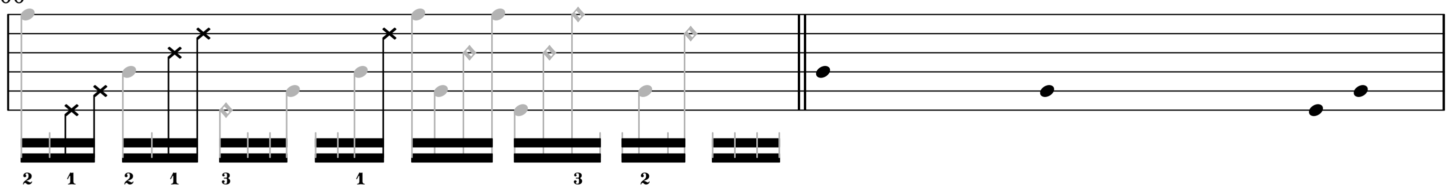
6'36"



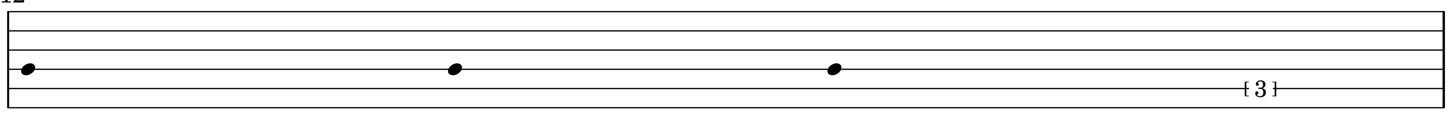
6'48"



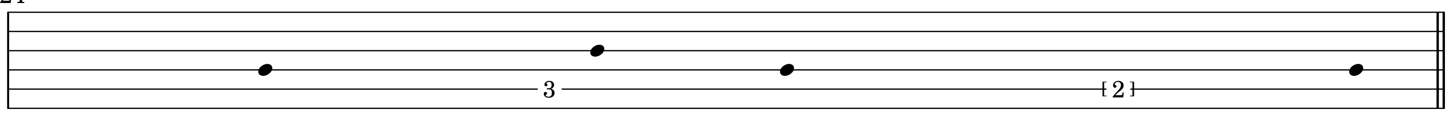
7'00"



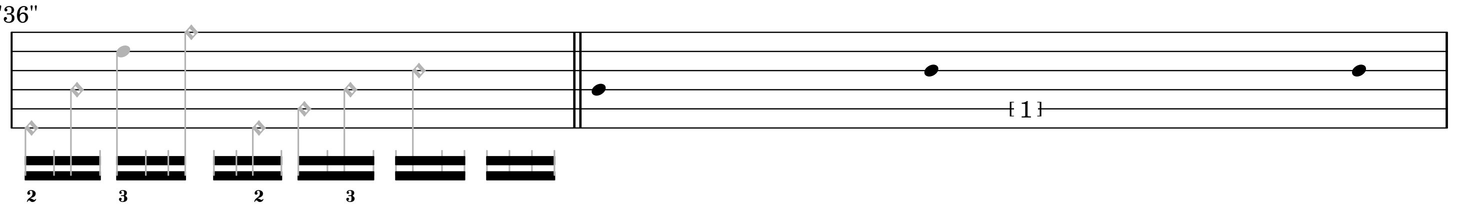
7'12"



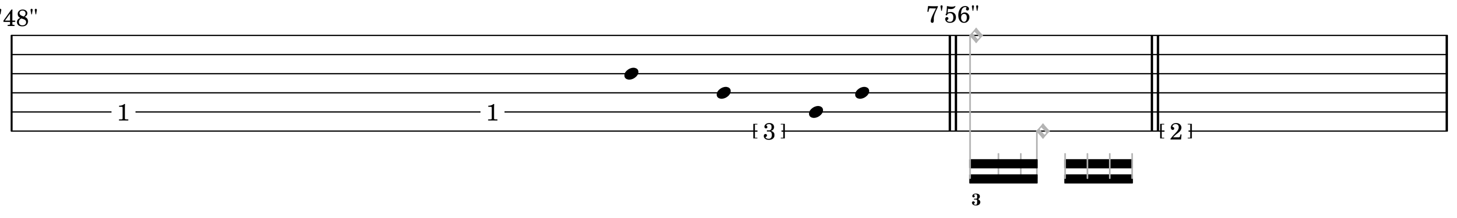
7'24"



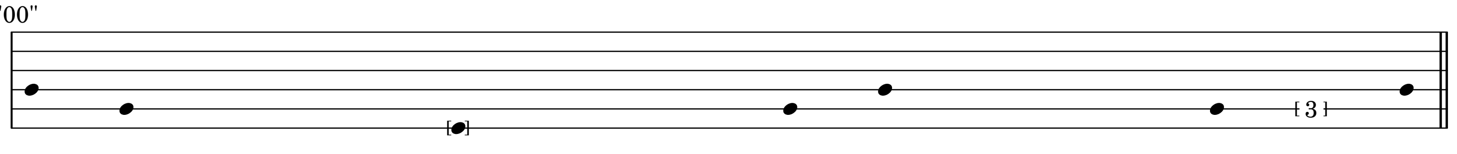
7'36"

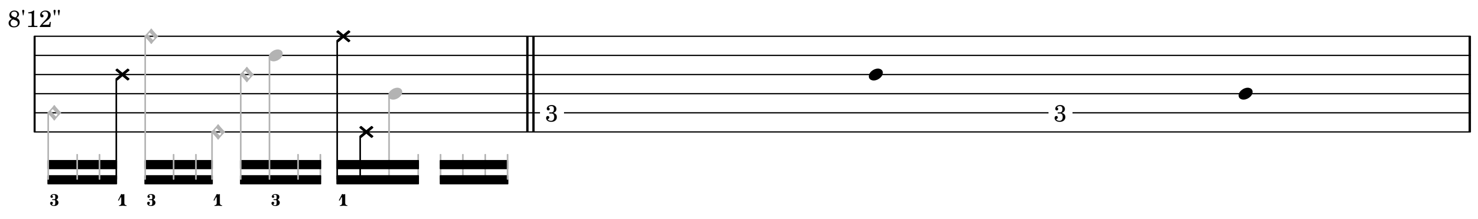


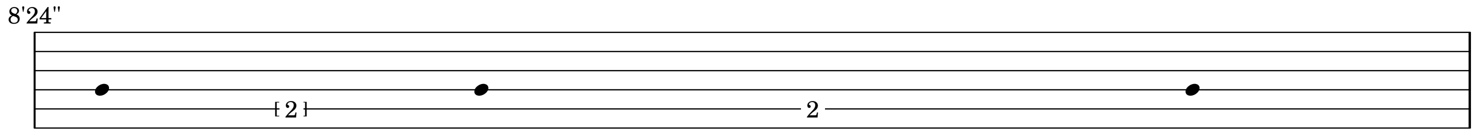
7'48"

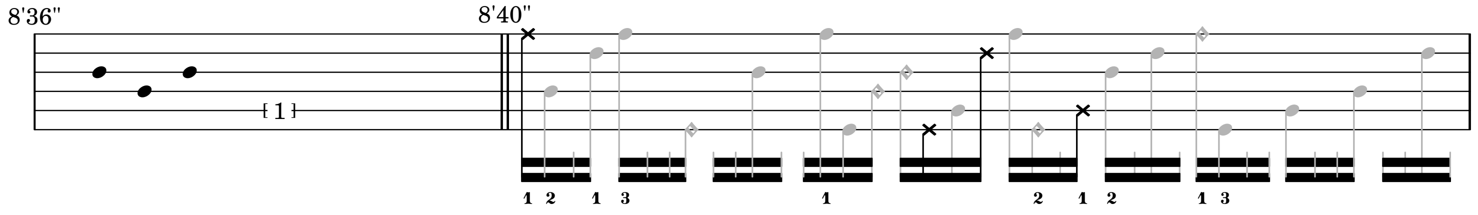


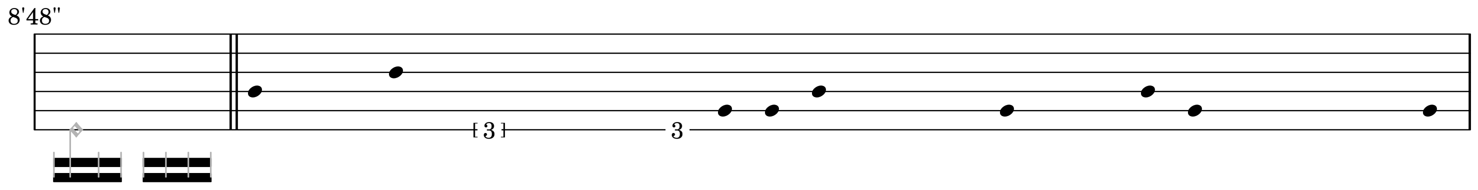
8'00"

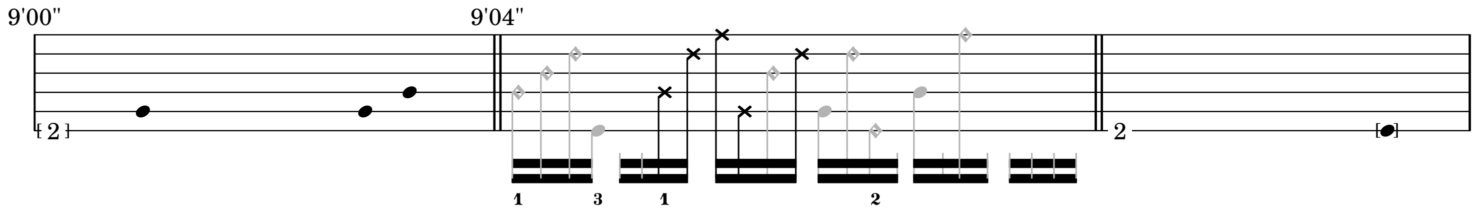


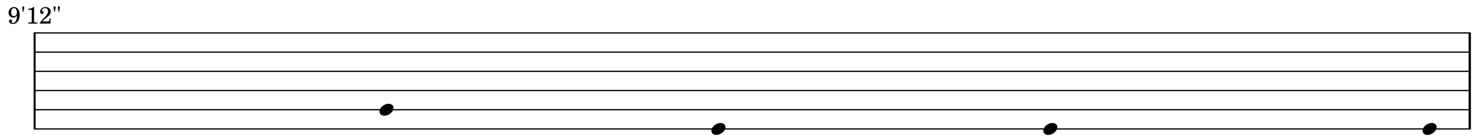

8'12"
3
3
3 1 3 1 3 1

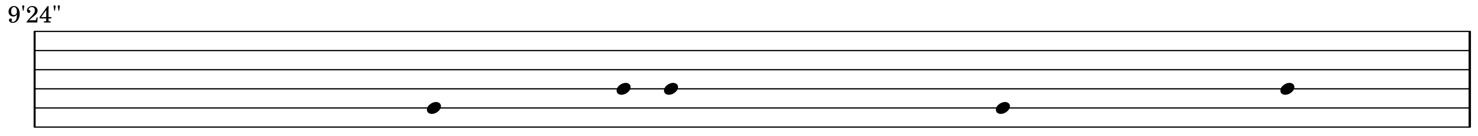

8'24"
{2}
2

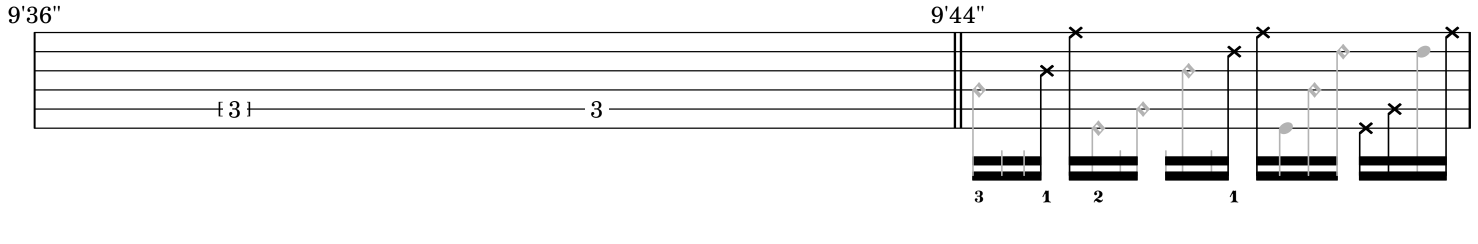

8'36"
{1}
8'40"
1 2 1 3 1 2 1 2 1 3


8'48"
{3}
3

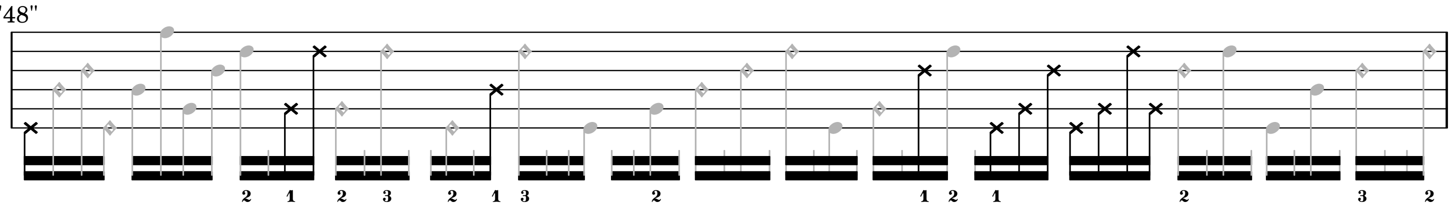

9'00"
{2}
9'04"
1 3 1 2
2


9'12"

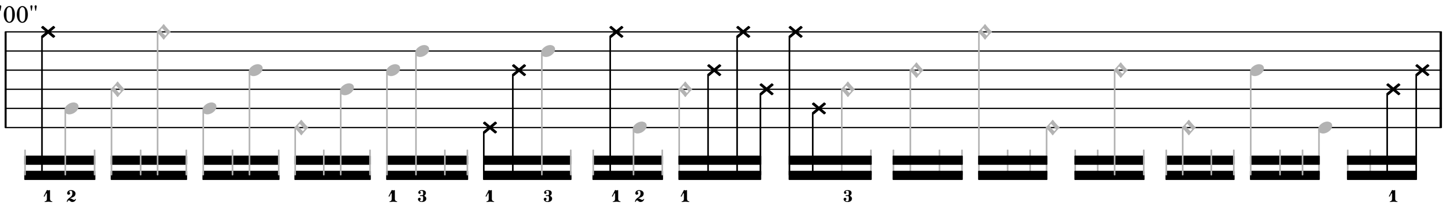

9'24"


9'36"
{3}
3
9'44"
3 1 2 1

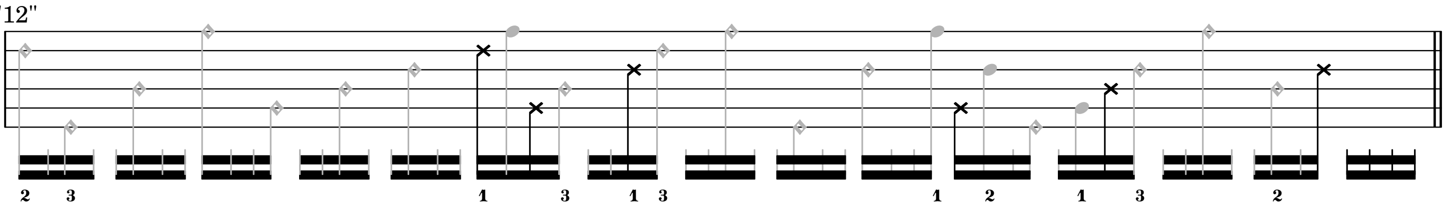
9'48"



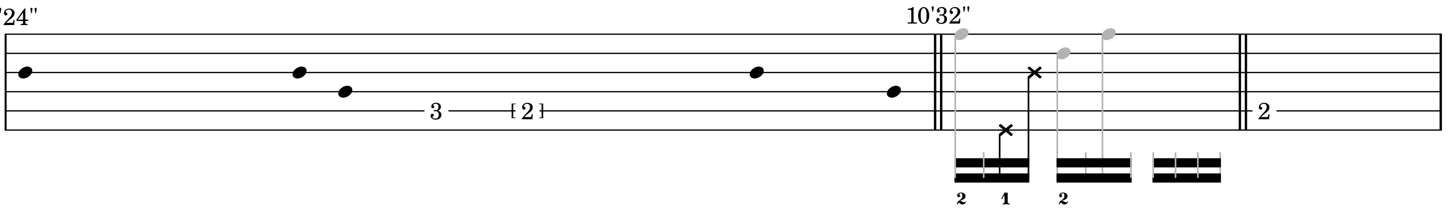
10'00"



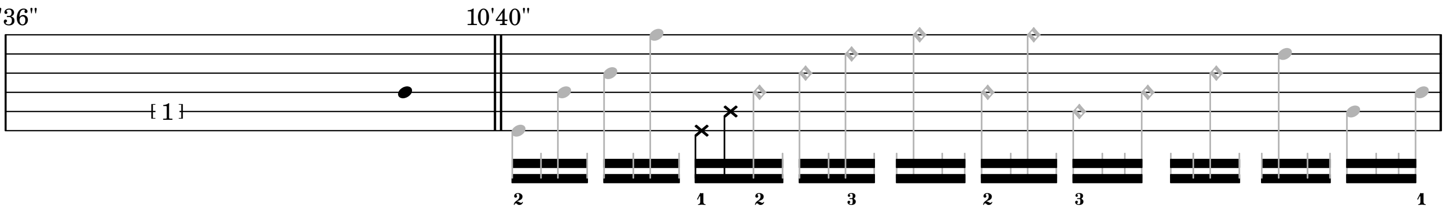
10'12"



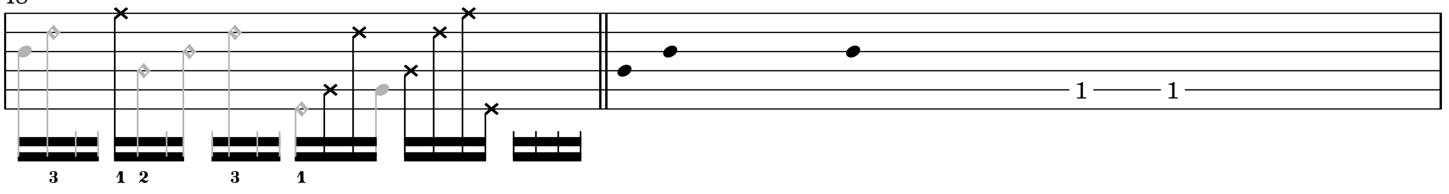
10'24"



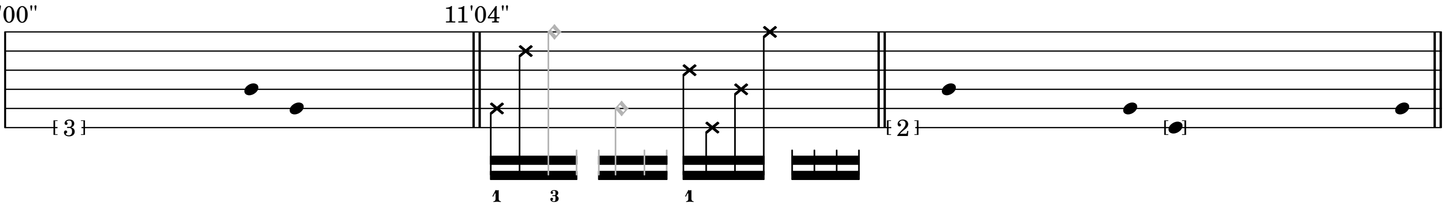
10'36"



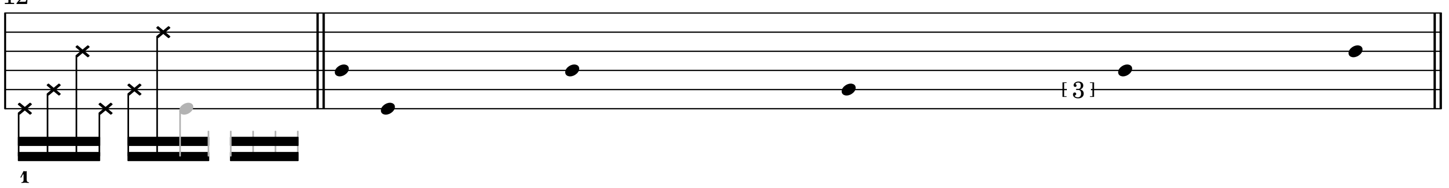
10'48"



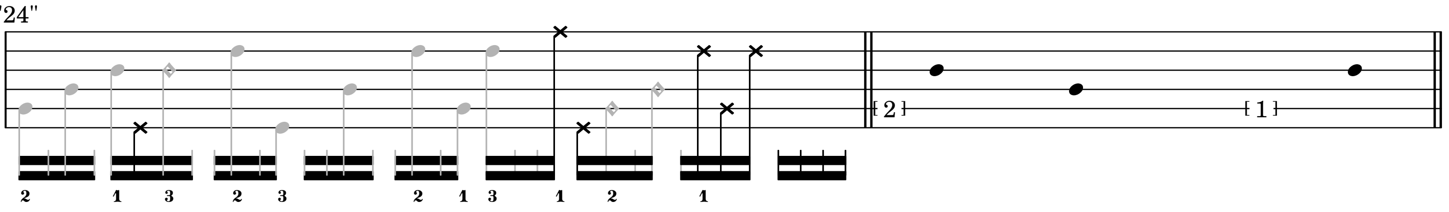
11'00"



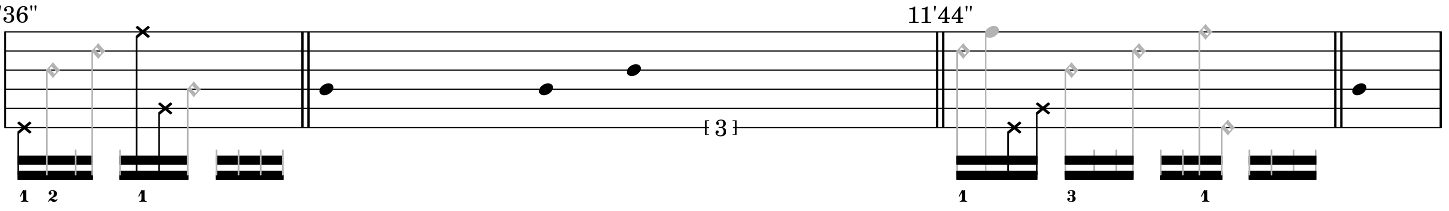
11'12"



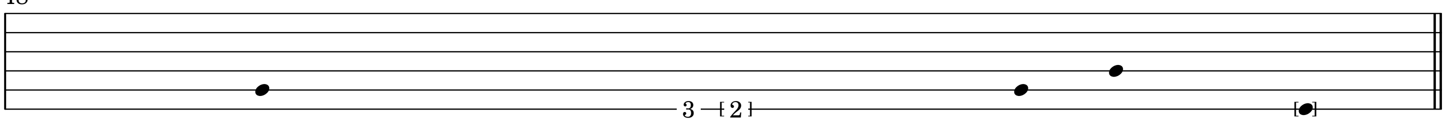
11'24"



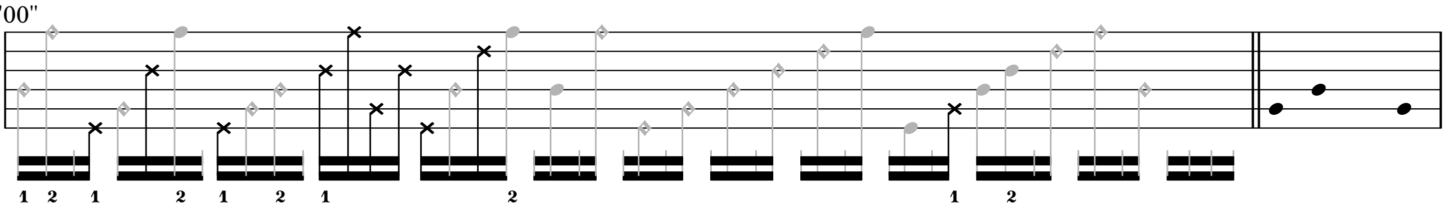
11'36" 11'44"



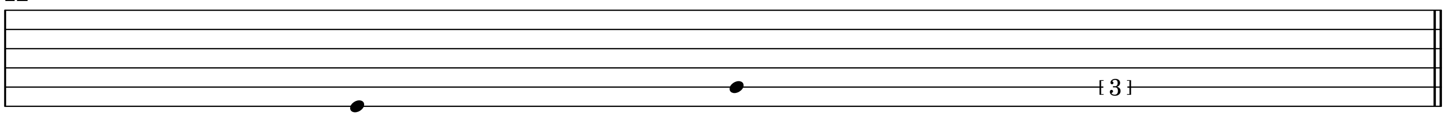
11'48"



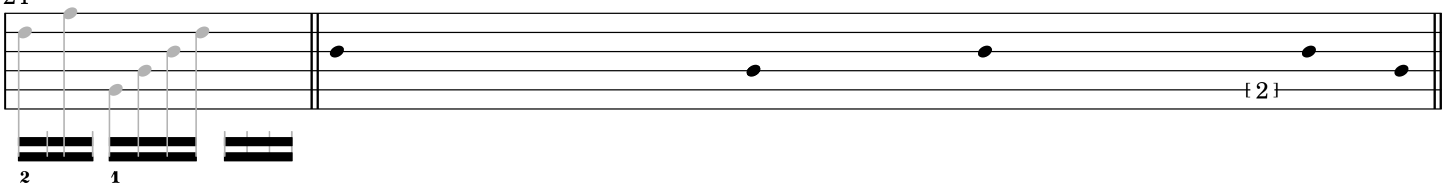
12'00"



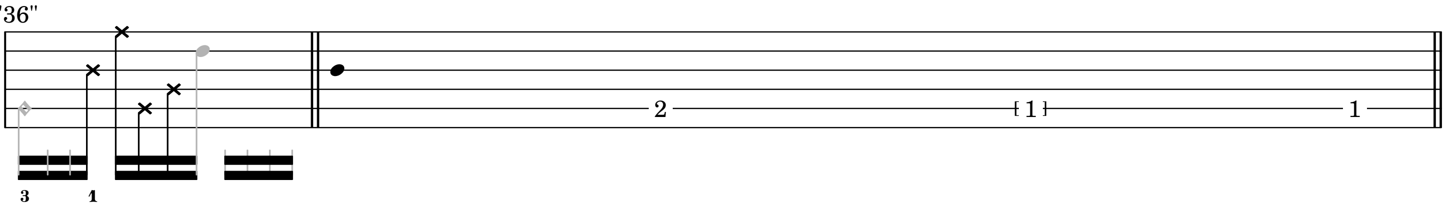
12'12"



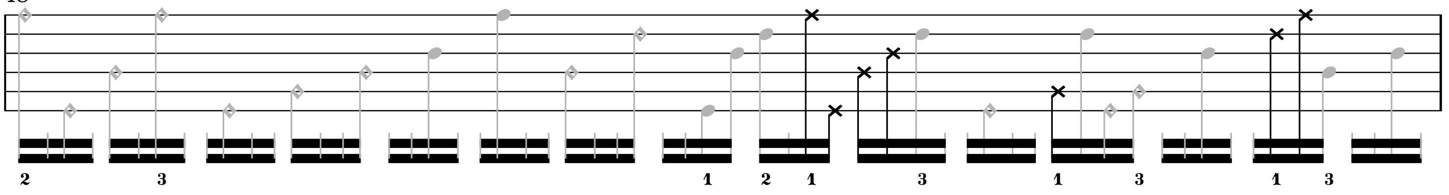
12'24"



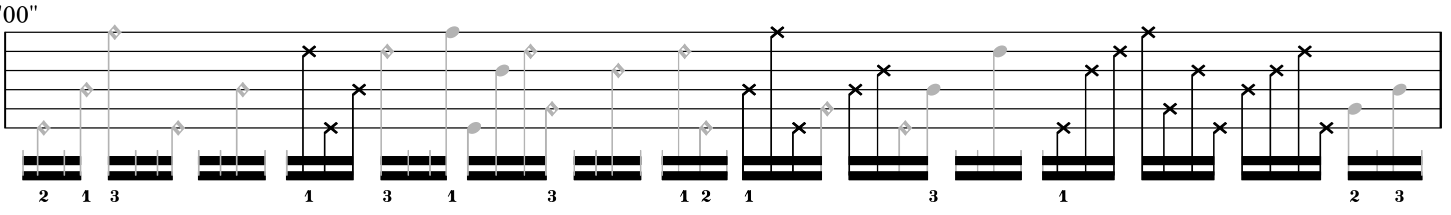
12'36"



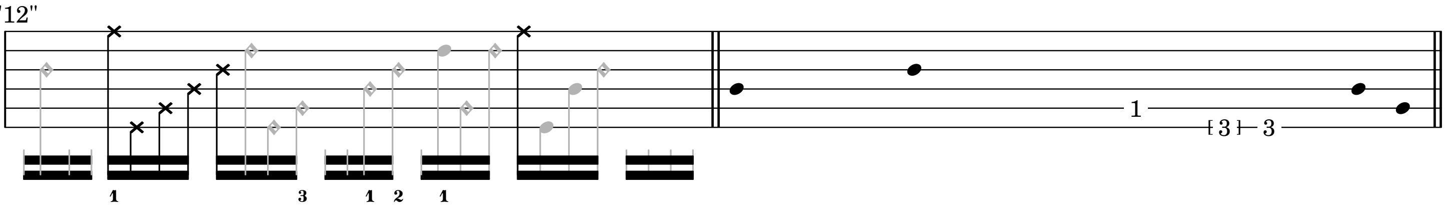
12'48"



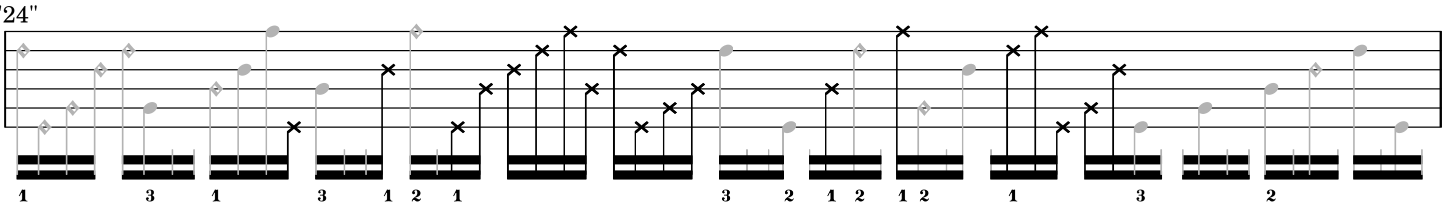
13'00"



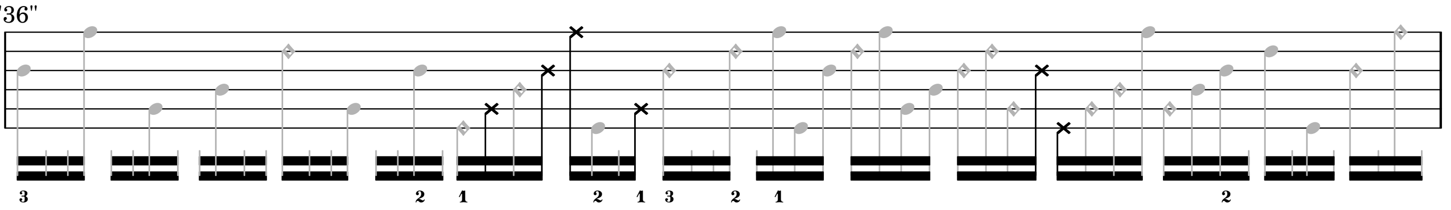
13'12"



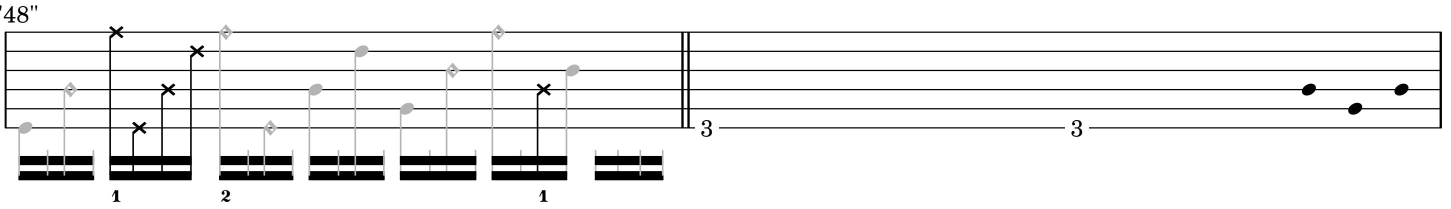
13'24"



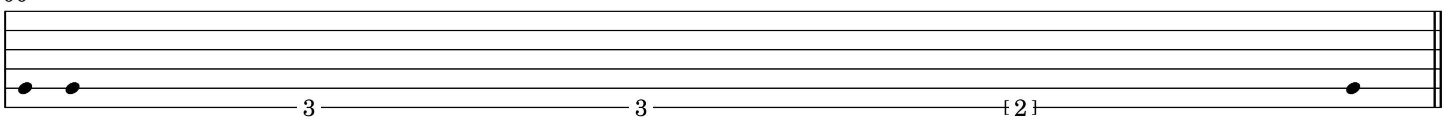
13'36"



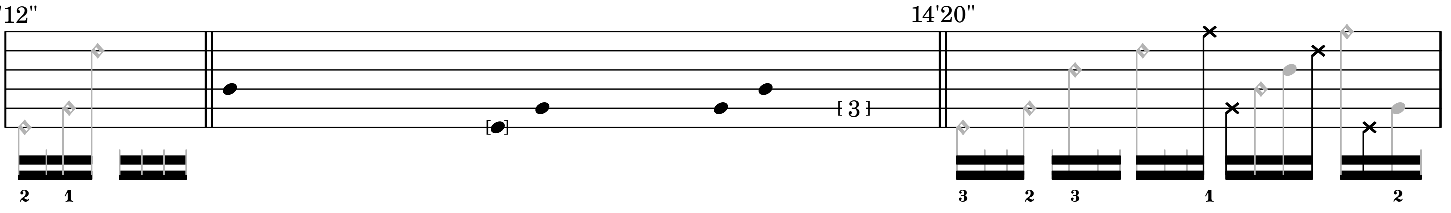
13'48"



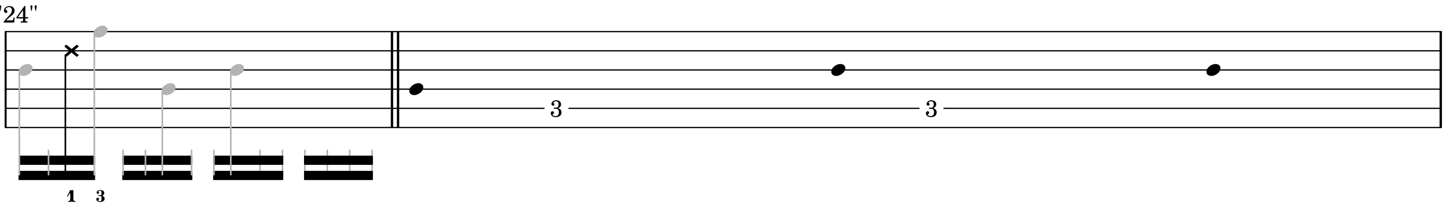
14'00"



14'12" 14'20"

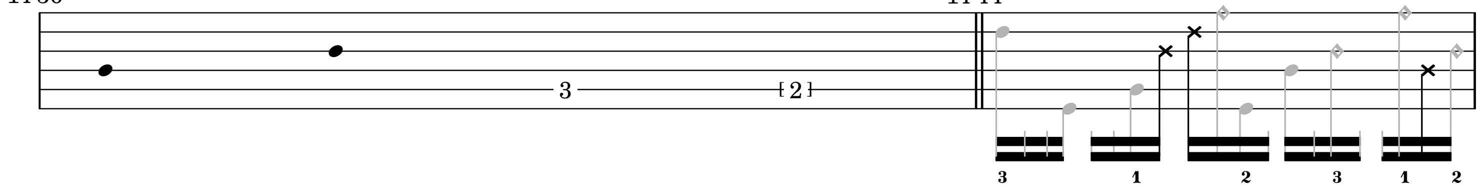


14'24"

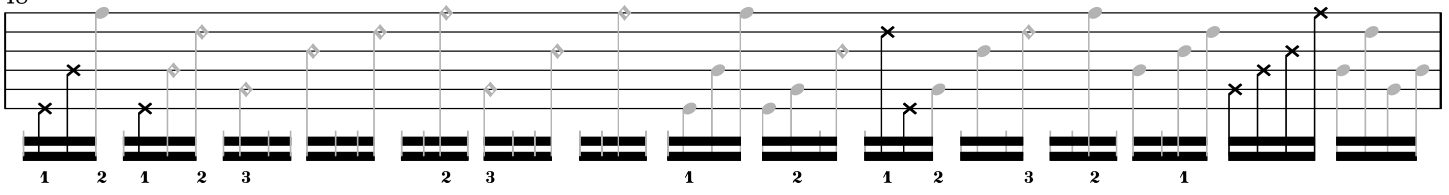


14'36"

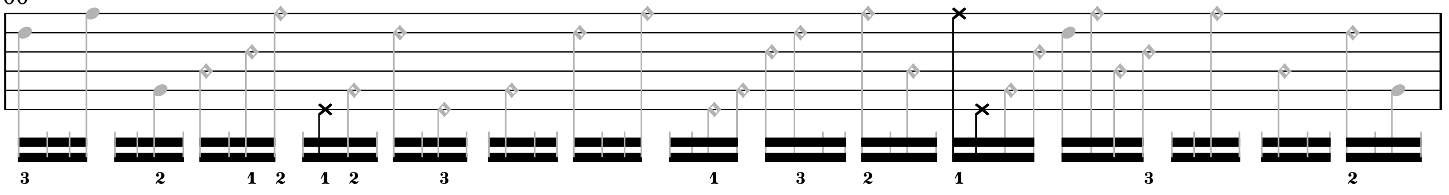
14'44"



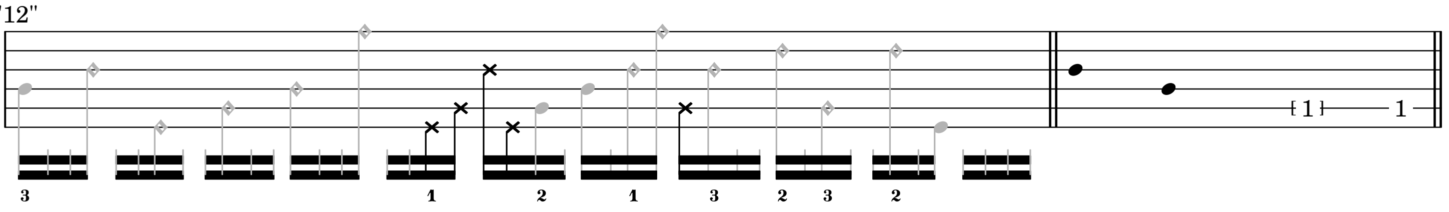
14'48"



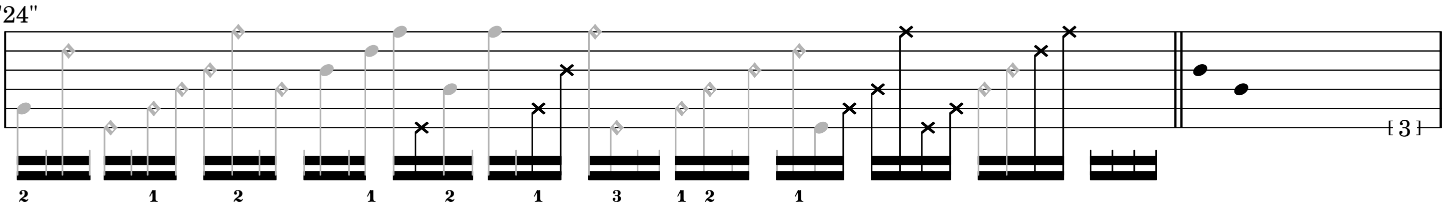
15'00"



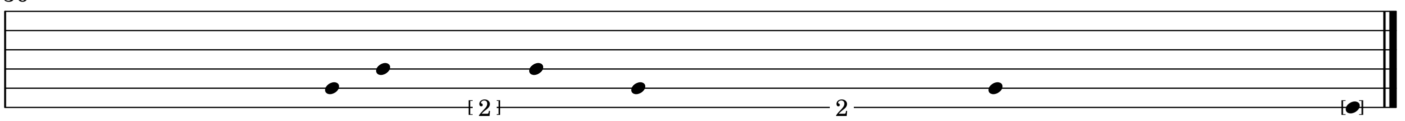
15'12"



15'24"



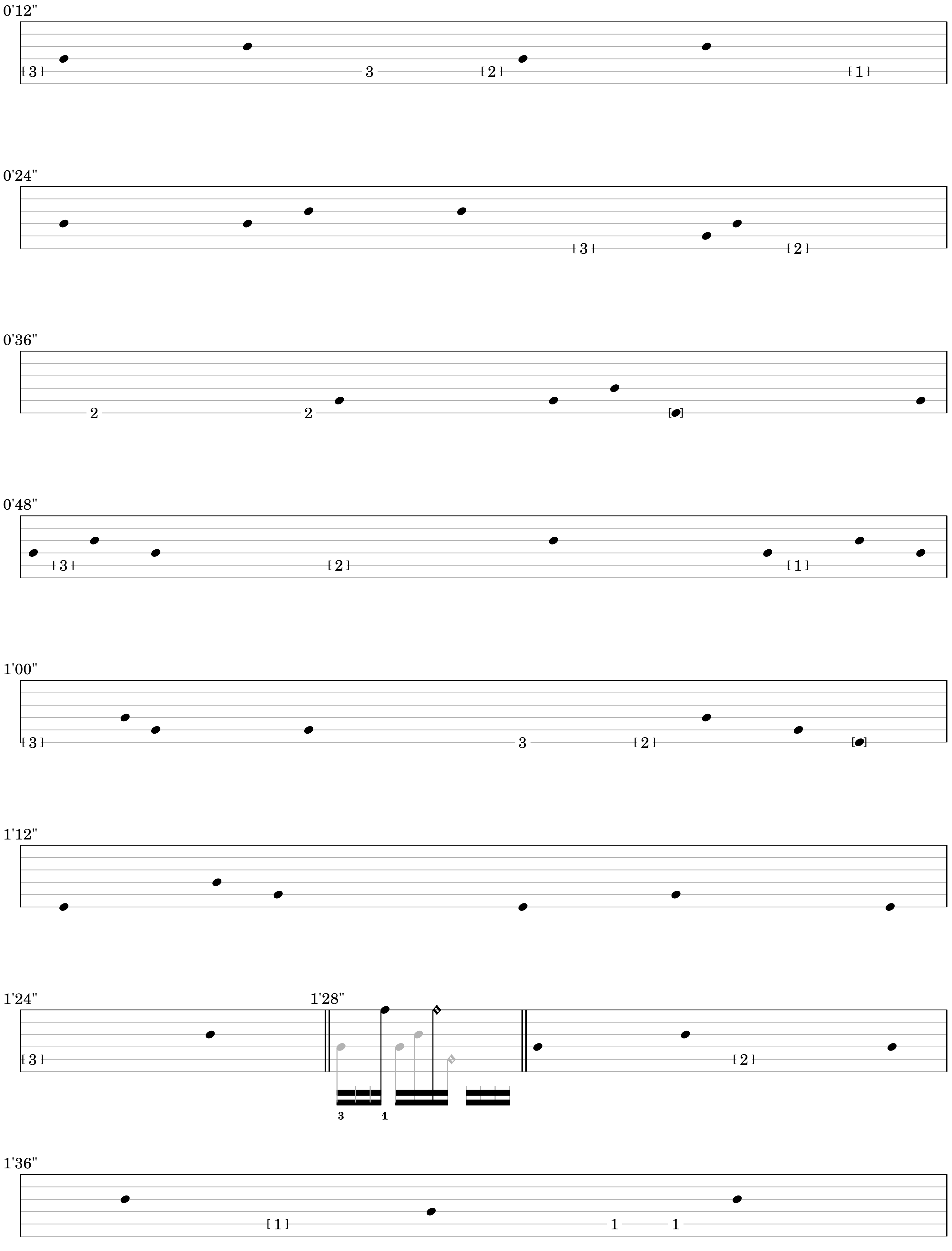
15'36"

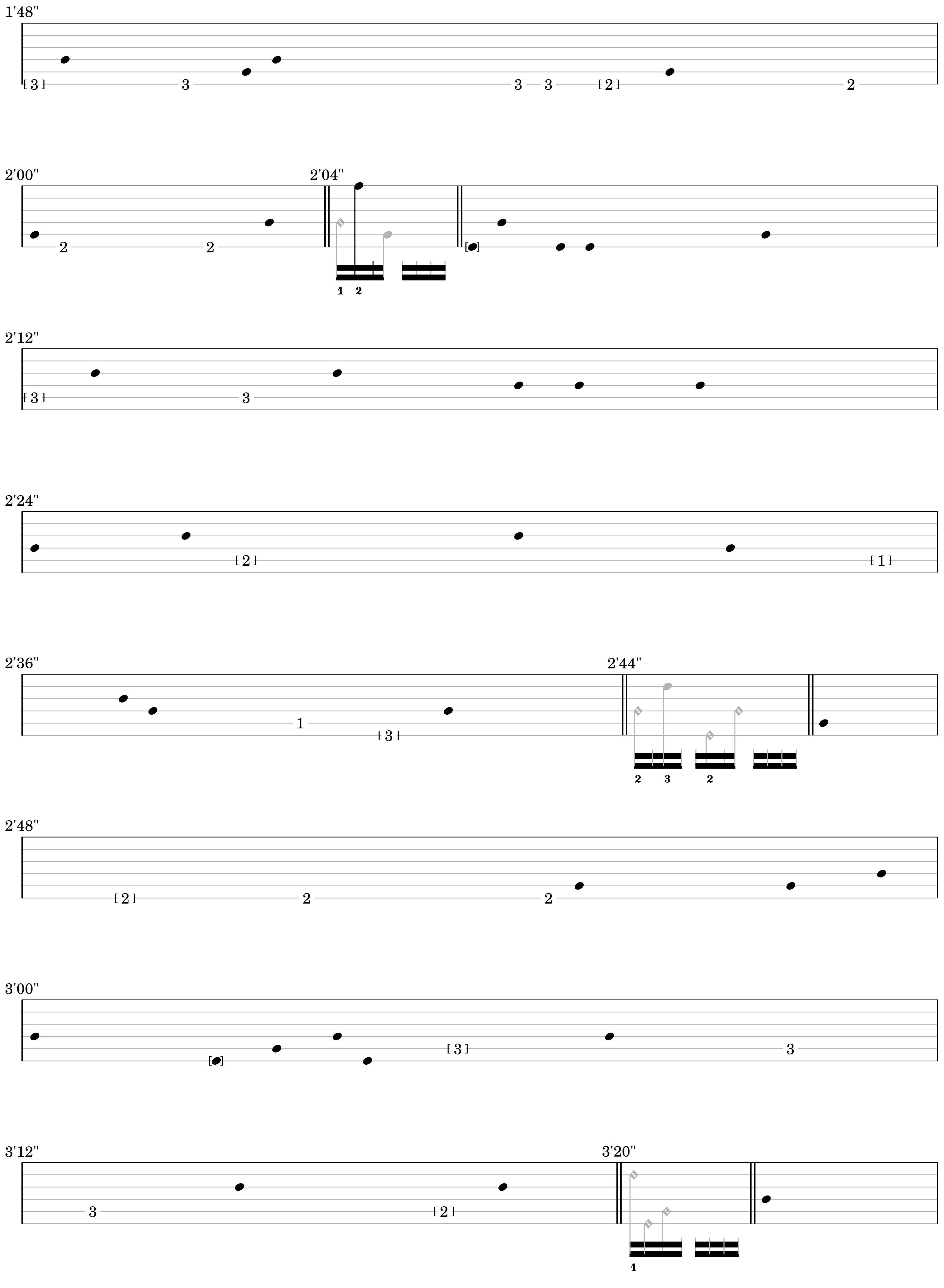


# *ostinato and interrupt*

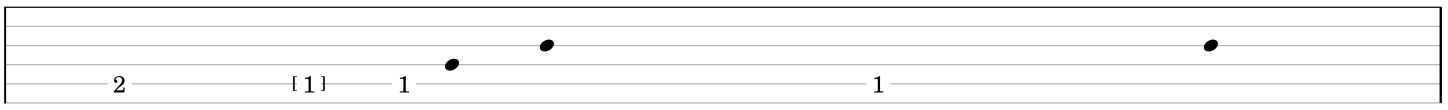
michael winter (mexico city, mx; 2017)
version generated: 2017.12.06

ensemble part 1

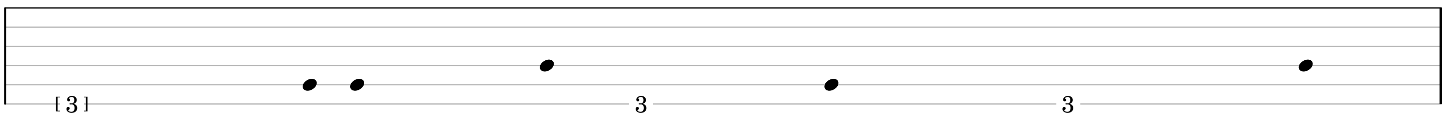



1'48"
[3] 3 3 3 [2] 2
2'00"
2 2
2'04"
1 2
2'12"
[3] 3
2'24"
[2] [1]
2'36"
1 [3]
2'44"
2 3 2
2'48"
[2] 2 2
3'00"
[3] 3
3'12"
3 [2]
3'20"
1

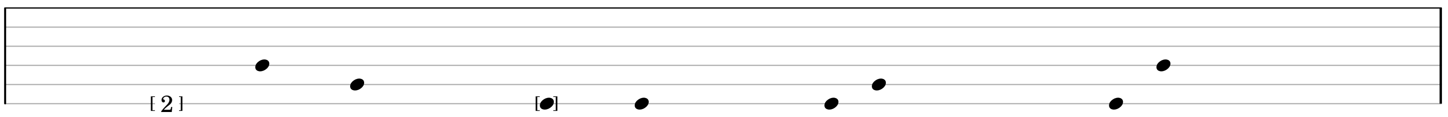
3'24"



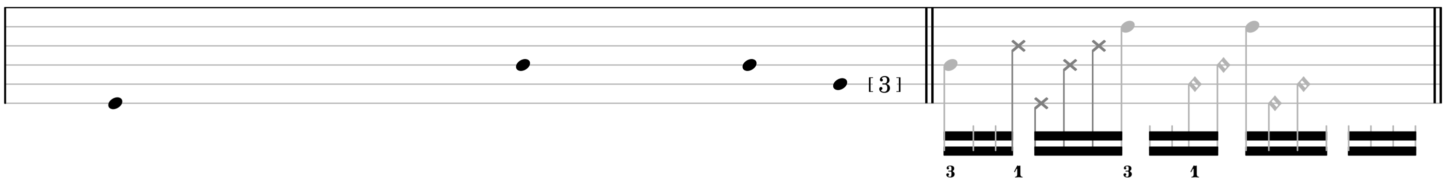
3'36"



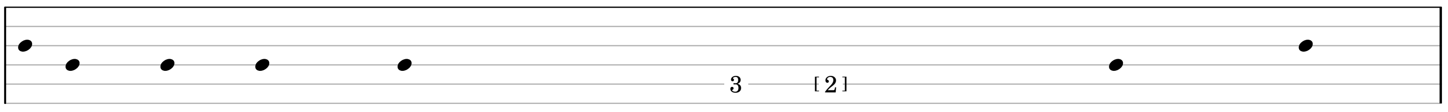
3'48"



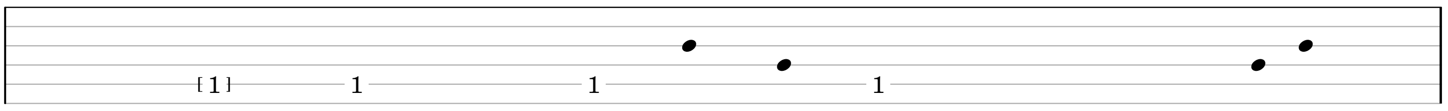
4'00"    4'08"



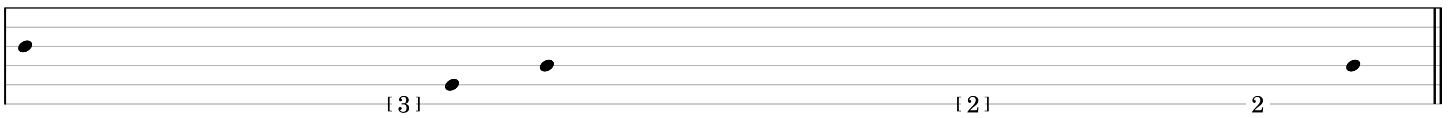
4'12"



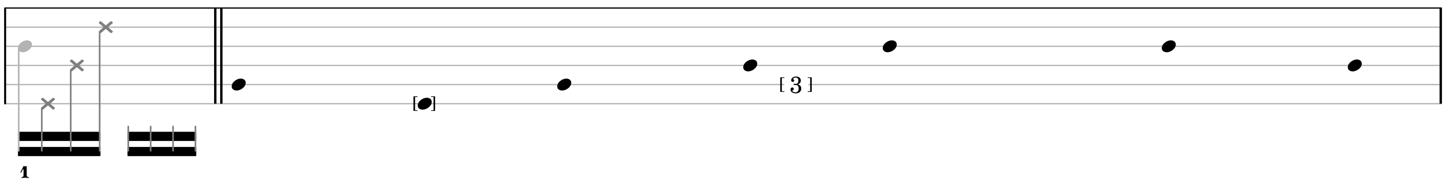
4'24"

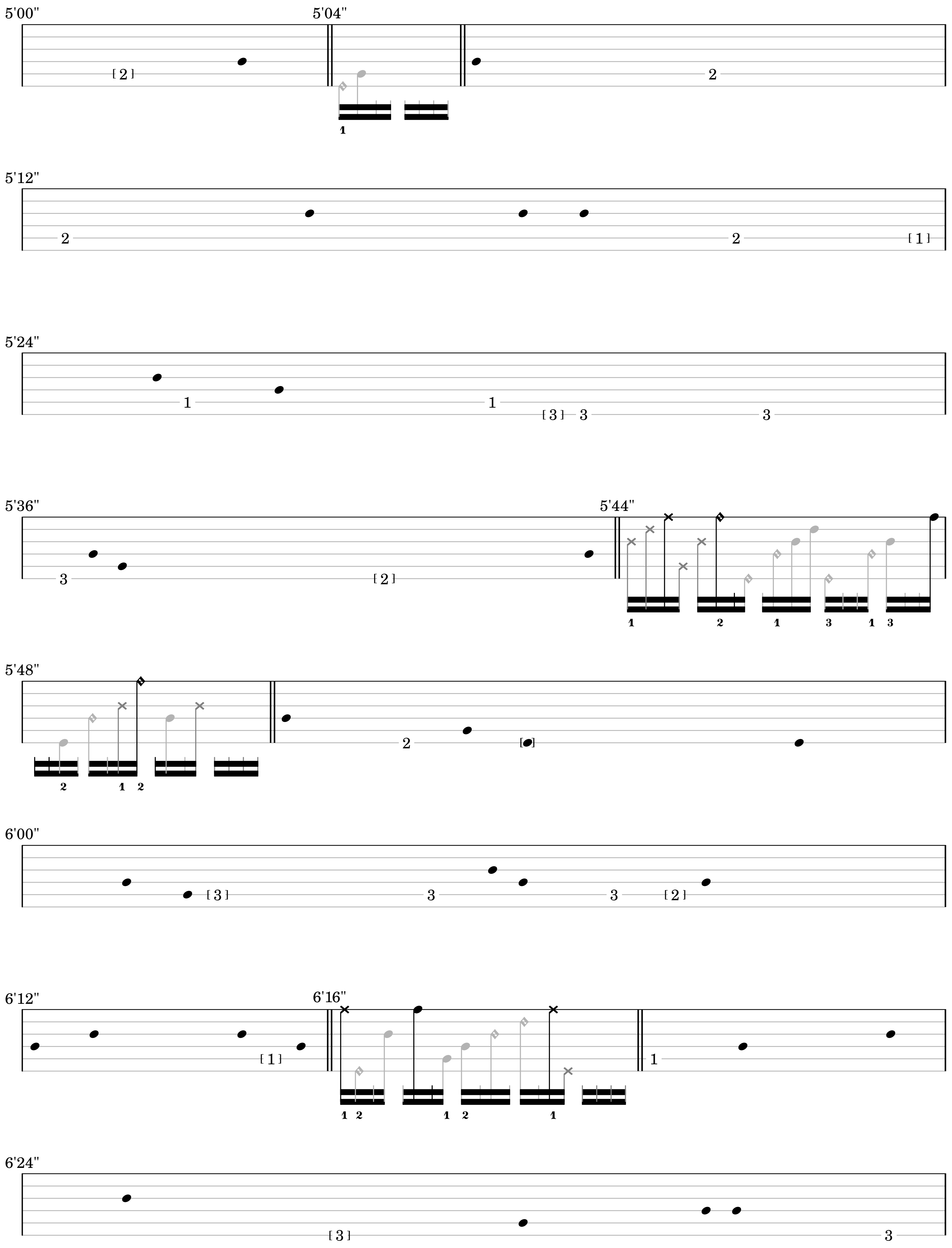


4'36"

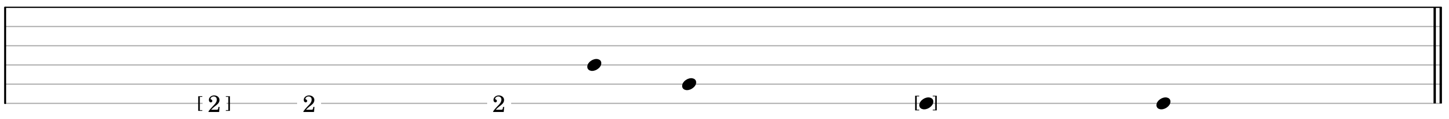


4'48"

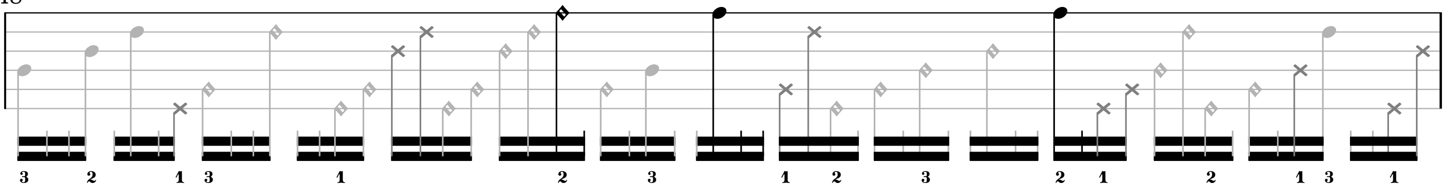



5'00"
5'04"
5'12"
5'24"
5'36"
5'44"
5'48"
6'00"
6'12"
6'16"
6'24"

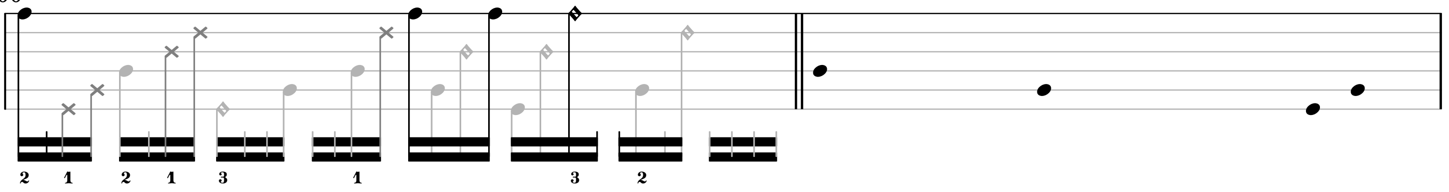
6'36"



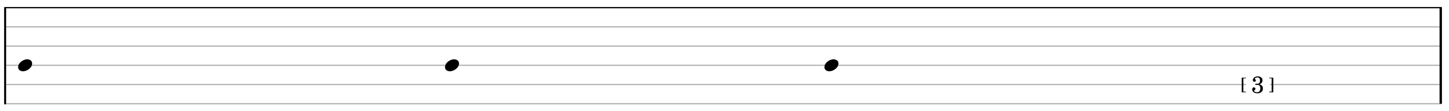
6'48"



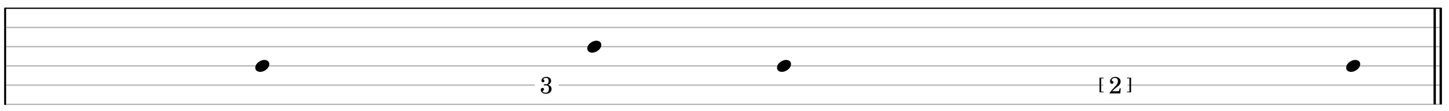
7'00"



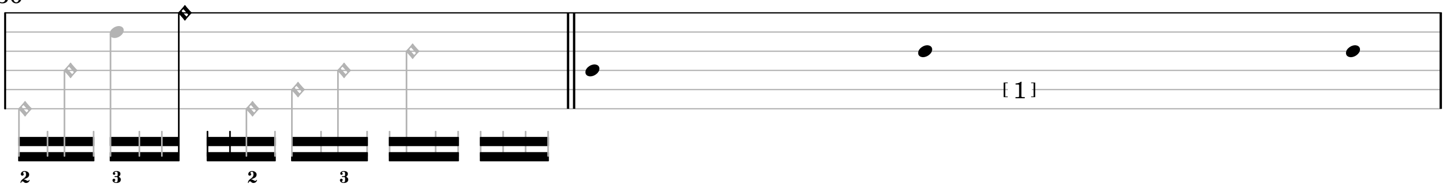
7'12"



7'24"

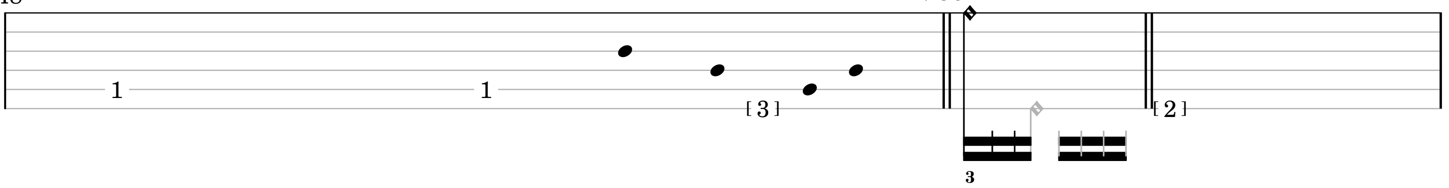


7'36"

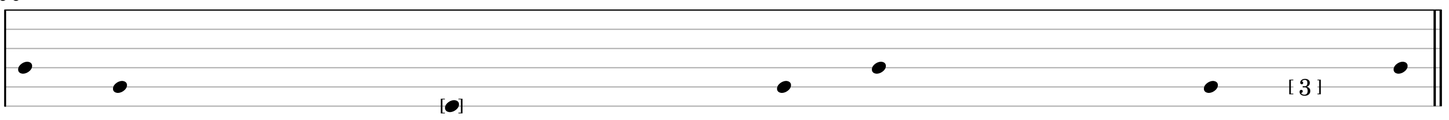


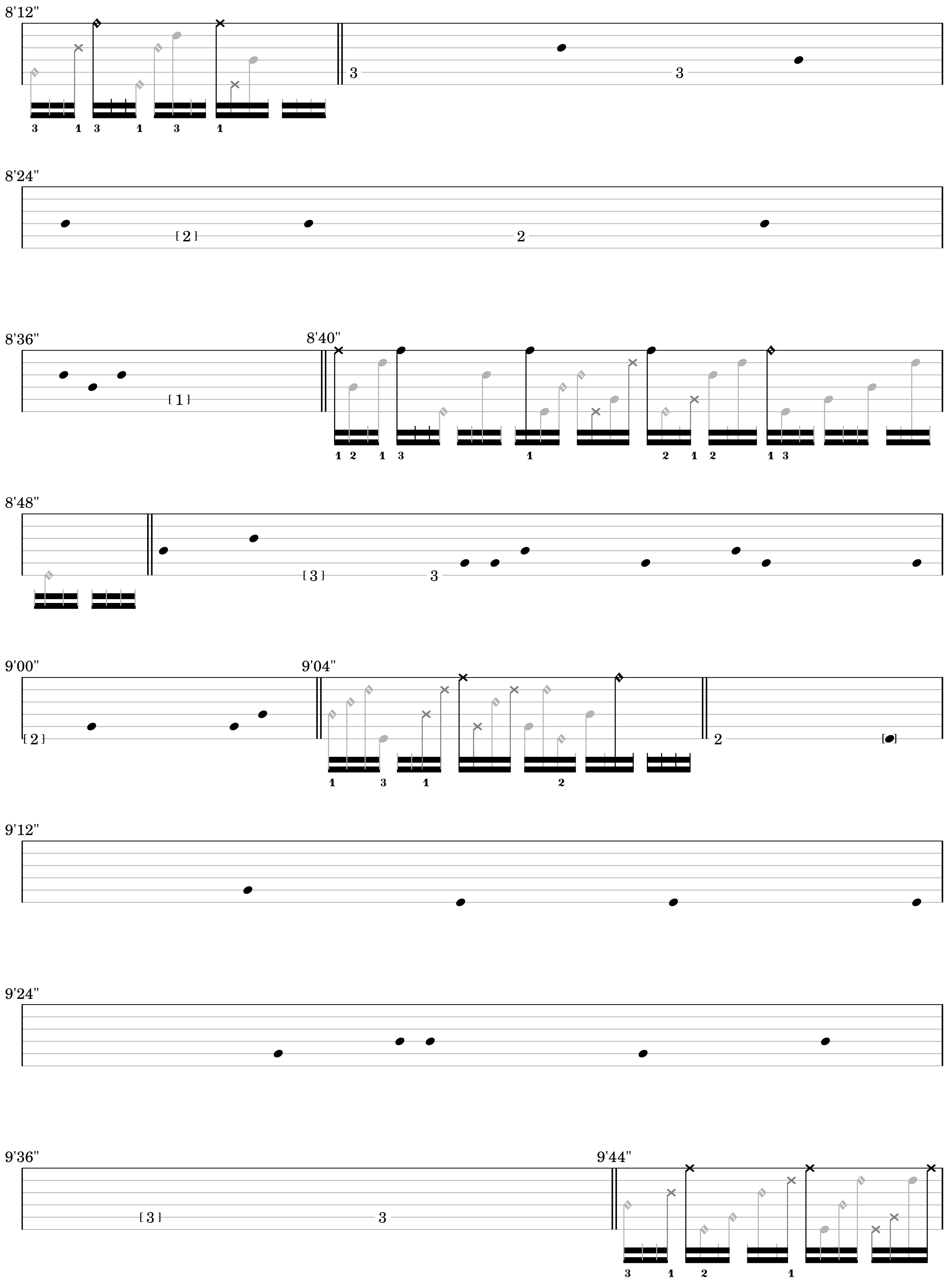
7'48"

7'56"

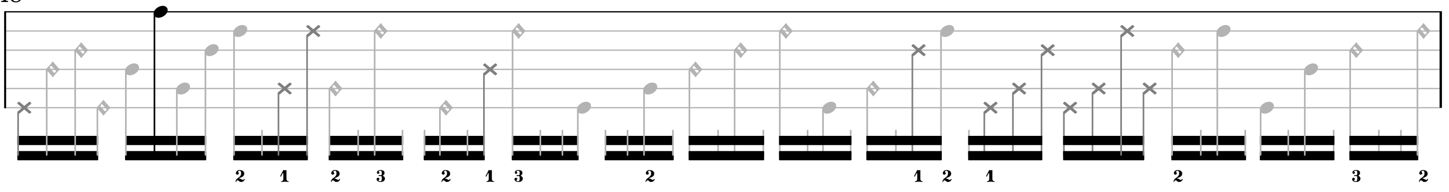


8'00"

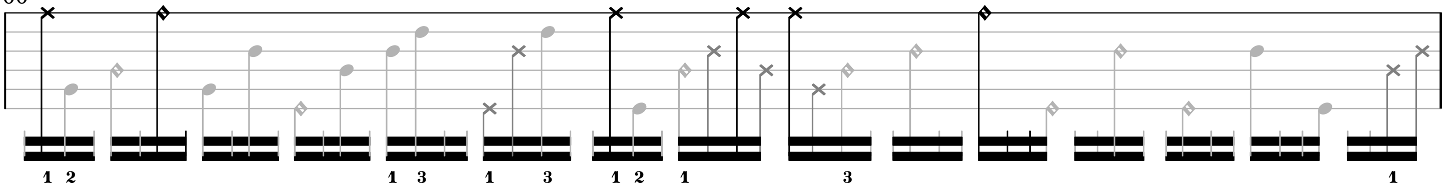



8'12"
3
3
8'24"
[2]
2
8'36"
[1]
8'40"
8'48"
[3]
3
9'00"
[2]
9'04"
2
9'12"
9'24"
9'36"
[3]
3
9'44"

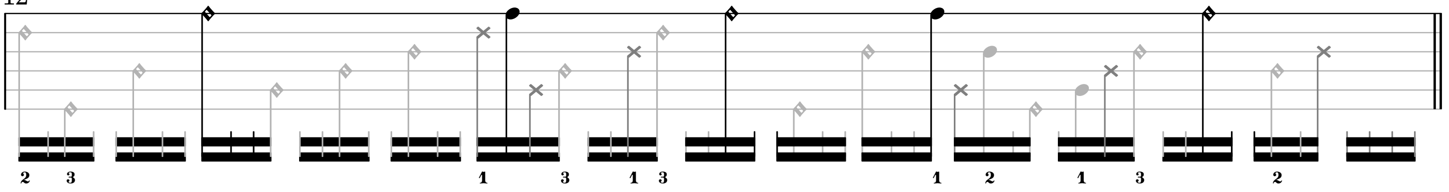
9'48"



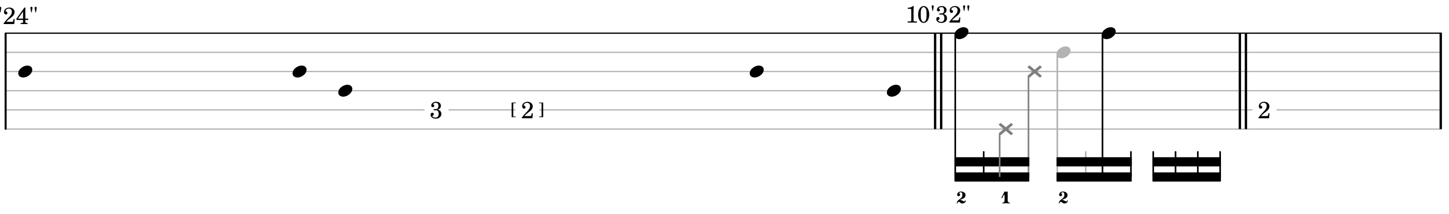
10'00"



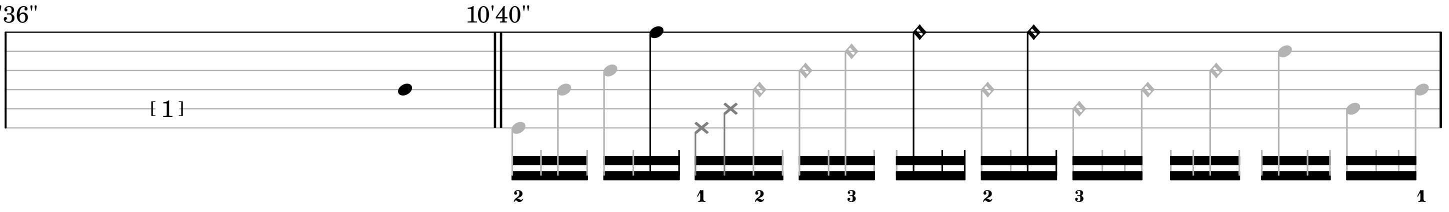
10'12"



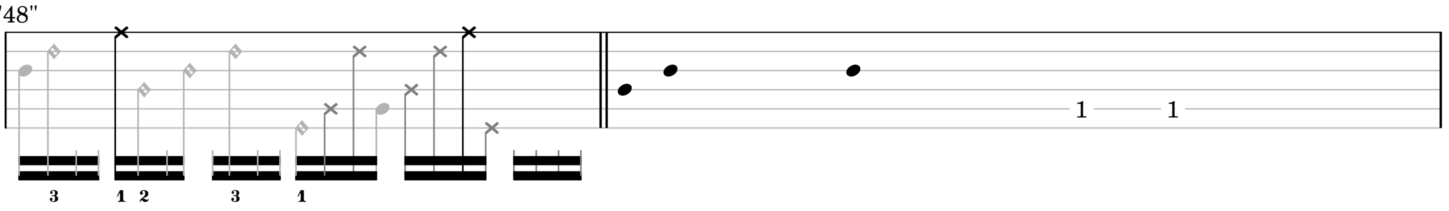
10'24"



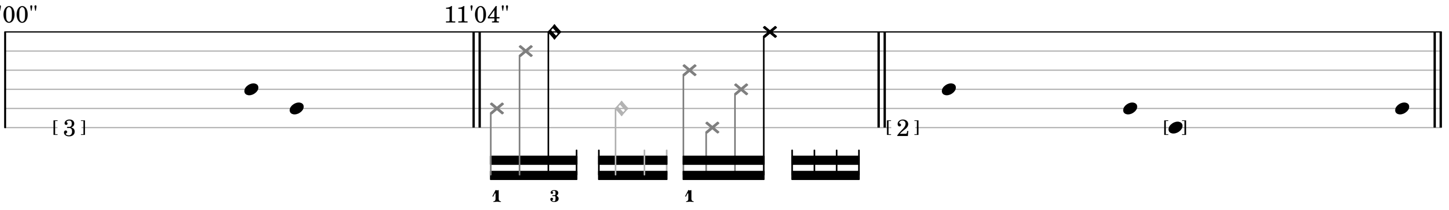
10'36"



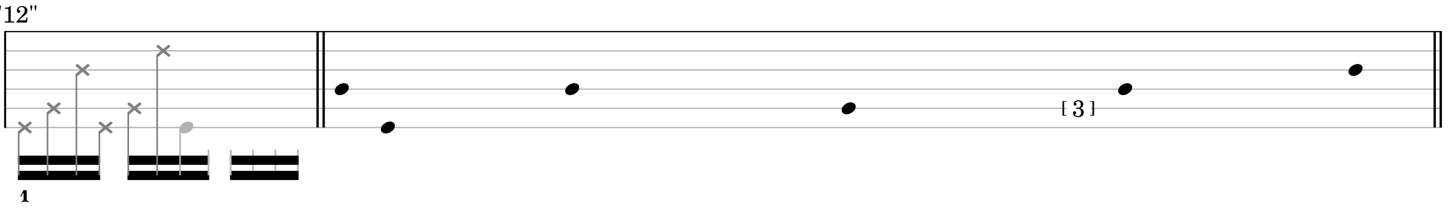
10'48"



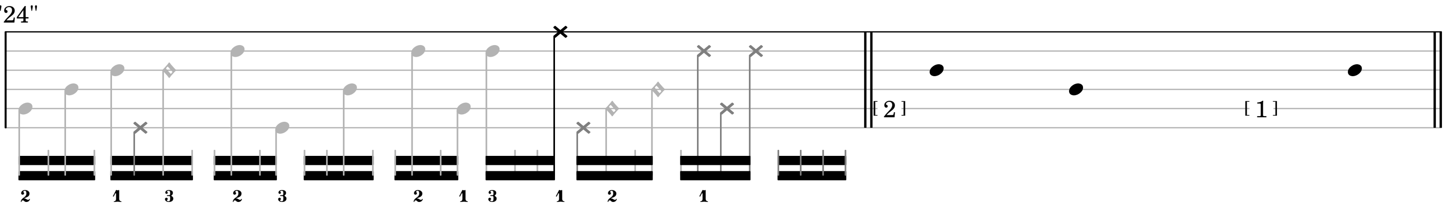
11'00"



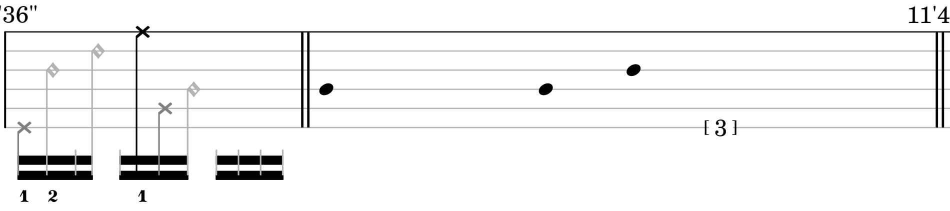
11'12"



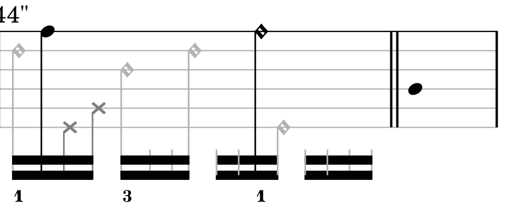
11'24"



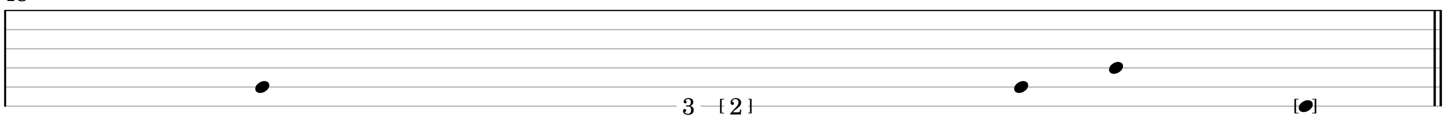
11'36"



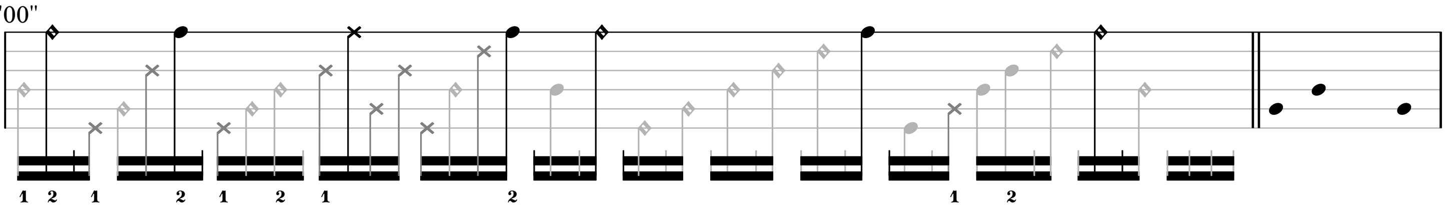
11'44"



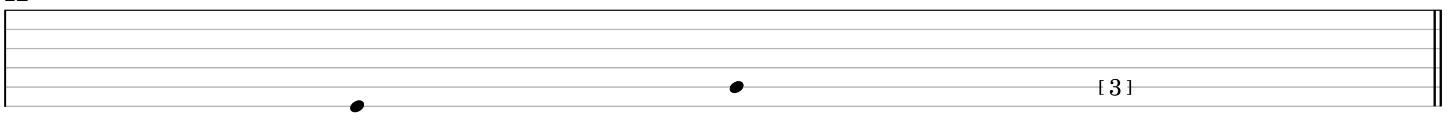
11'48"



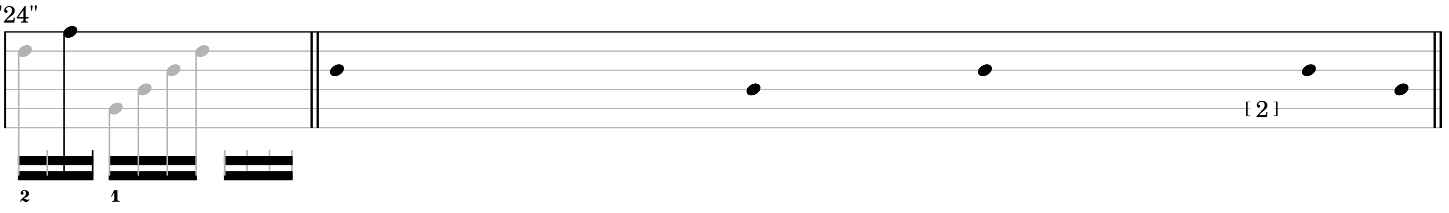
12'00"



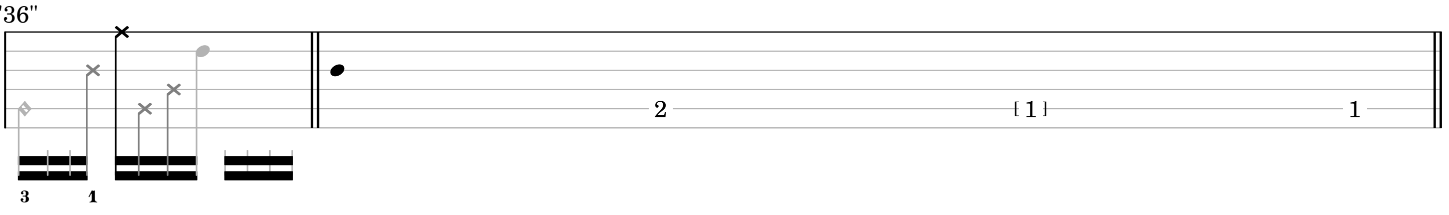
12'12"



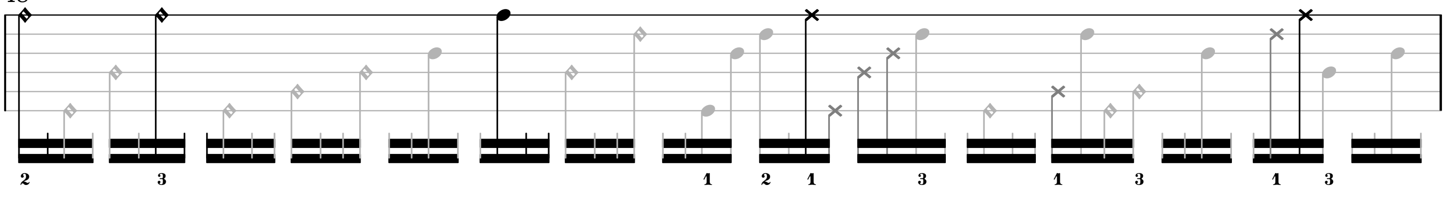
12'24"



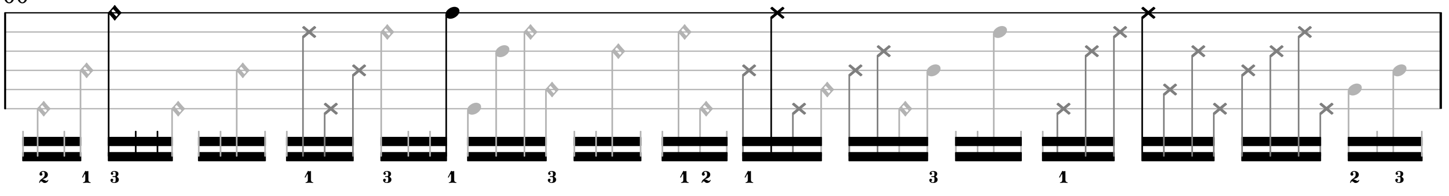
12'36"



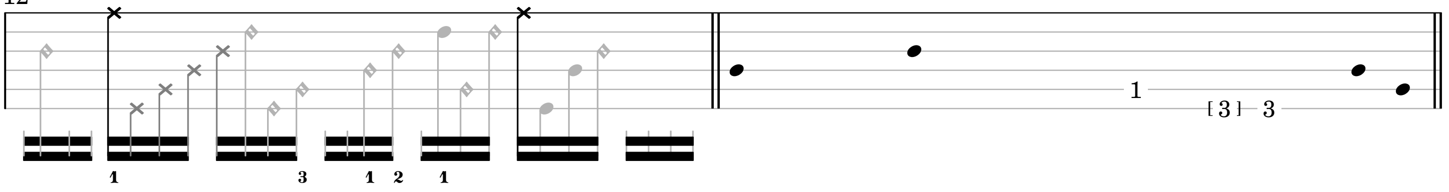
12'48"



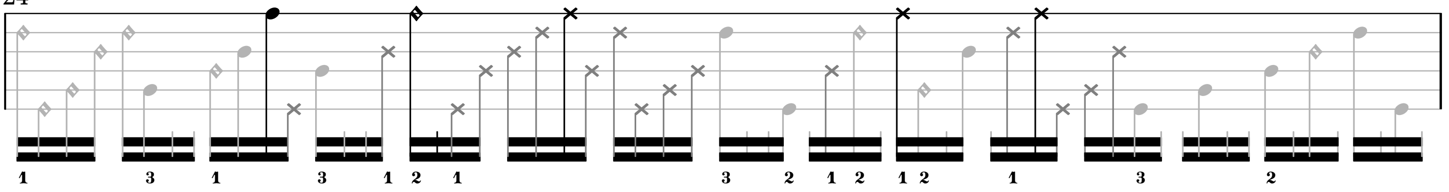
13'00"



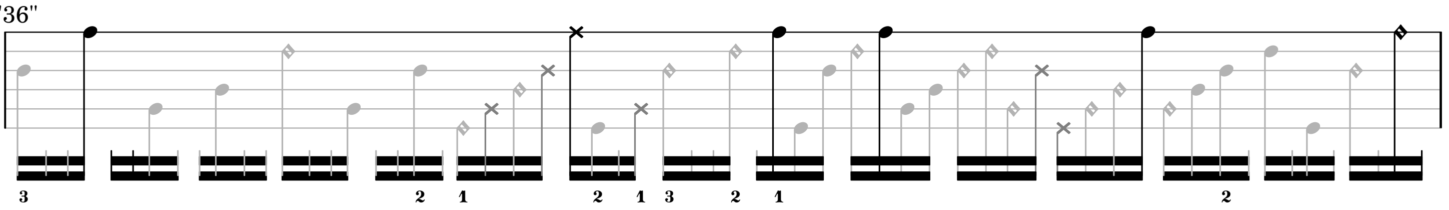
13'12"



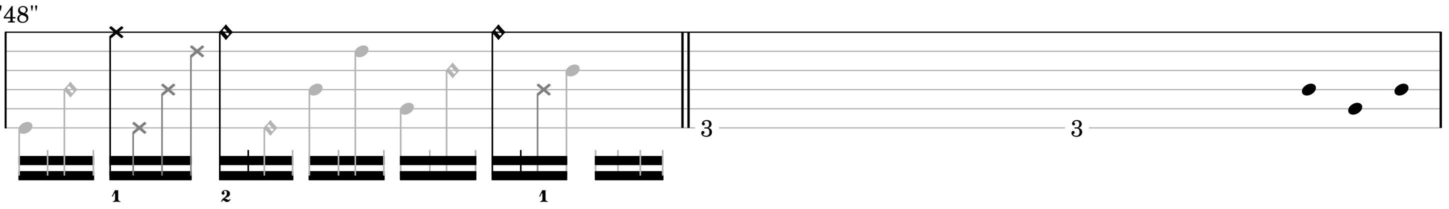
13'24"



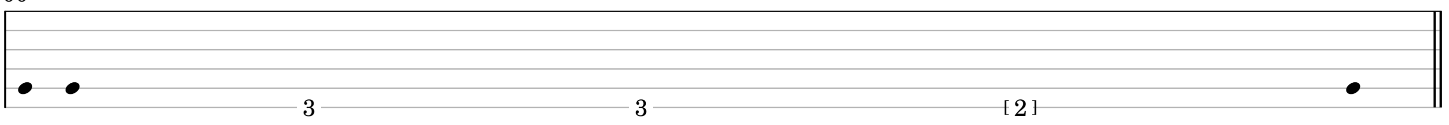
13'36"



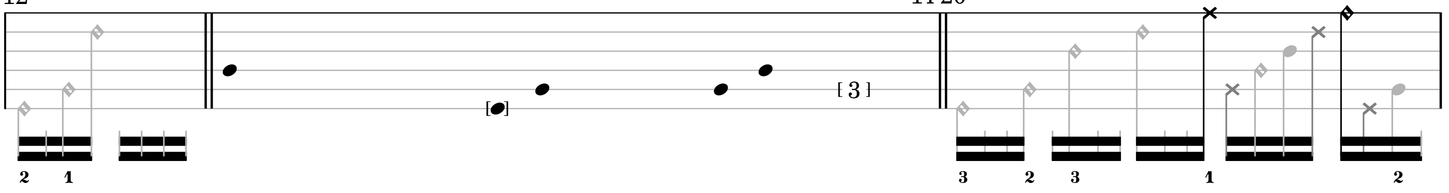
13'48"



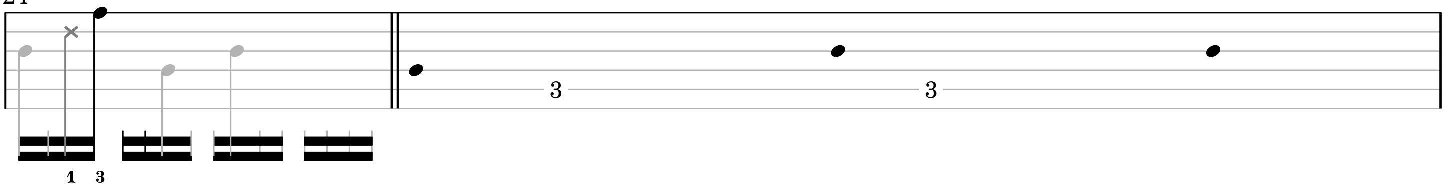
14'00"



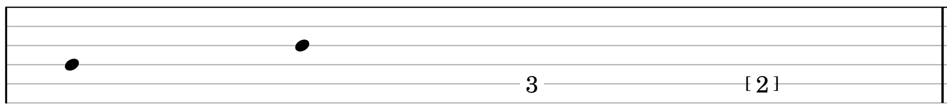
14'12" 14'20"



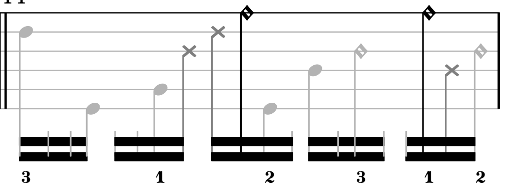
14'24"



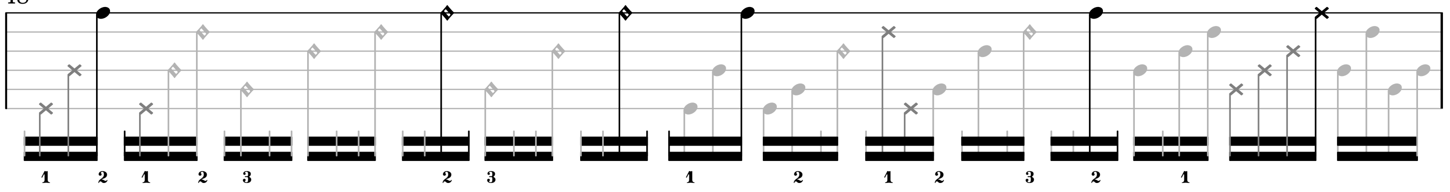
14'36"



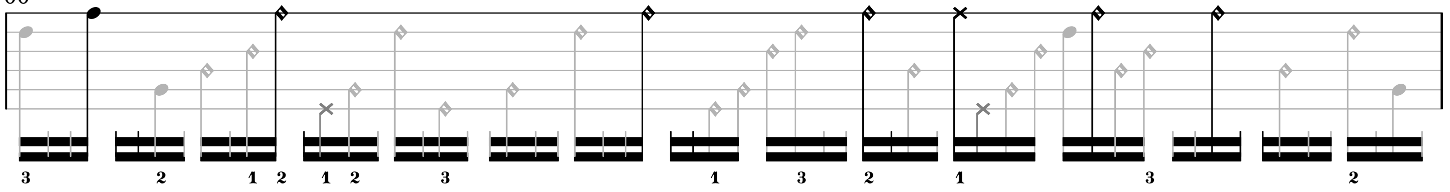
14'44"



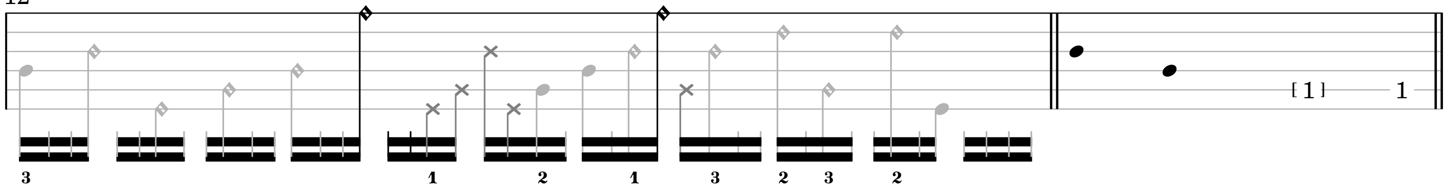
14'48"



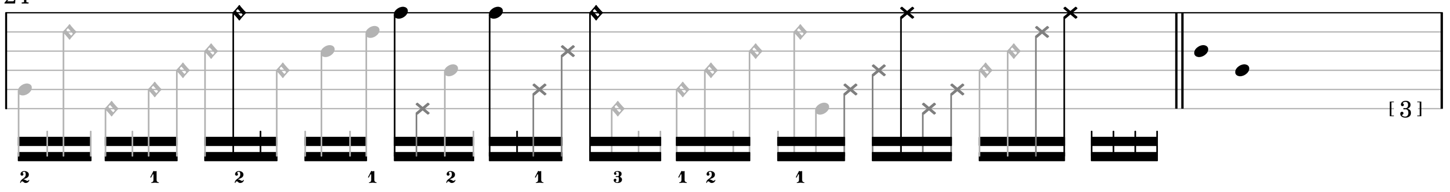
15'00"



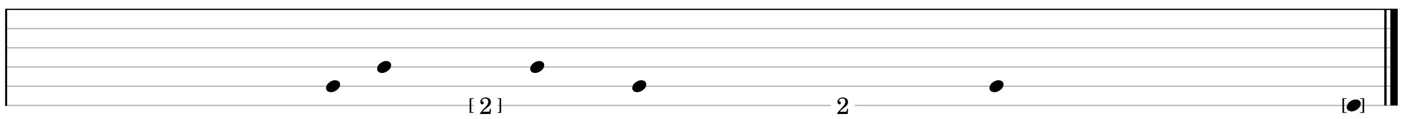
15'12"



15'24"



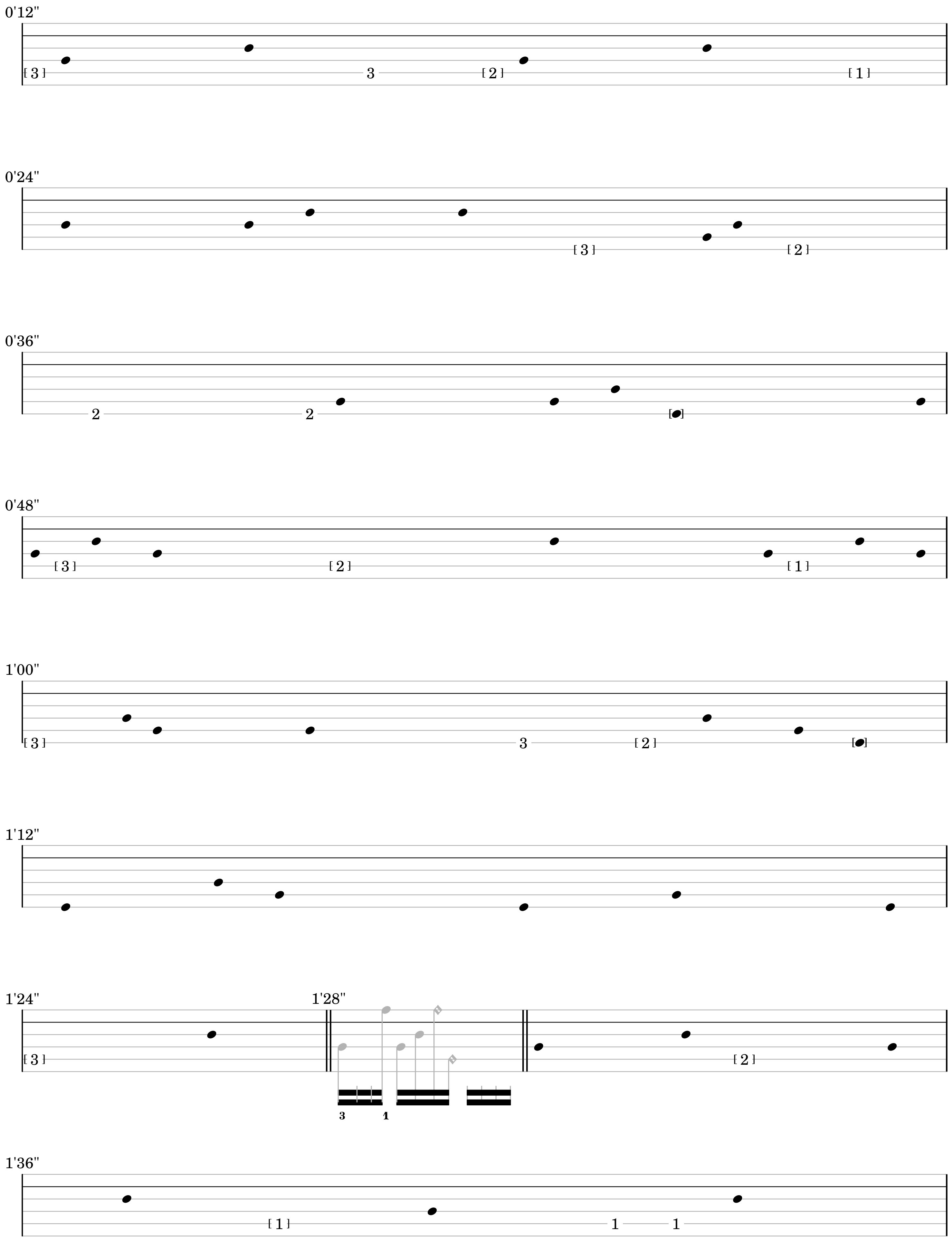
15'36"

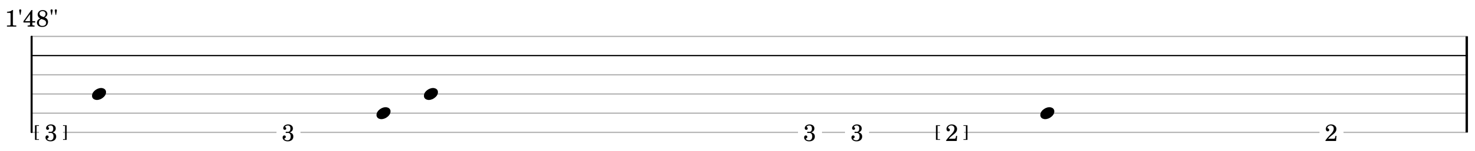


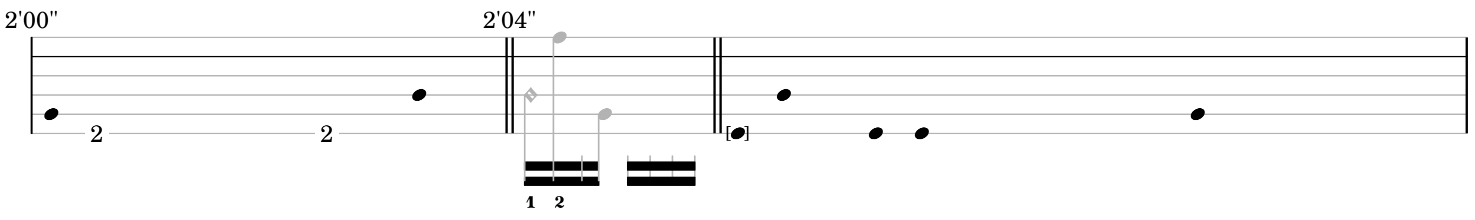
# *ostinato and interrupt*

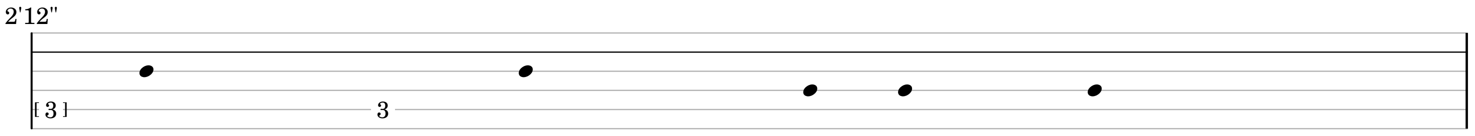
michael winter (mexico city, mx; 2017)
version generated: 2017.12.06

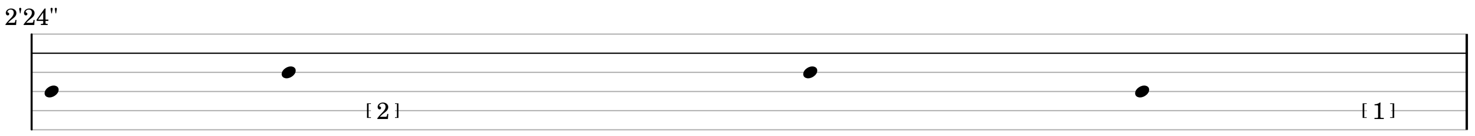
ensemble part 2

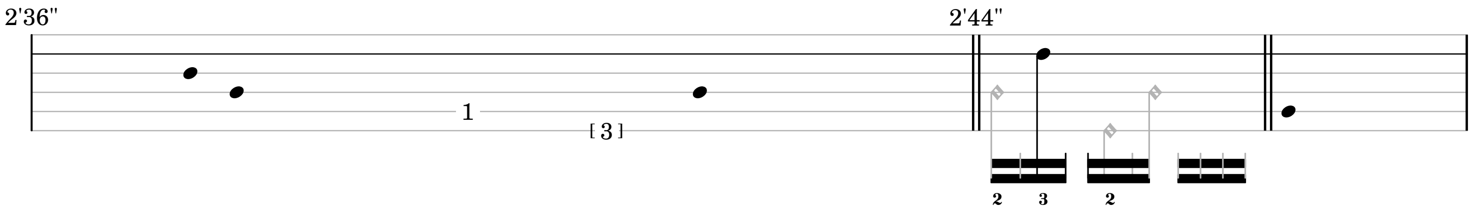


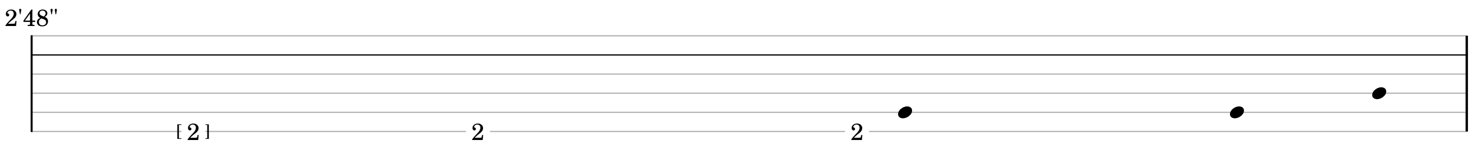

1'48"
[3]
3
3
3
[2]
2

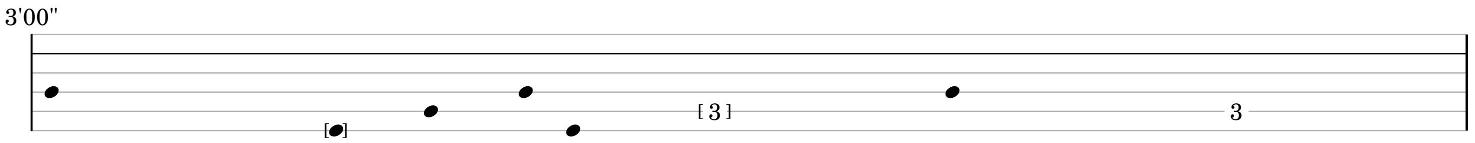

2'00"
2
2
2'04"
1
2

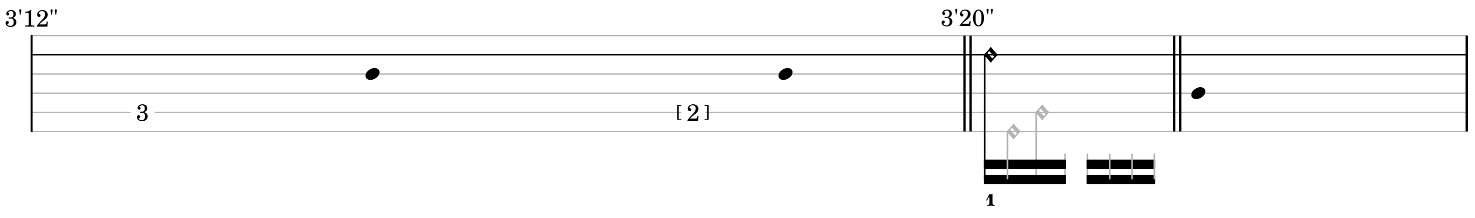

2'12"
[3]
3


2'24"
[2]
[1]

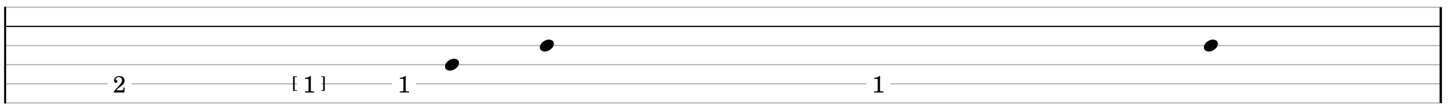

2'36"
1
[3]
2'44"
2
3
2


2'48"
[2]
2
2

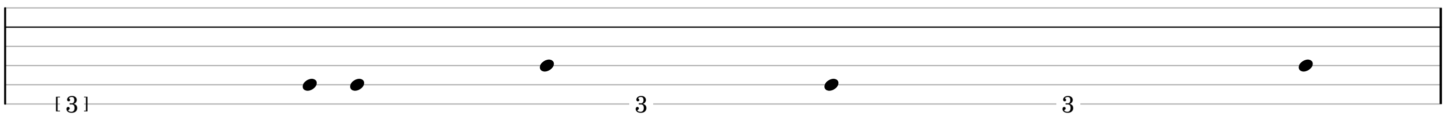

3'00"
[3]
3


3'12"
3
[2]
3'20"
1

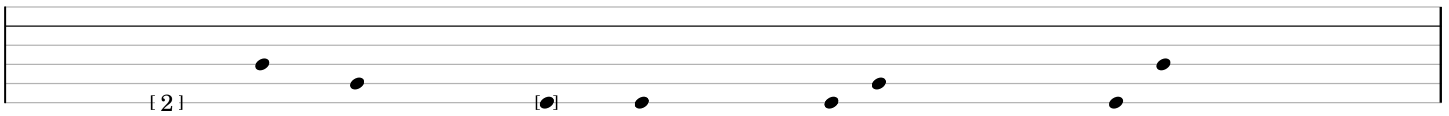
3'24"



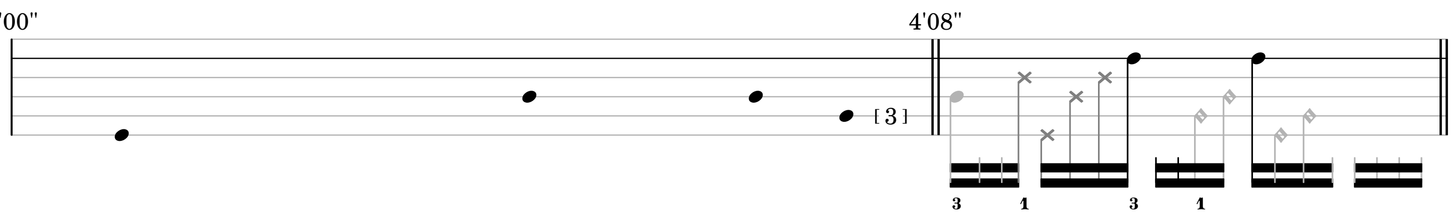
3'36"



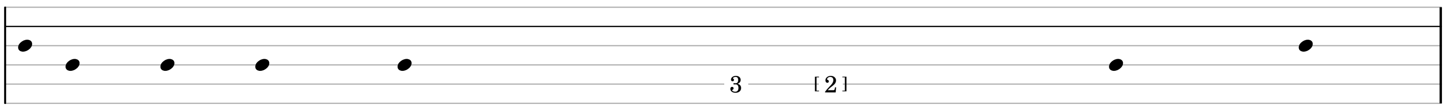
3'48"



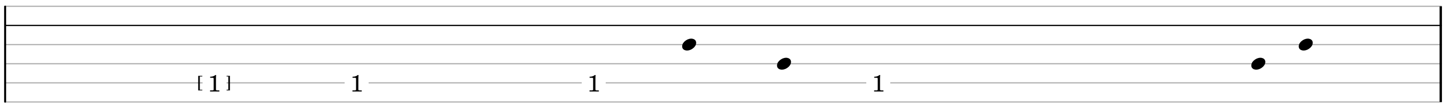
4'00"



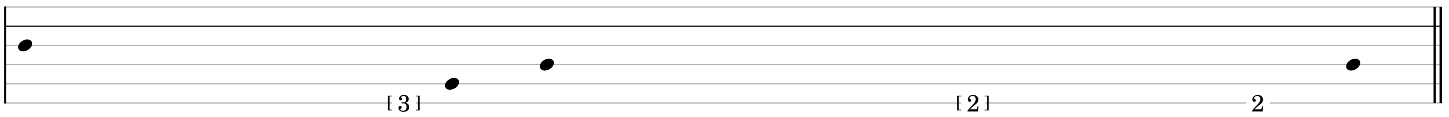
4'12"



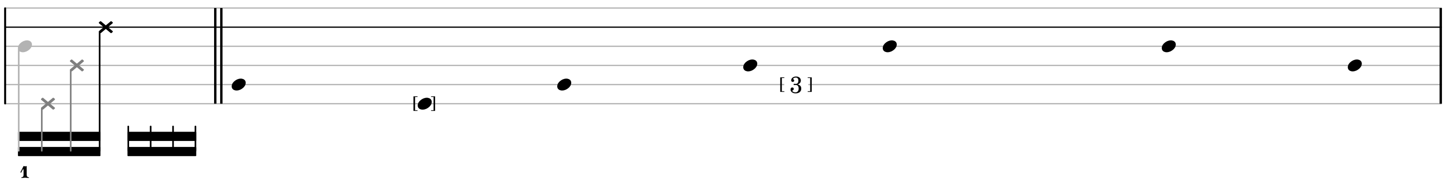
4'24"

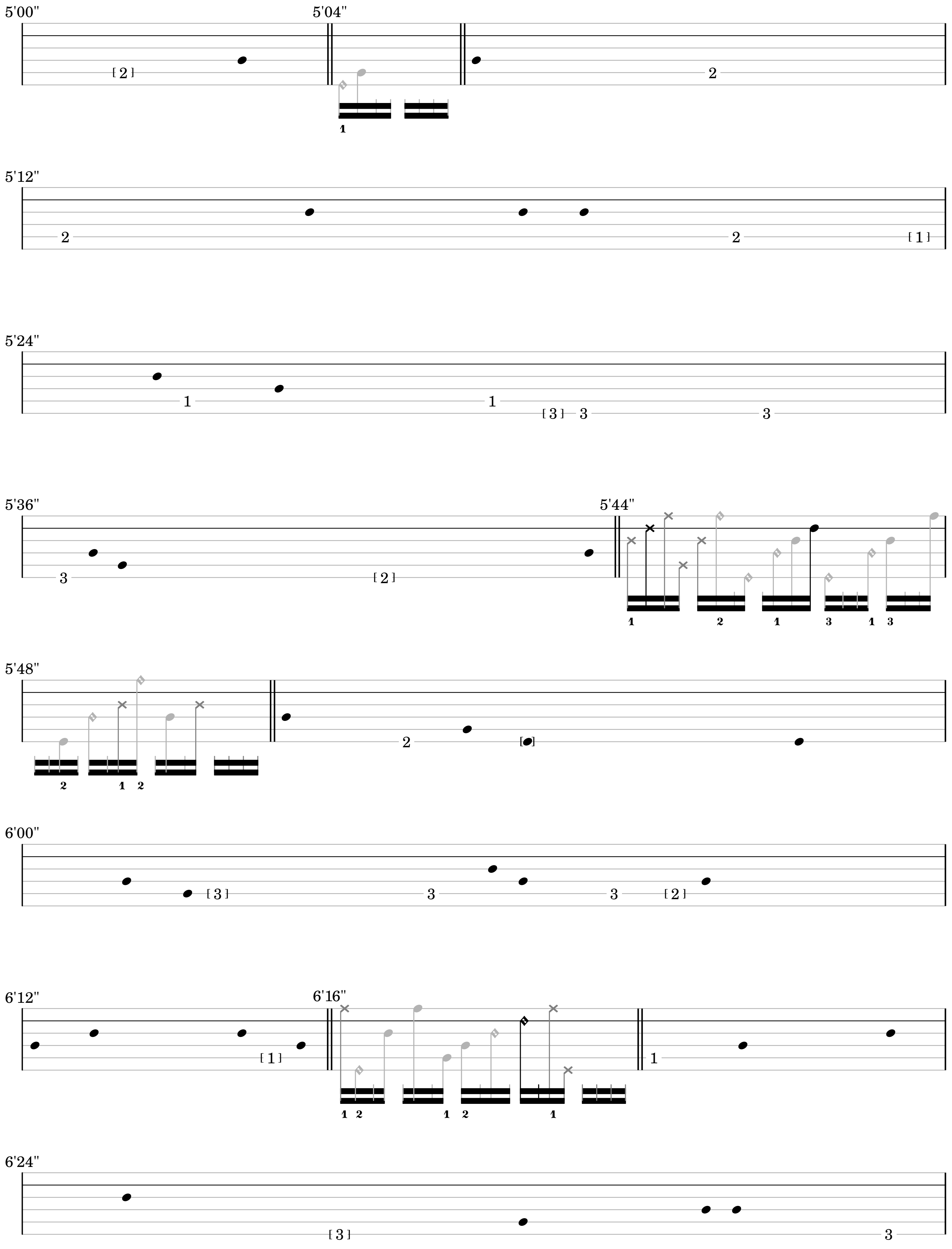


4'36"

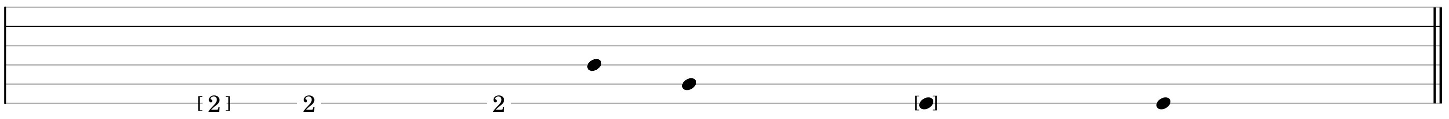


4'48"

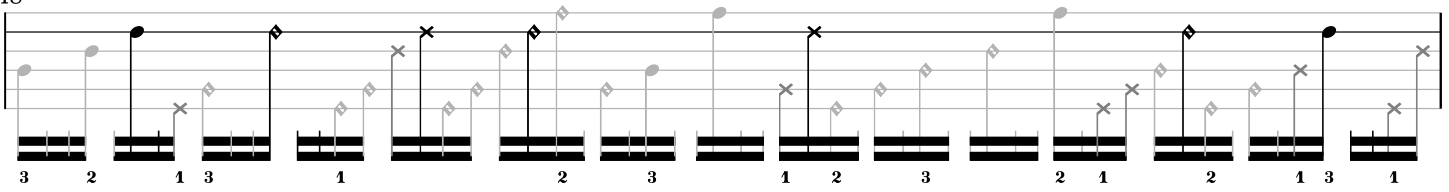



5'00"
5'04"
5'12"
5'24"
5'36"
5'44"
5'48"
6'00"
6'12"
6'16"
6'24"

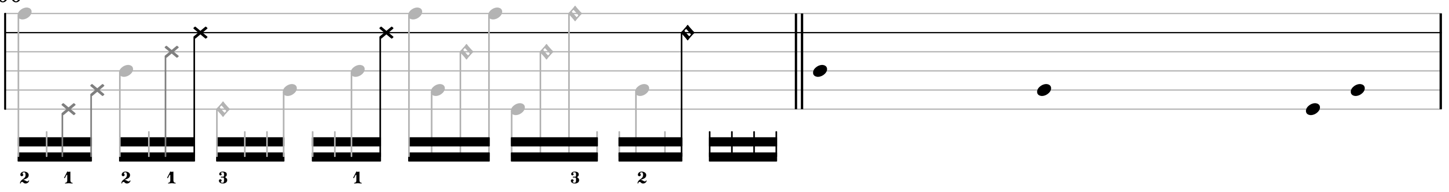
6'36"



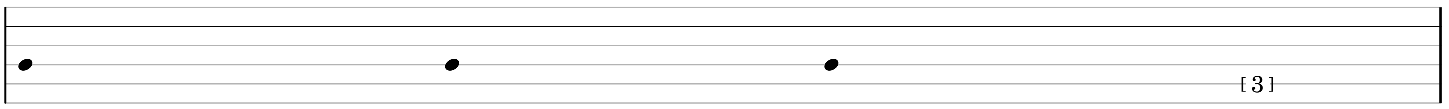
6'48"



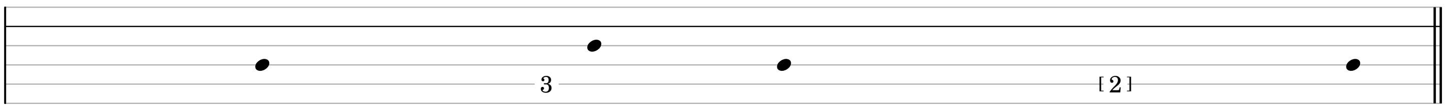
7'00"



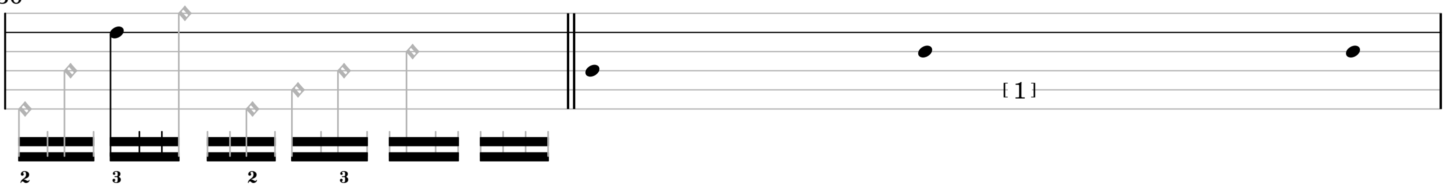
7'12"



7'24"

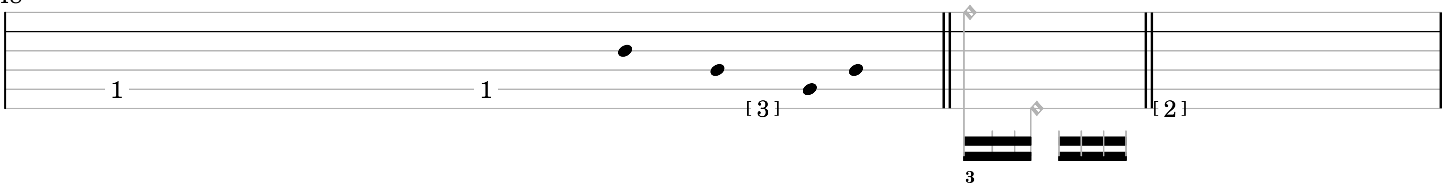


7'36"

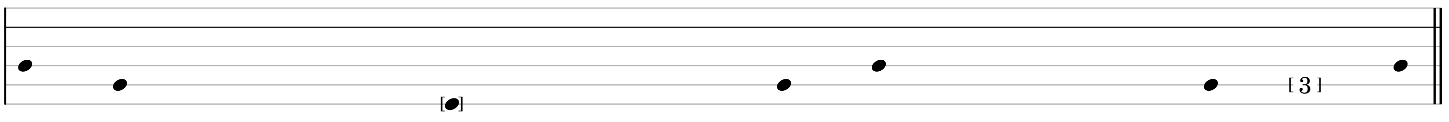


7'48"

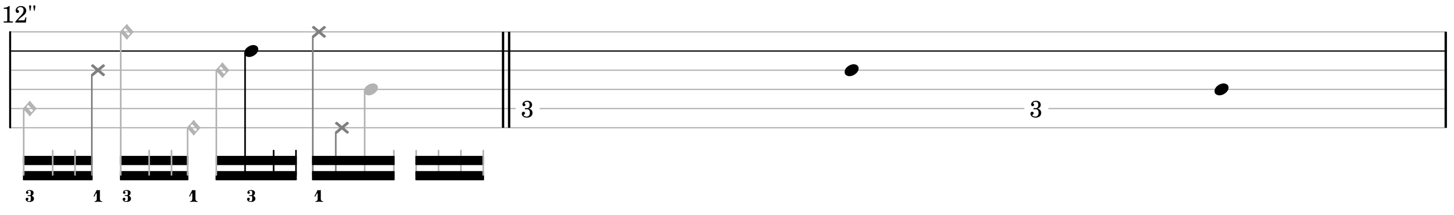
7'56"



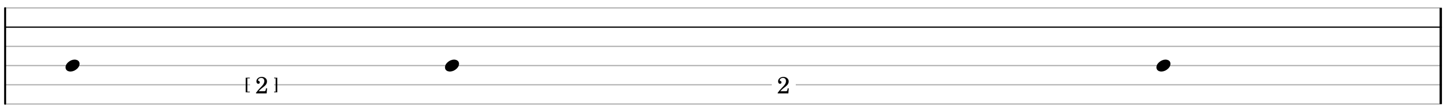
8'00"



8'12"

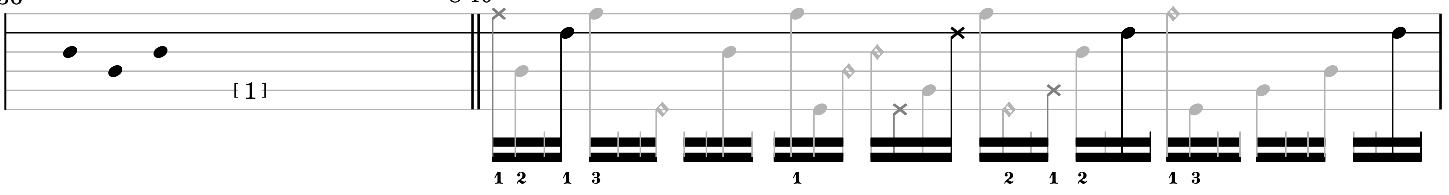


8'24"

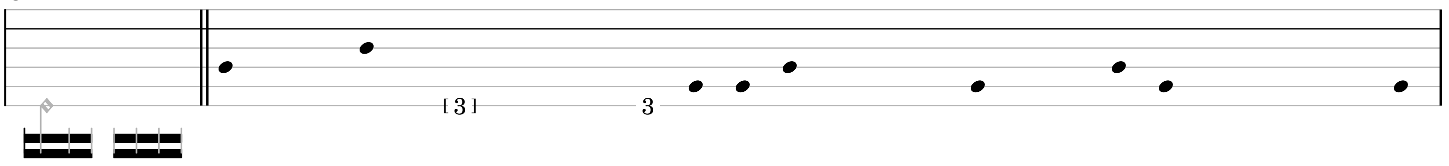


8'36"

8'40"

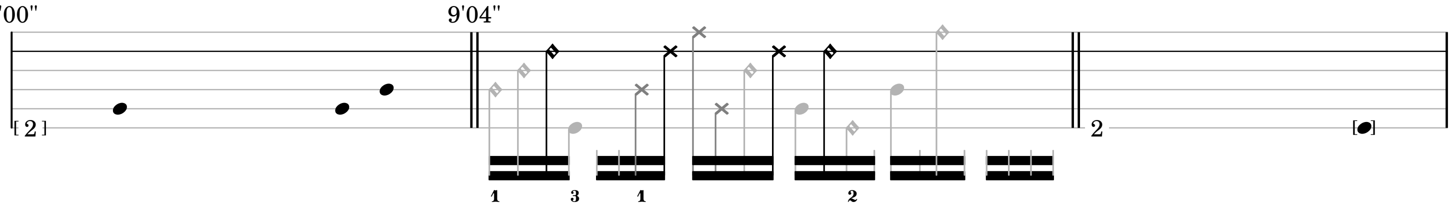


8'48"

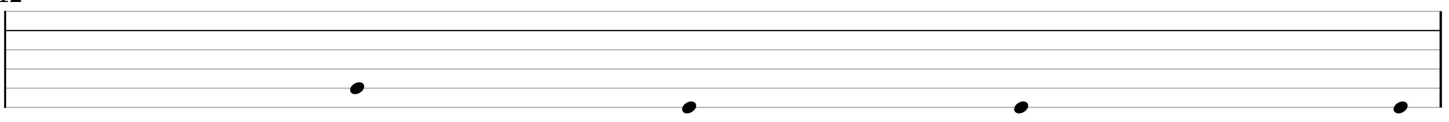


9'00"

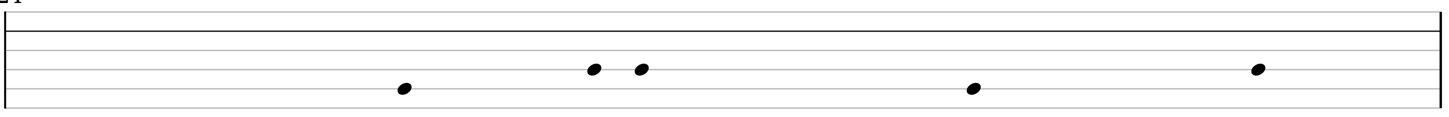
9'04"



9'12"

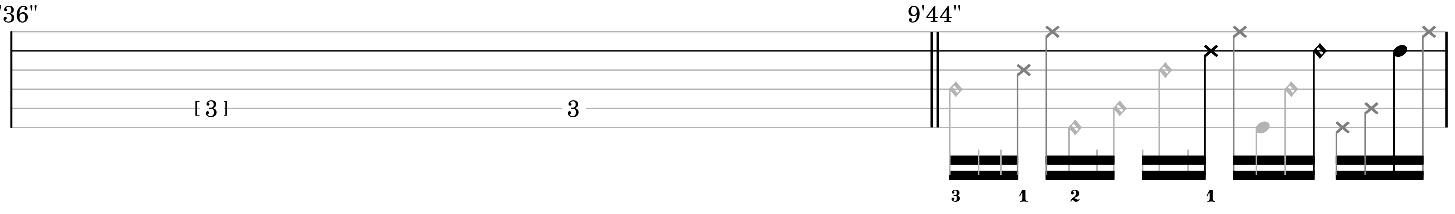


9'24"

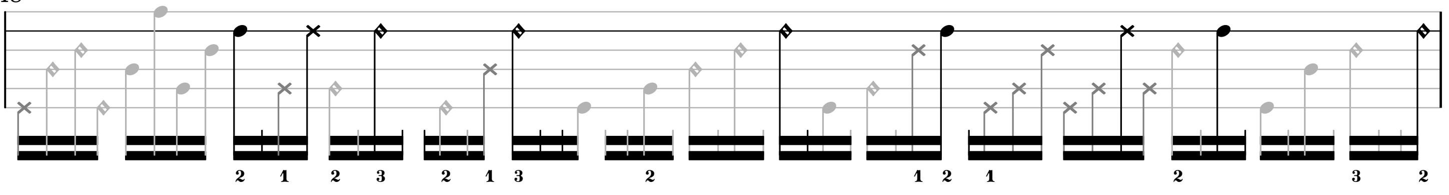


9'36"

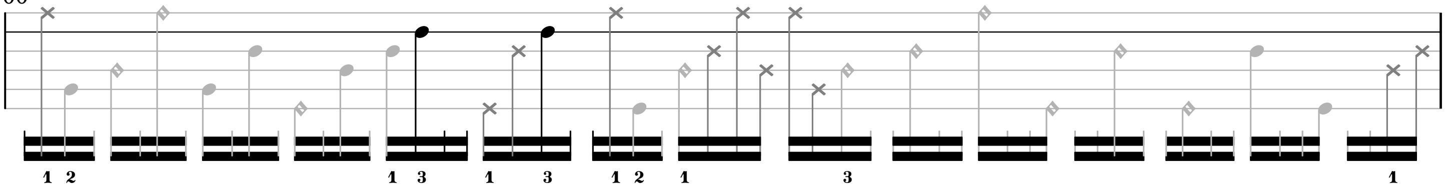
9'44"



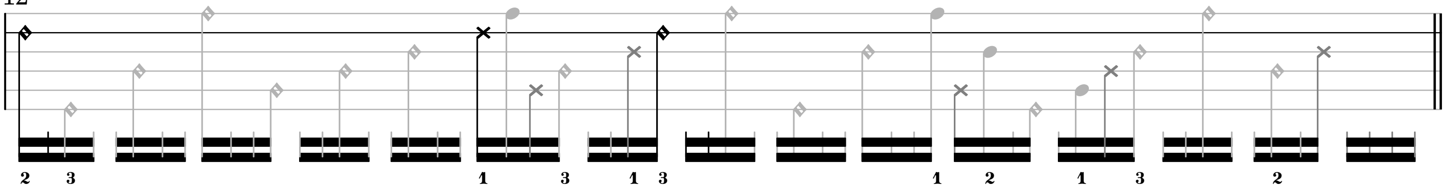
9'48"



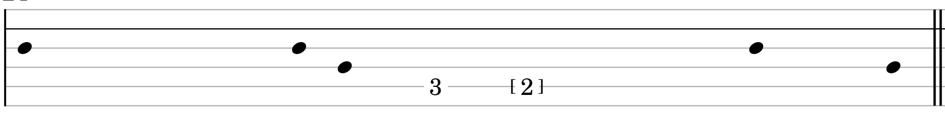
10'00"



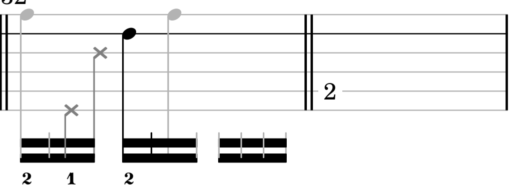
10'12"



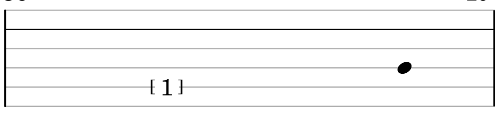
10'24"



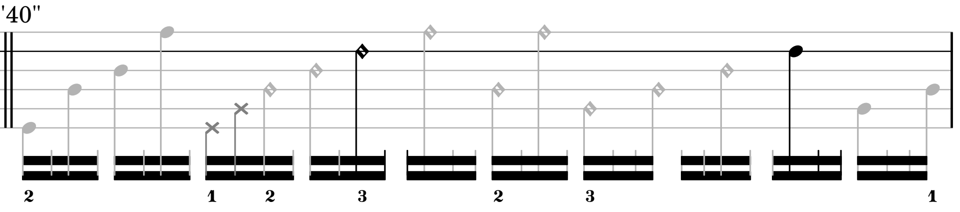
10'32"



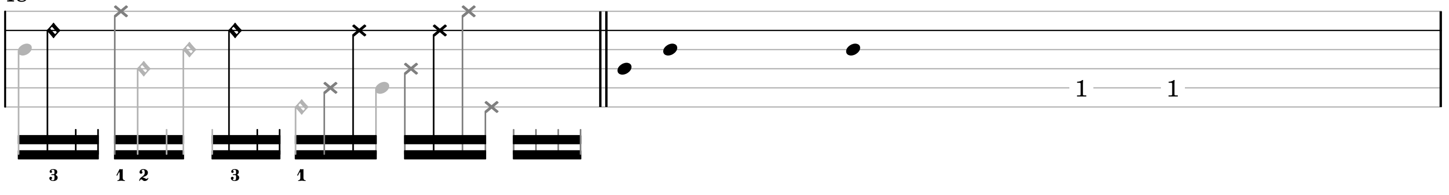
10'36"



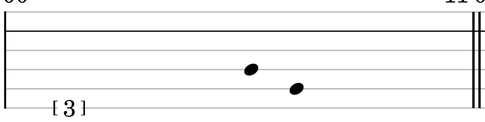
10'40"



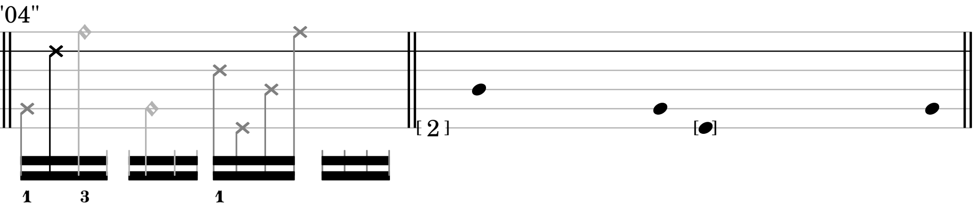
10'48"



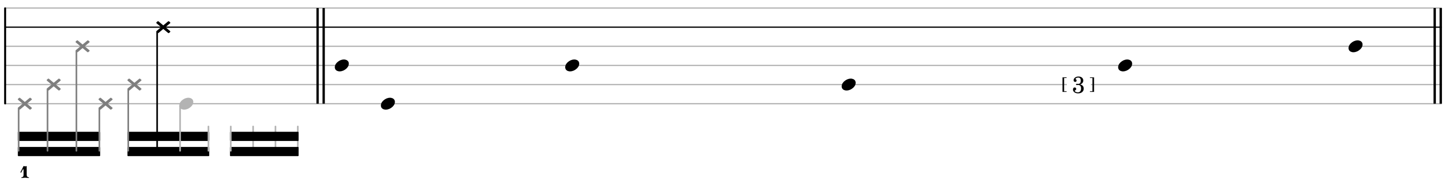
11'00"



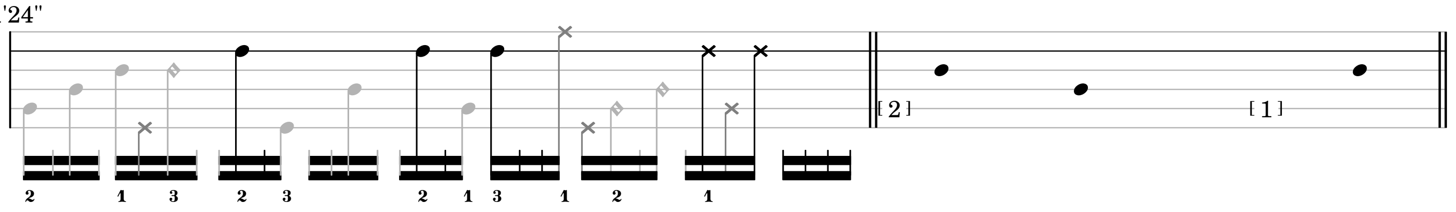
11'04"



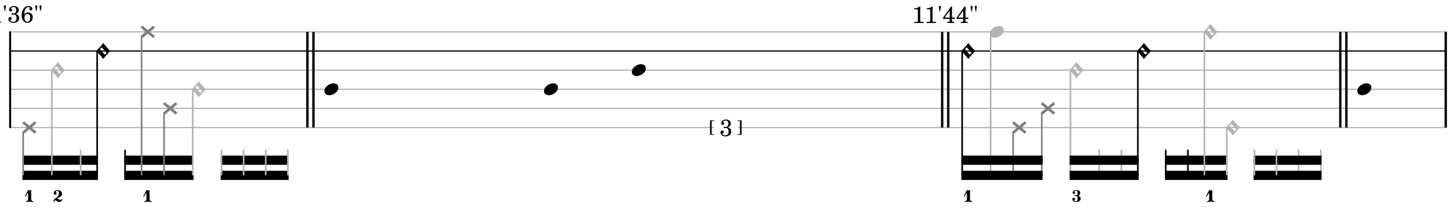
11'12"



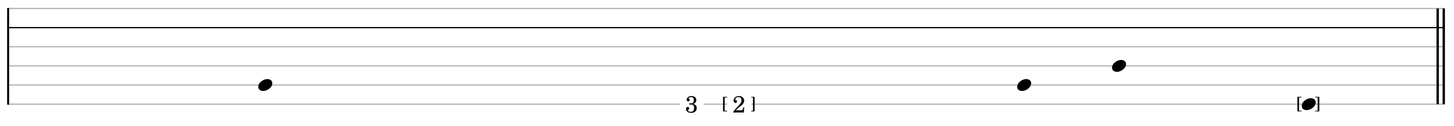
11'24"



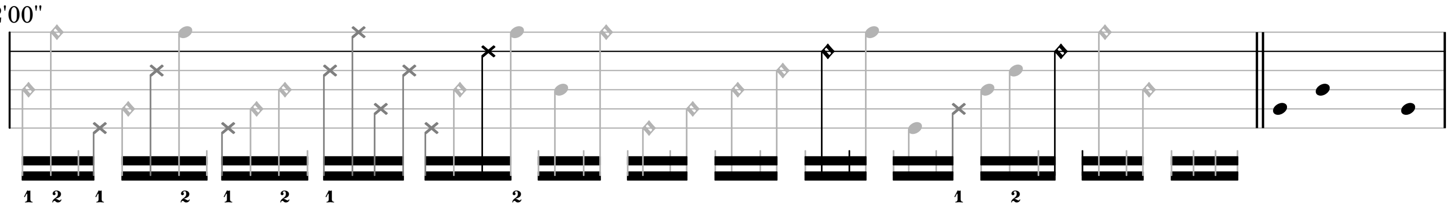
11'36"



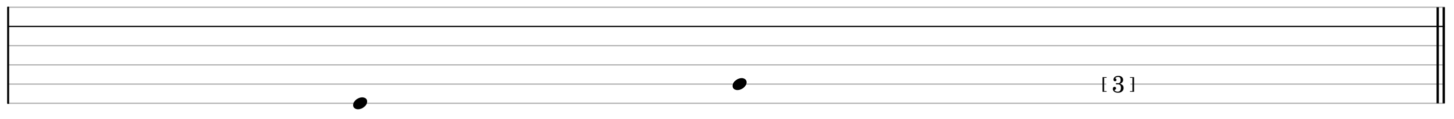
11'48"



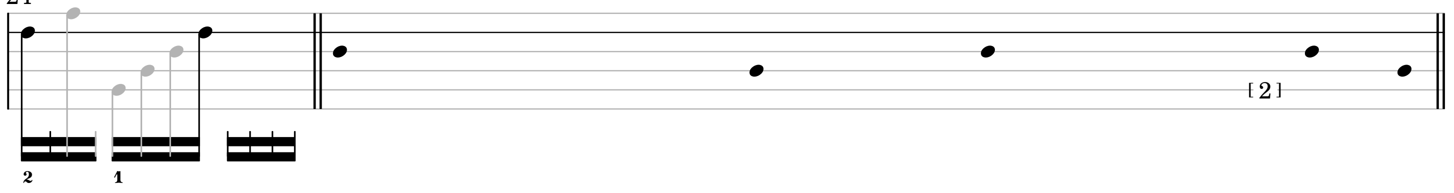
12'00"



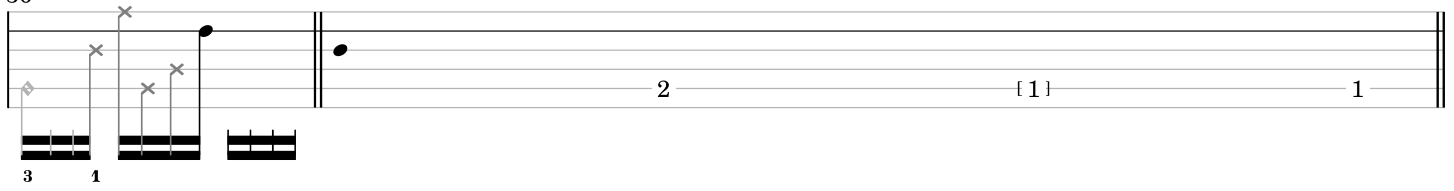
12'12"



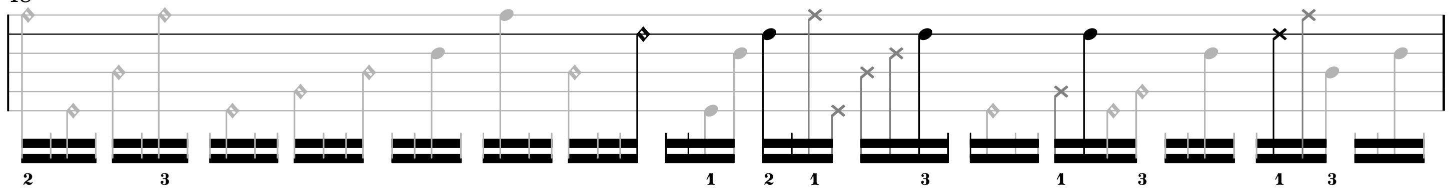
12'24"



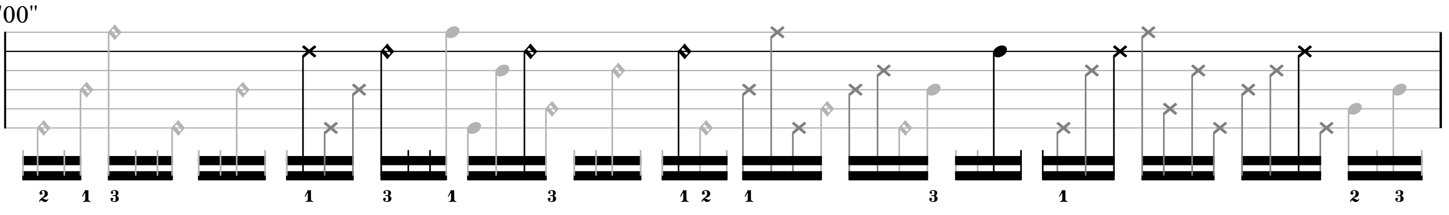
12'36"



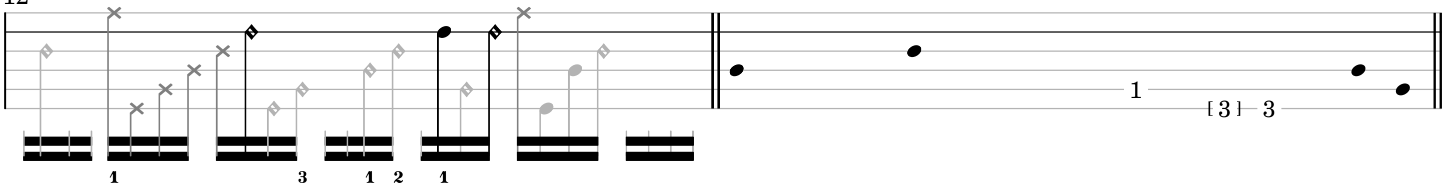
12'48"



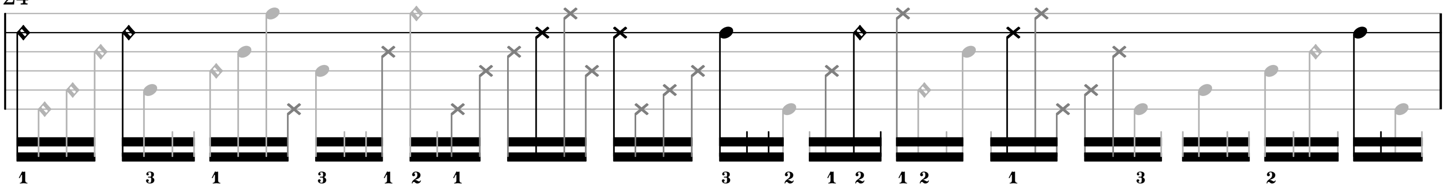
13'00"



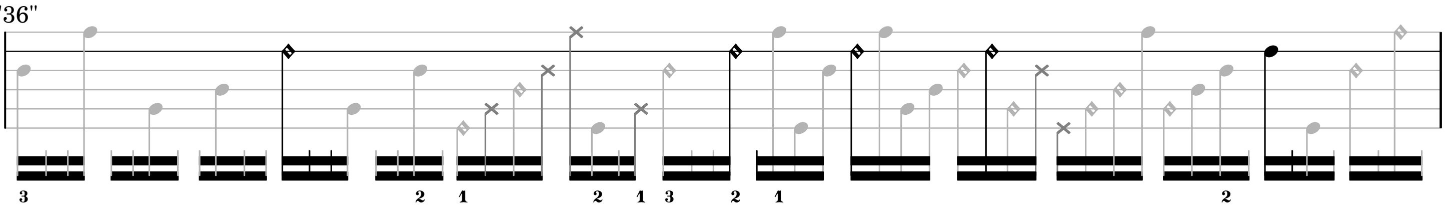
13'12"



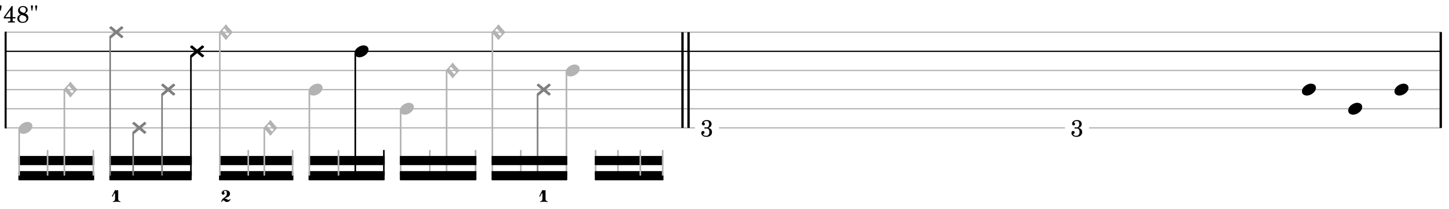
13'24"



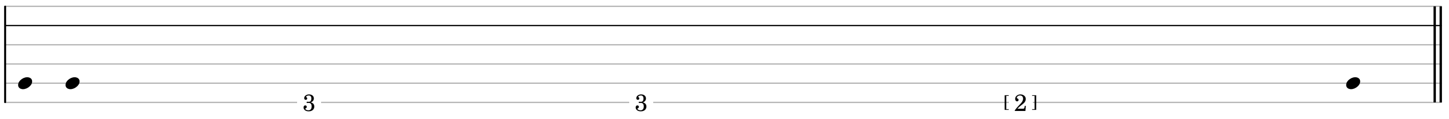
13'36"



13'48"

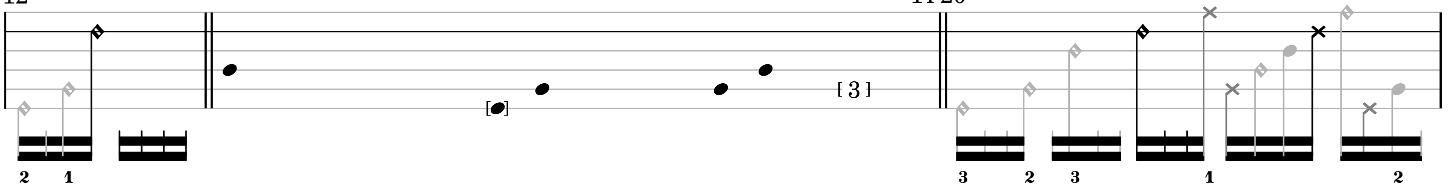


14'00"

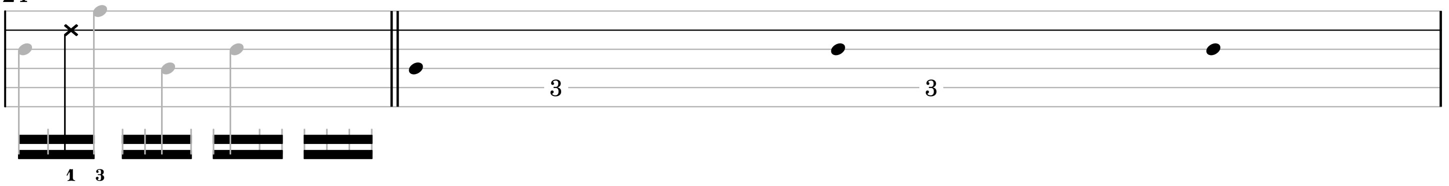


14'12"

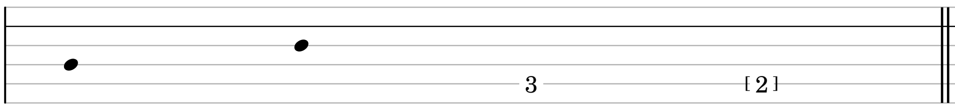
14'20"



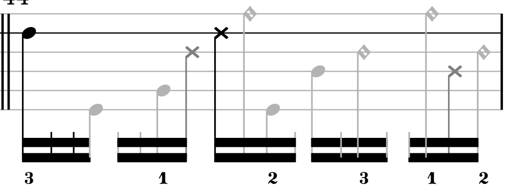
14'24"



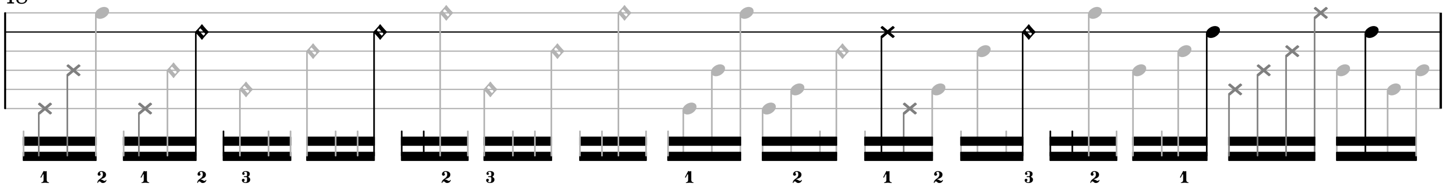
14'36"



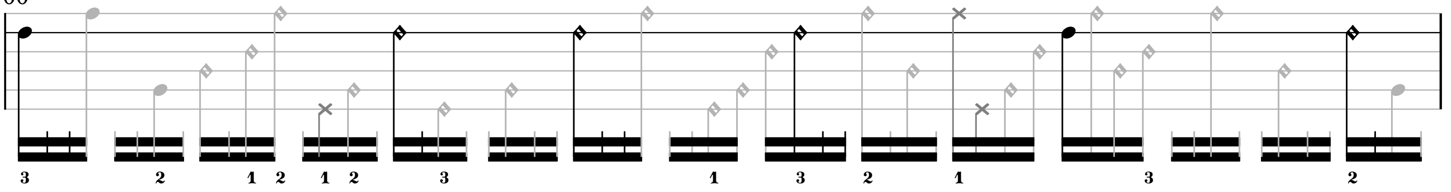
14'44"



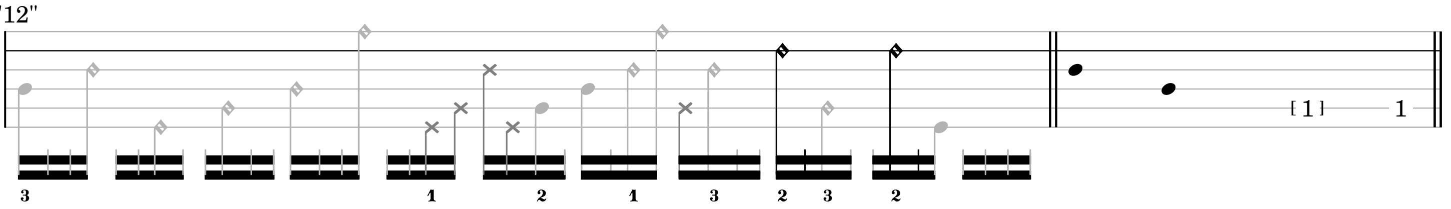
14'48"



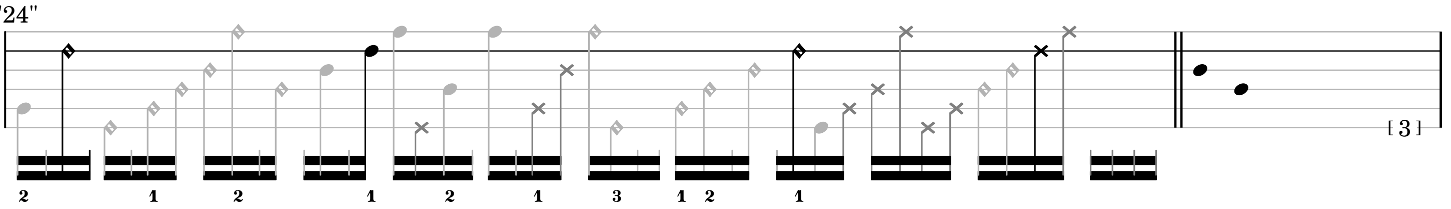
15'00"



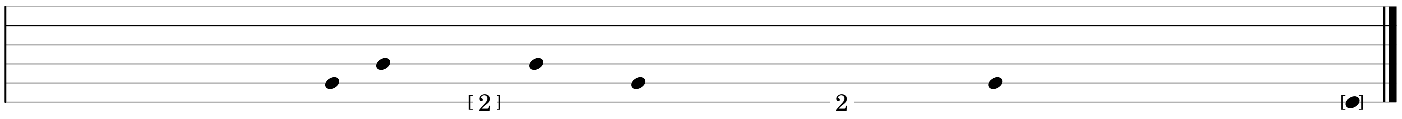
15'12"



15'24"



15'36"

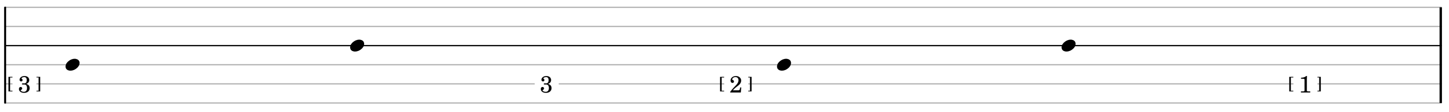


# *ostinato and interrupt*

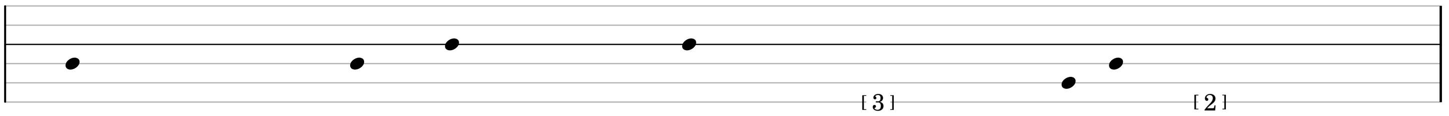
michael winter (mexico city, mx; 2017)
version generated: 2017.12.06

ensemble part 3

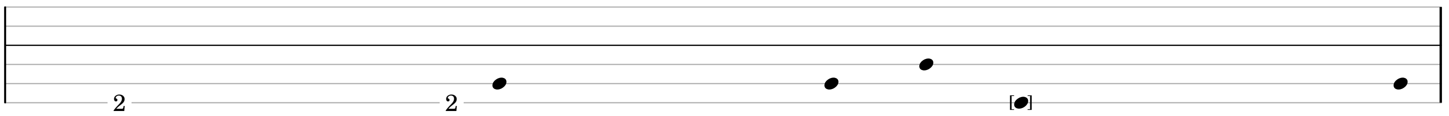
0'12"



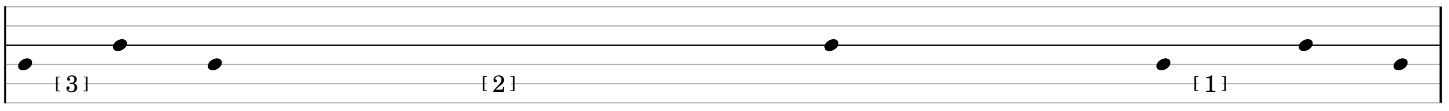
0'24"



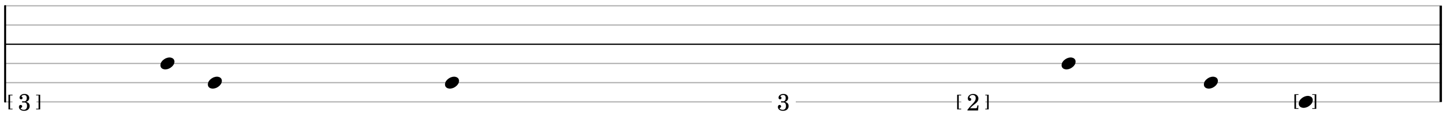
0'36"



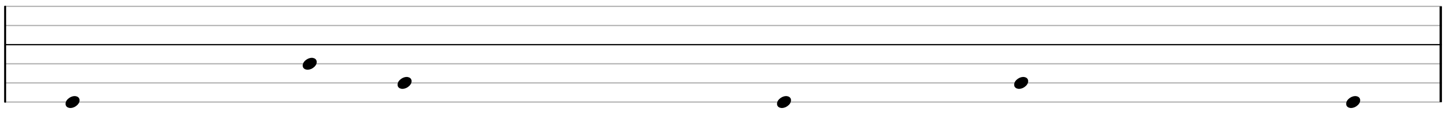
0'48"



1'00"

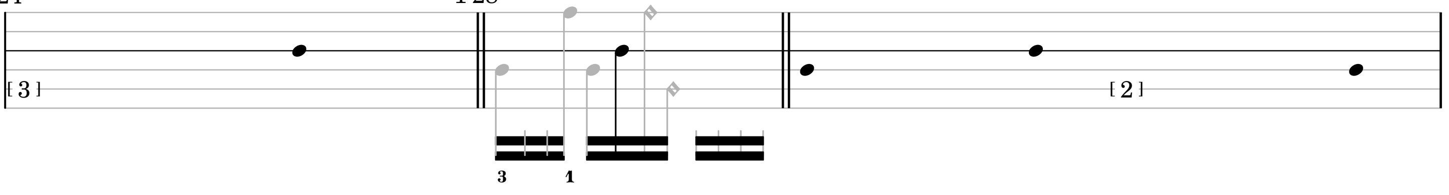


1'12"

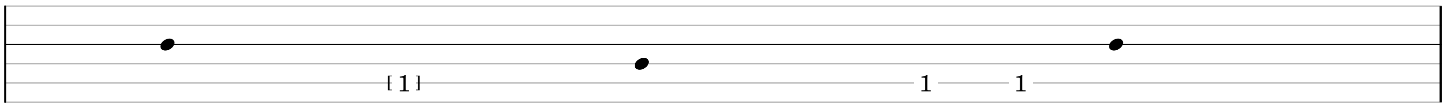


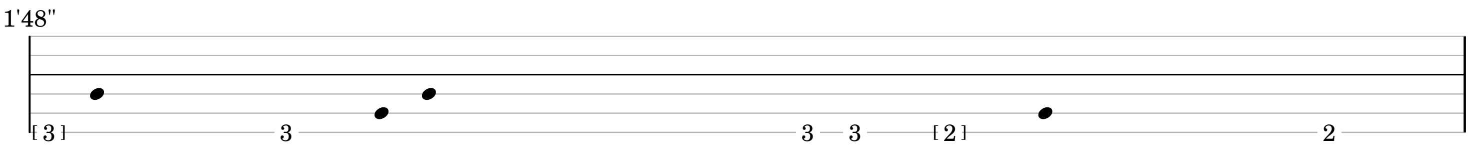
1'24"

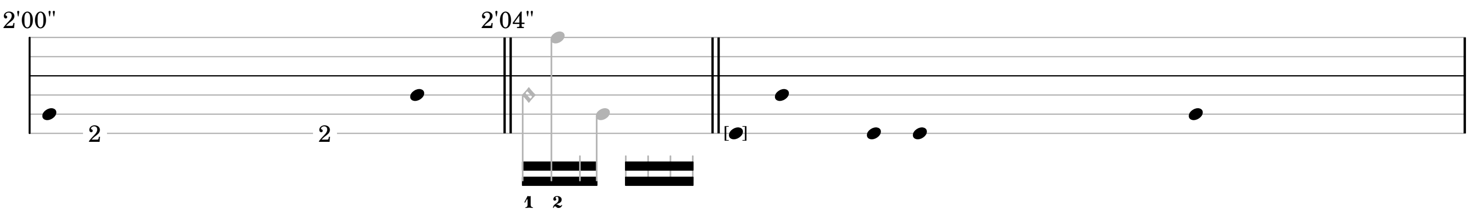
1'28"

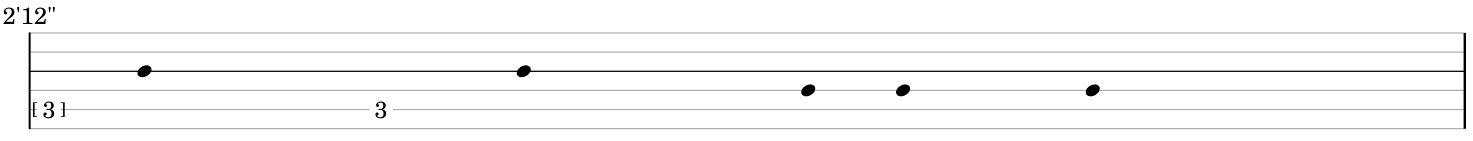


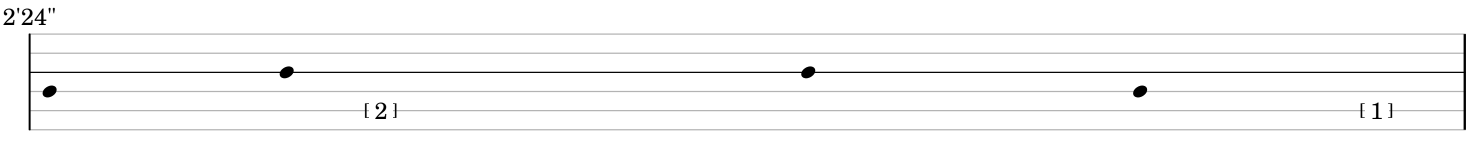
1'36"

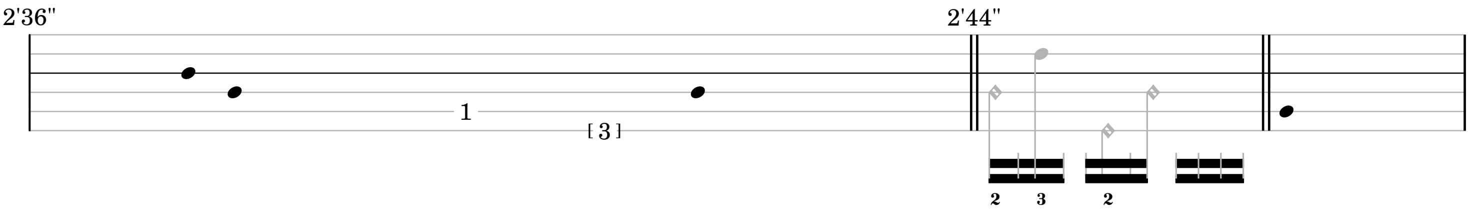


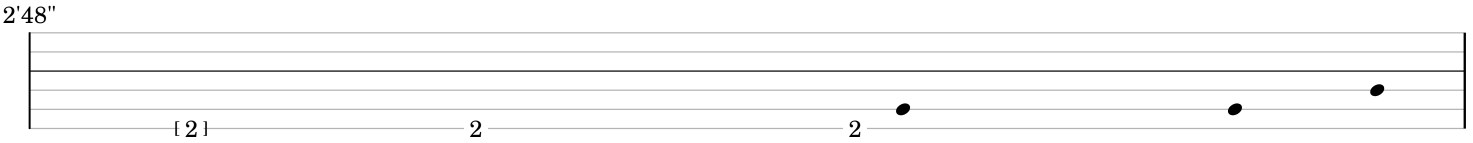

1'48"
[ 3 ]
3
3
3
[ 2 ]
2

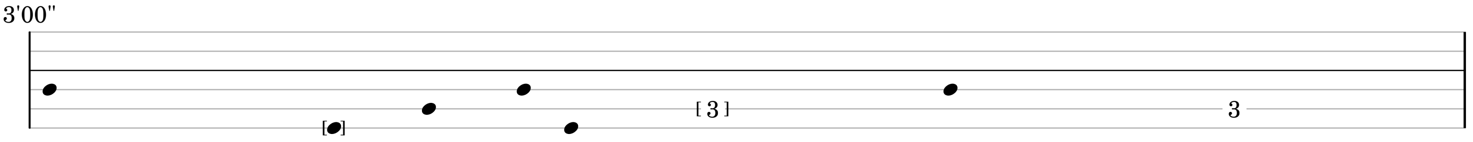

2'00"
2
2
2'04"
1
2

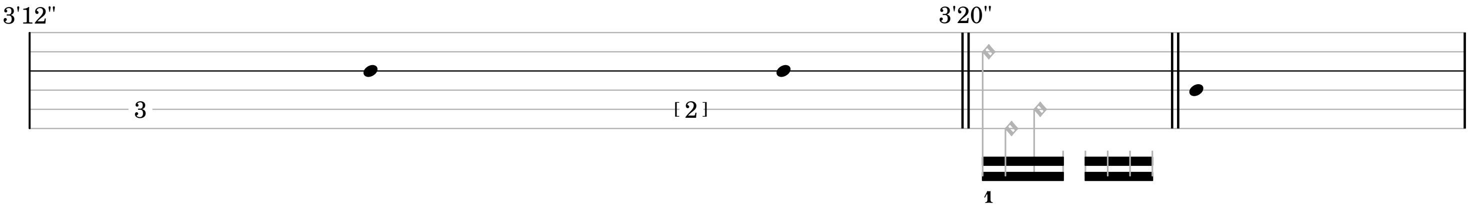

2'12"
[ 3 ]
3


2'24"
[ 2 ]
[ 1 ]

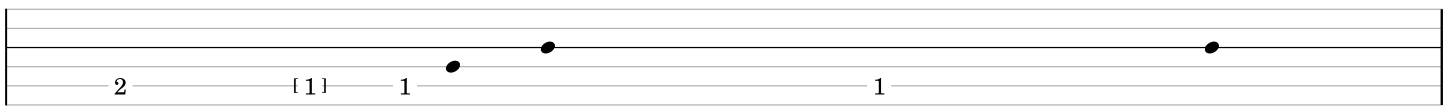

2'36"
1
[ 3 ]
2'44"
2
3
2


2'48"
[ 2 ]
2
2

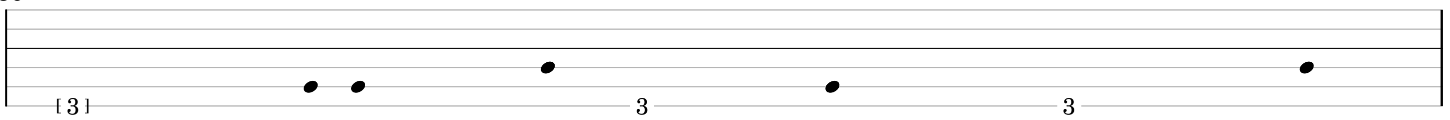

3'00"
[ 3 ]
3


3'12"
3
[ 2 ]
3'20"
1

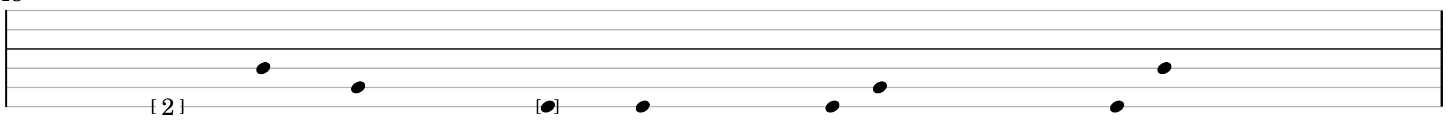
3'24"



3'36"

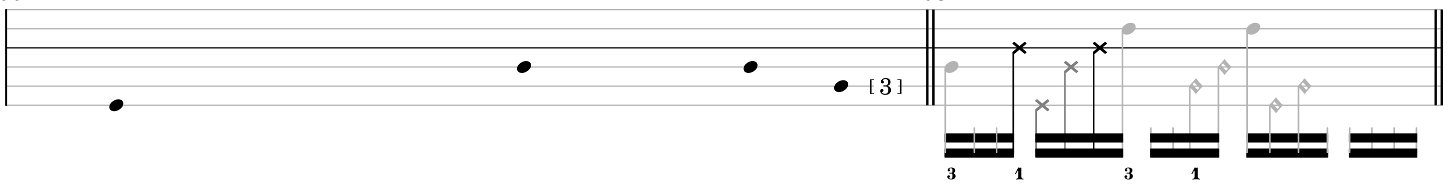


3'48"

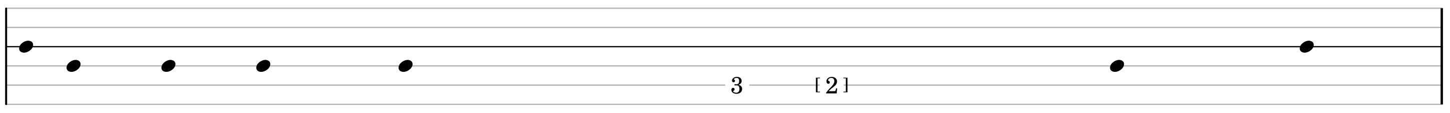


4'00"

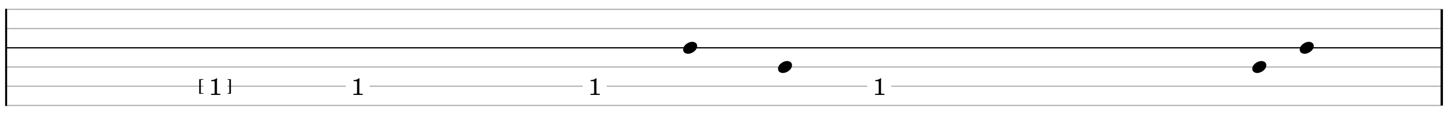
4'08"



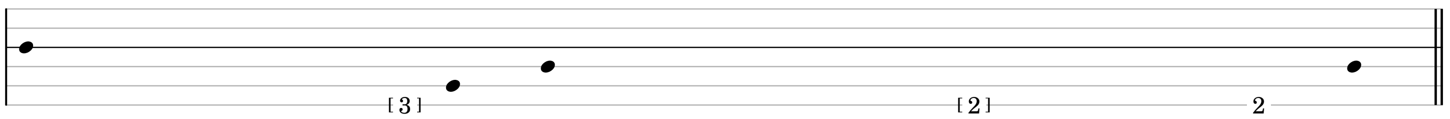
4'12"



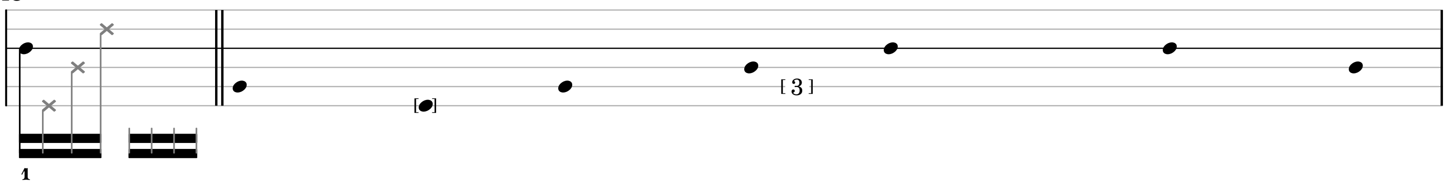
4'24"

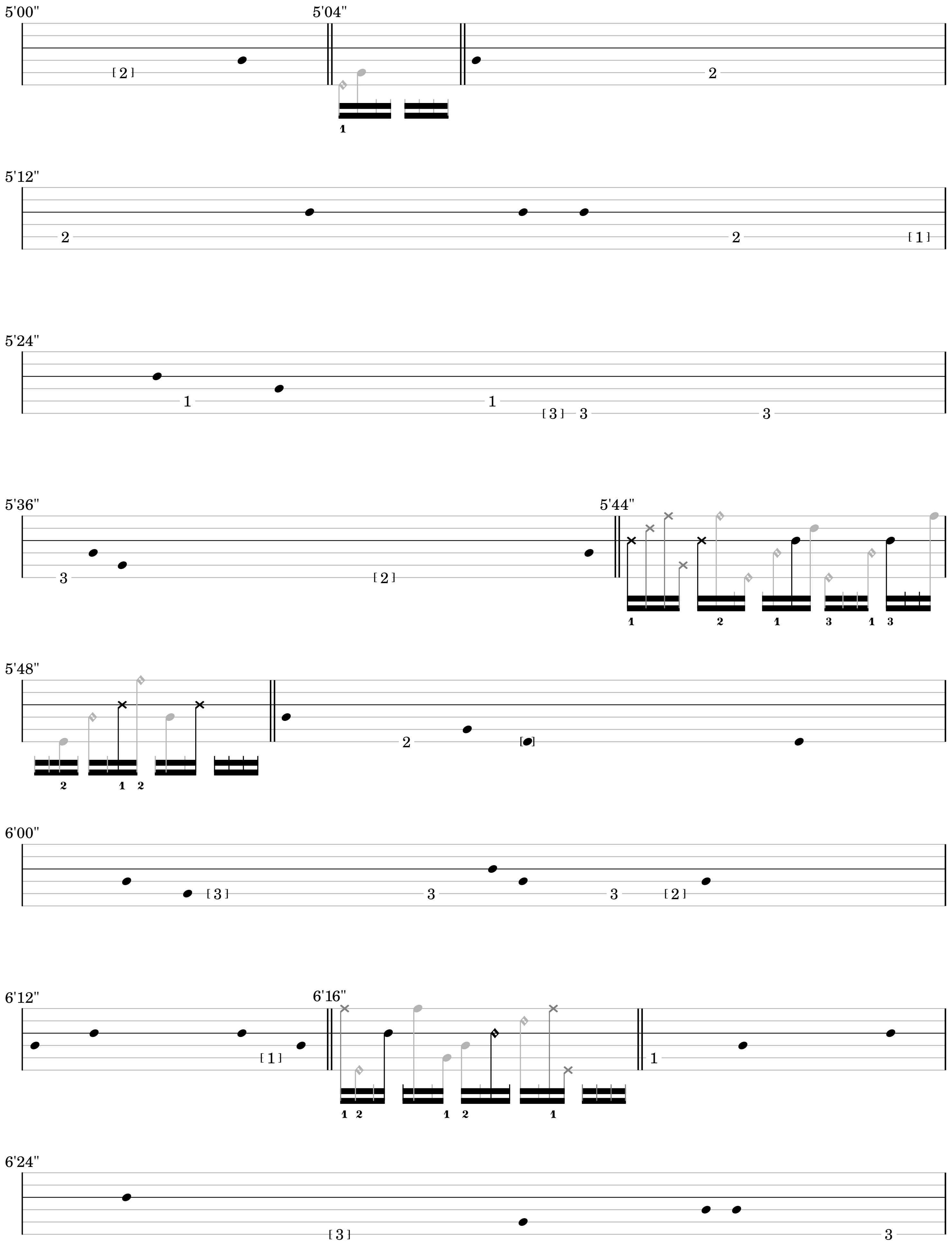


4'36"

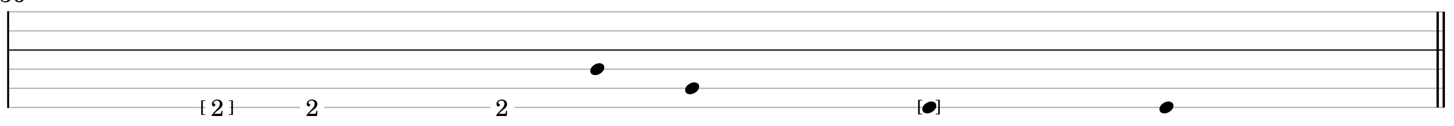


4'48"

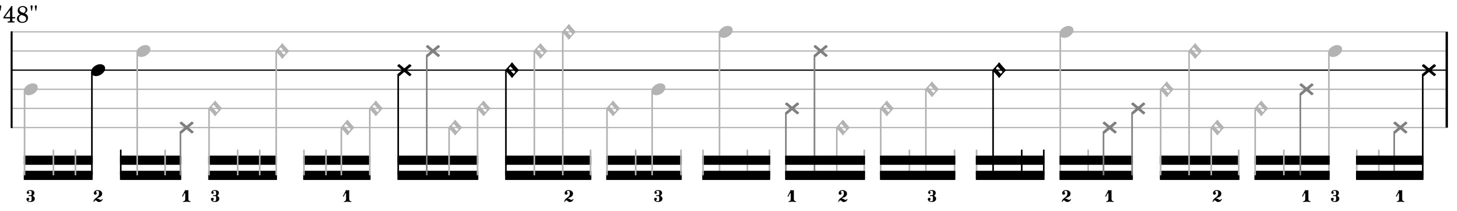



5'00"
5'04"
5'12"
5'24"
5'36"
5'44"
5'48"
6'00"
6'12"
6'16"
6'24"

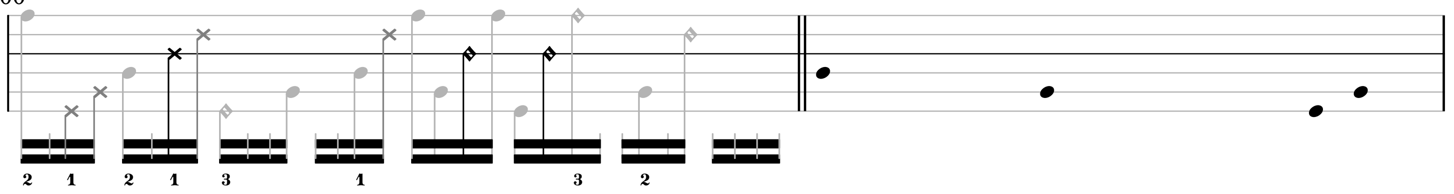
6'36"



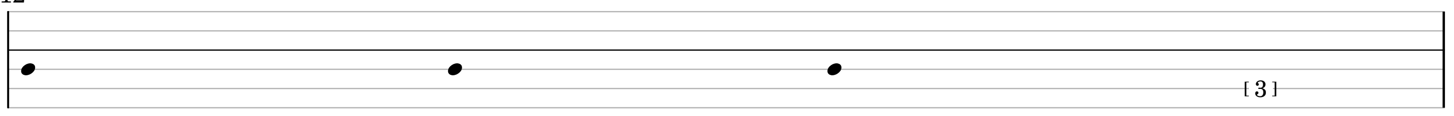
6'48"



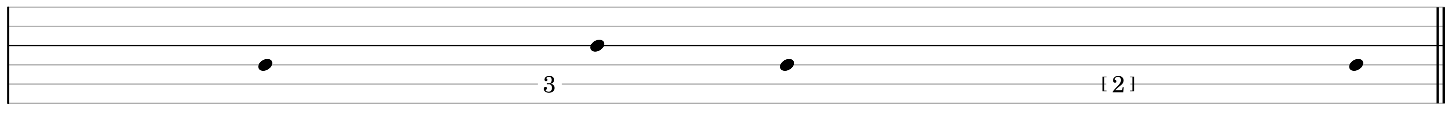
7'00"



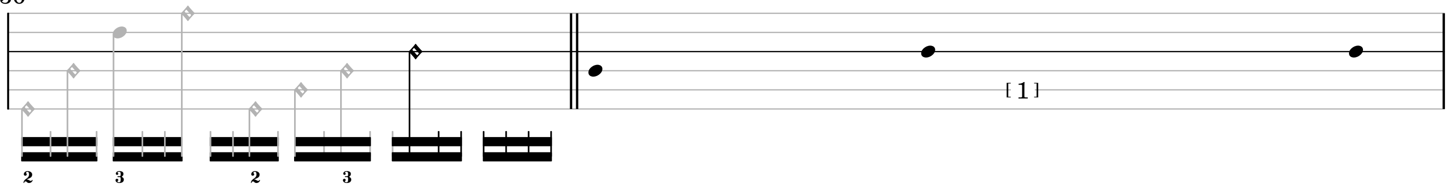
7'12"



7'24"

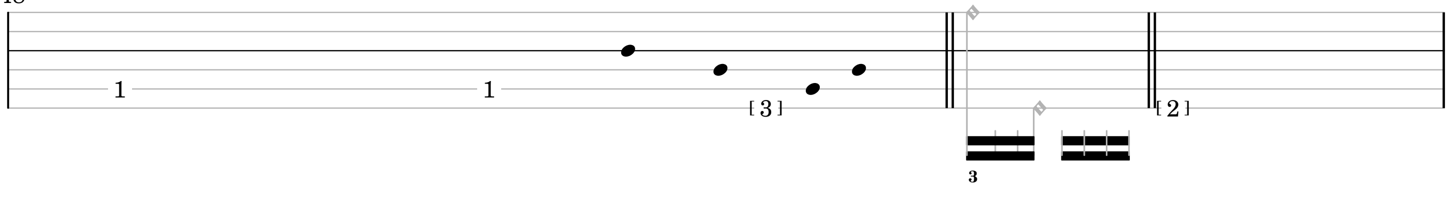


7'36"

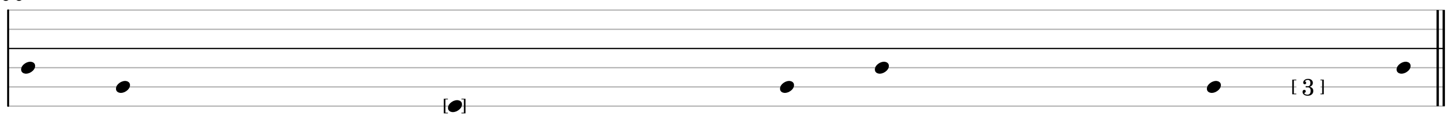


7'48"

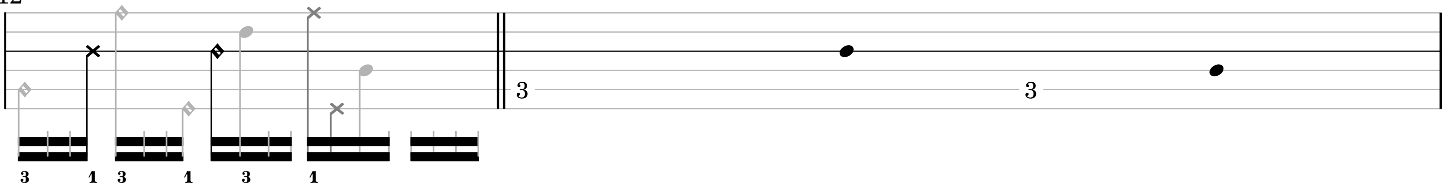
7'56"



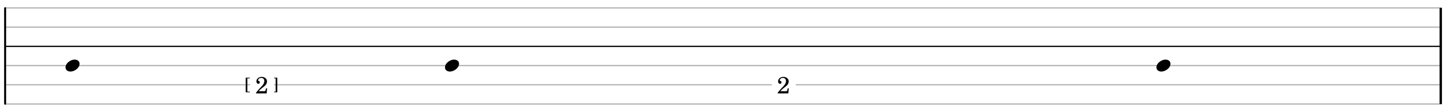
8'00"



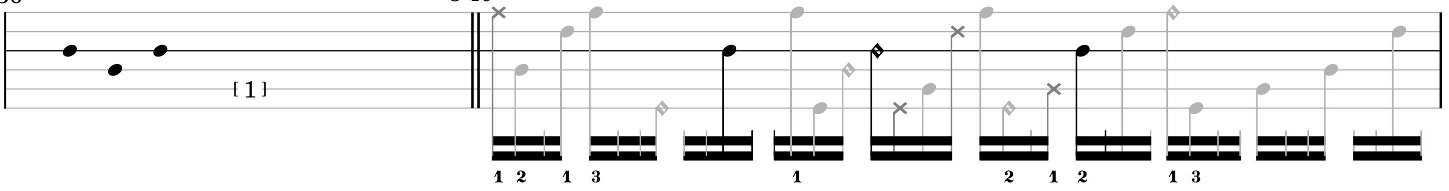
8'12"



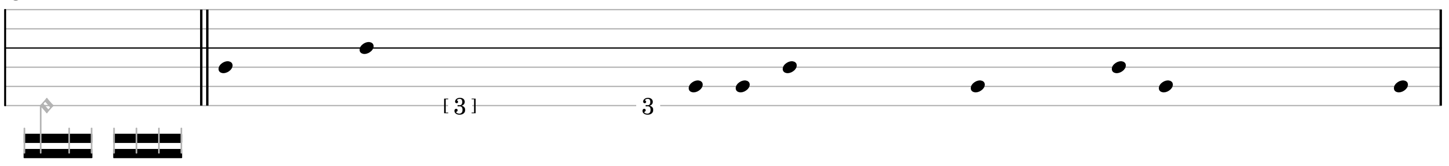
8'24"



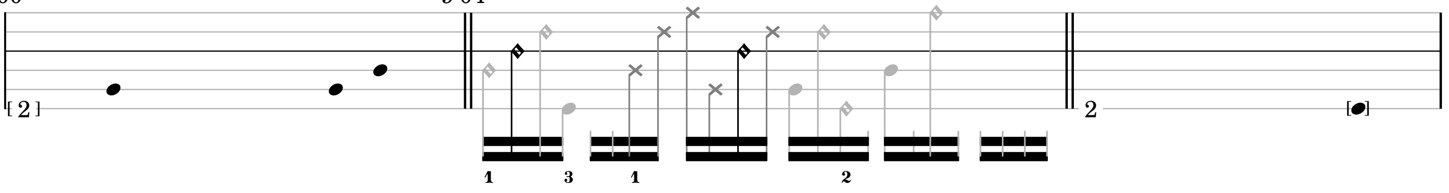
8'36" 8'40"



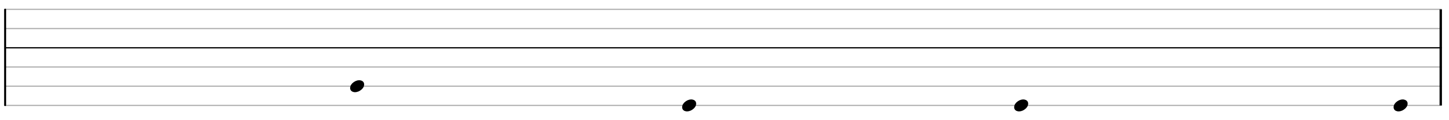
8'48"



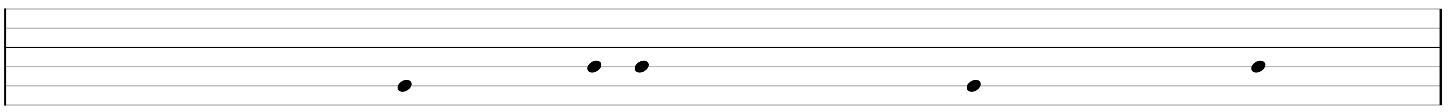
9'00" 9'04"



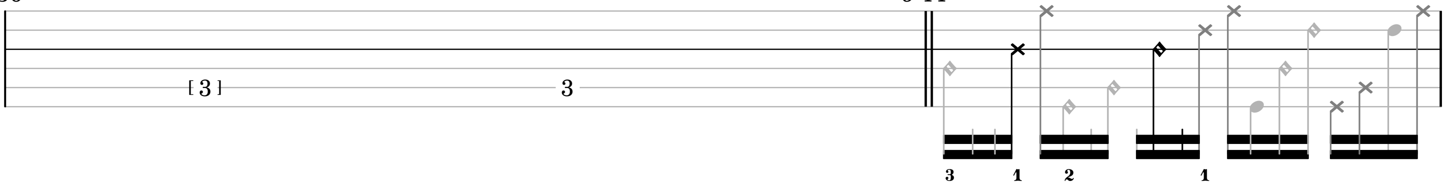
9'12"

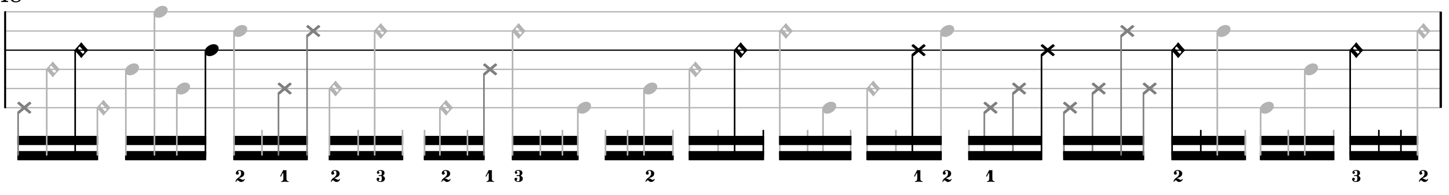
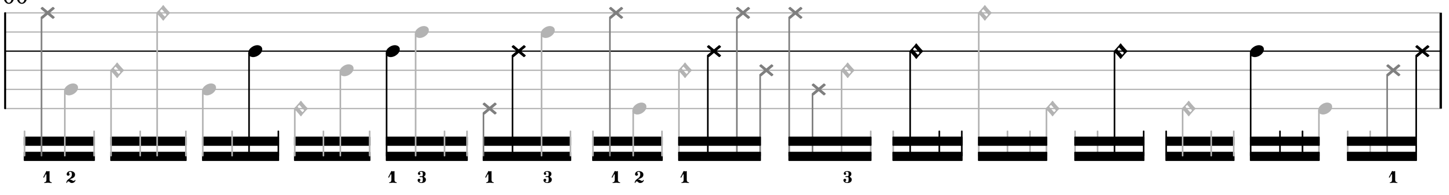
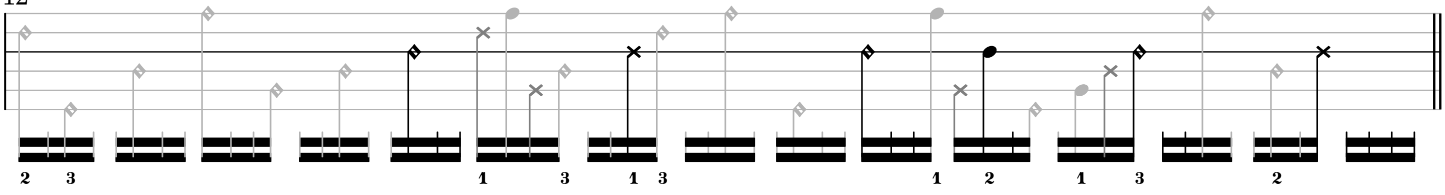
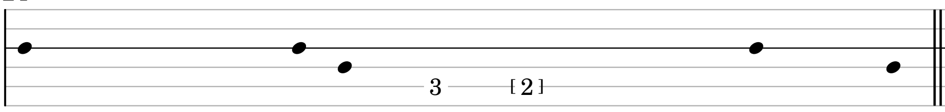
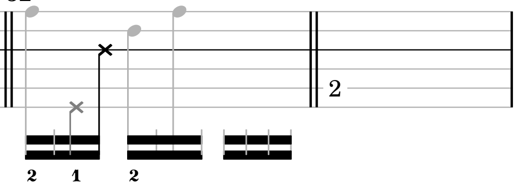


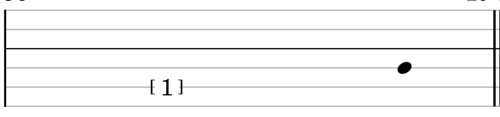
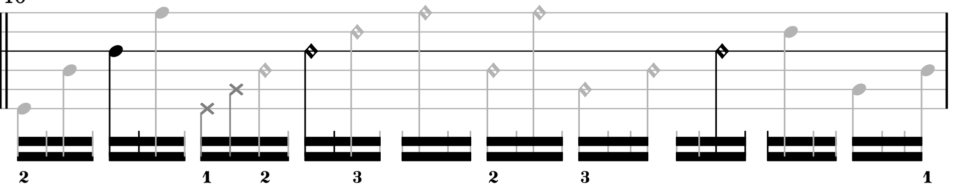
9'24"

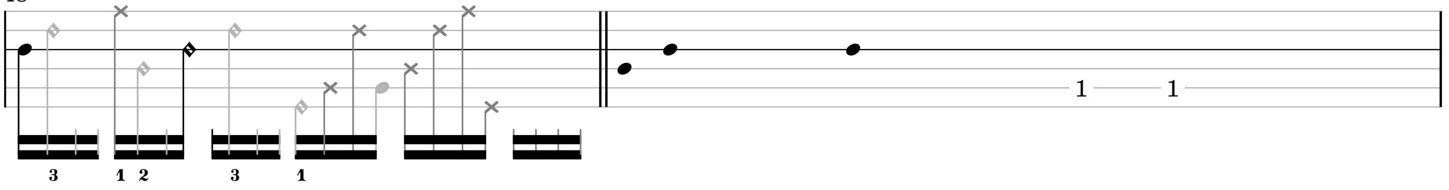


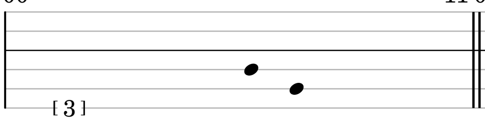
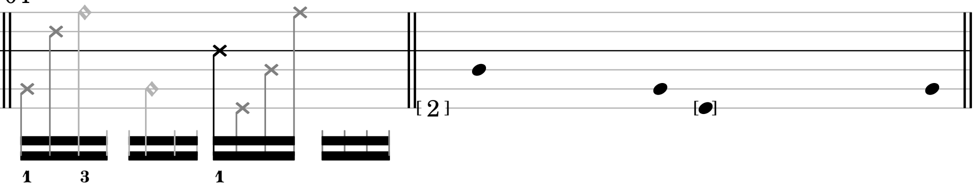
9'36" 9'44"

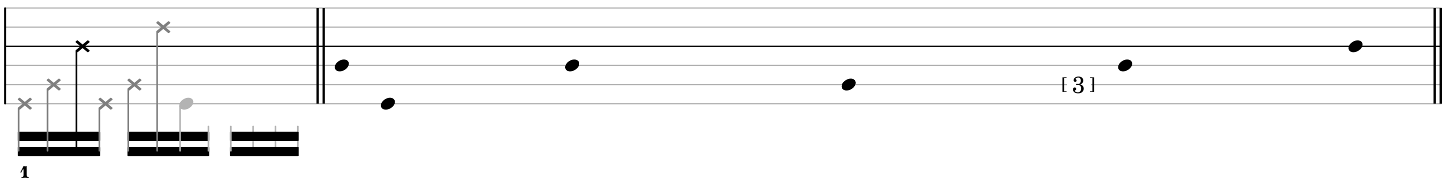


9'48"

10'00"

10'12"

10'24"

10'32"


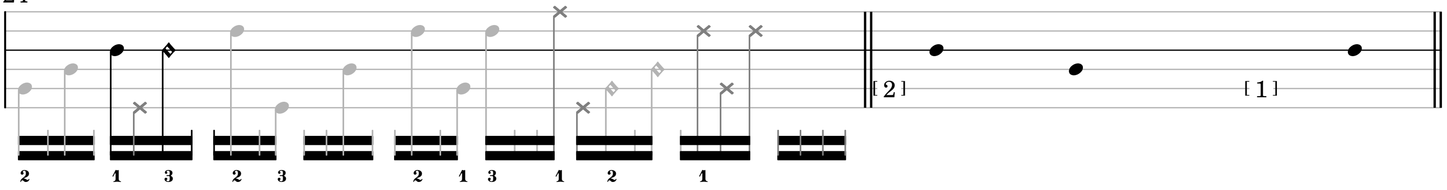
10'36"

10'40"


10'48"


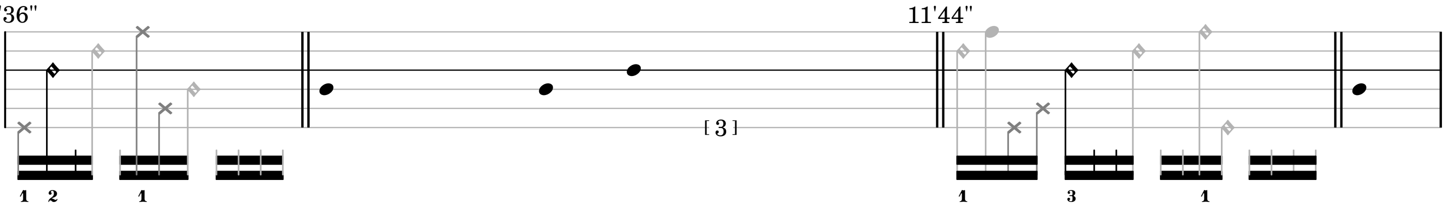
11'00"

11'04"


11'12"


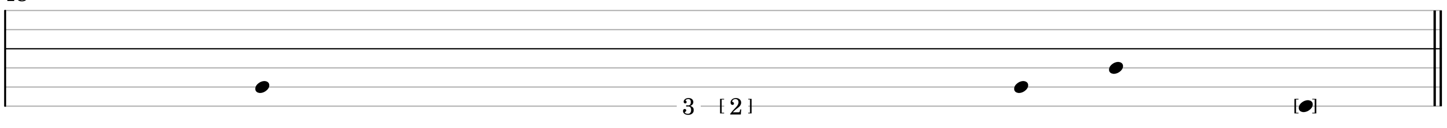
11'24"



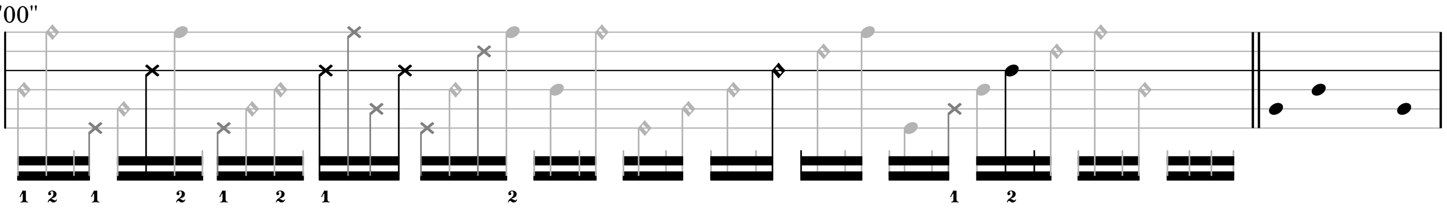
11'36"



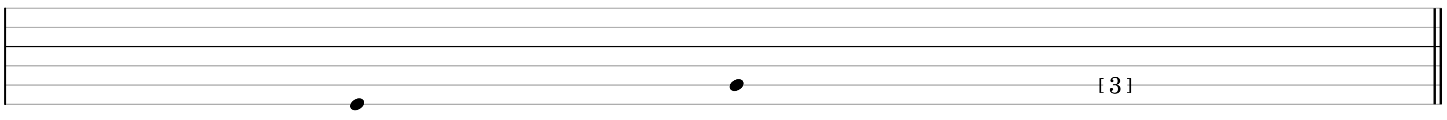
11'48"



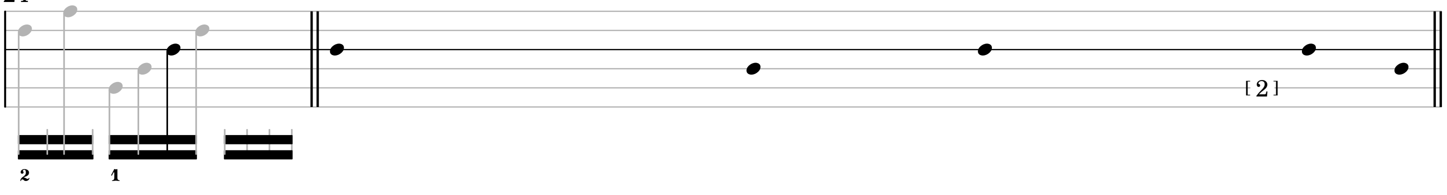
12'00"



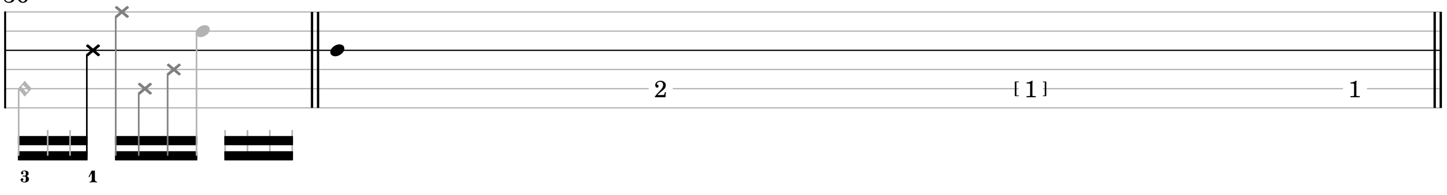
12'12"



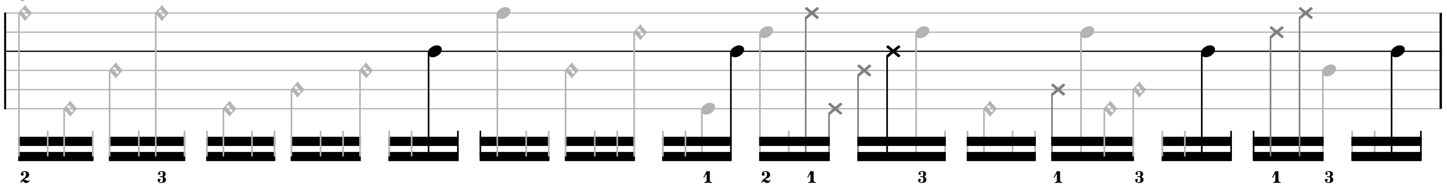
12'24"



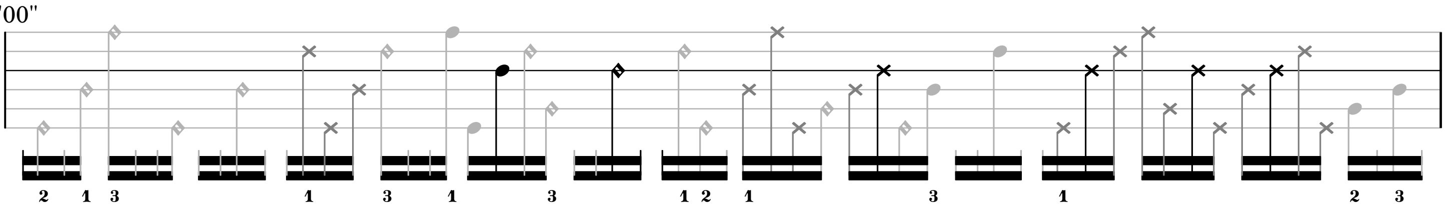
12'36"



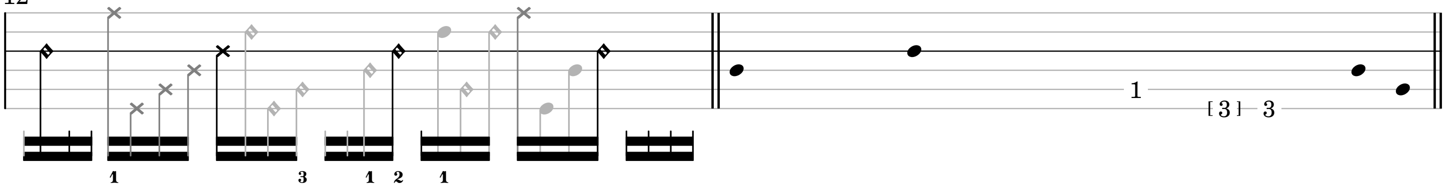
12'48"



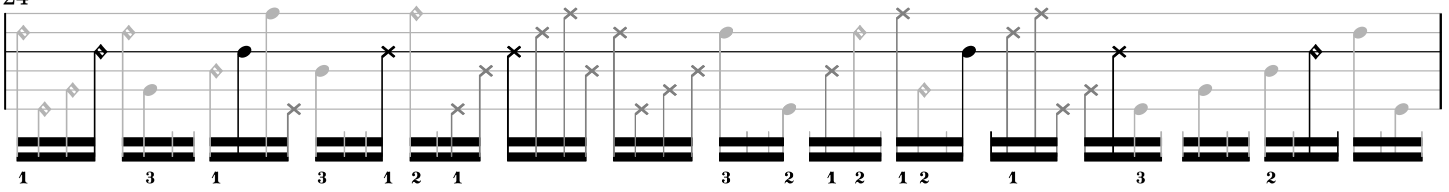
13'00"



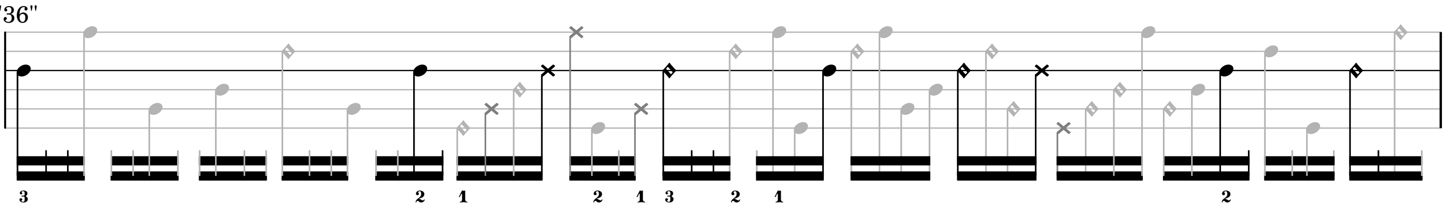
13'12"



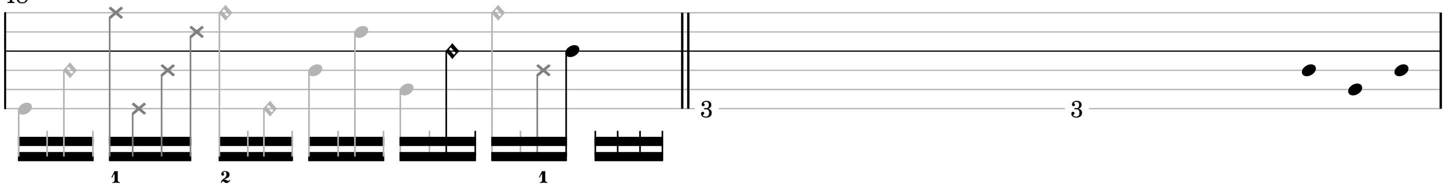
13'24"



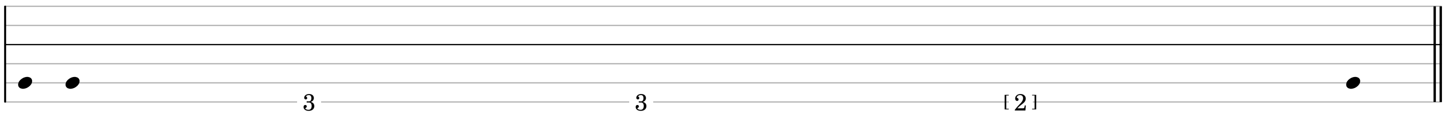
13'36"



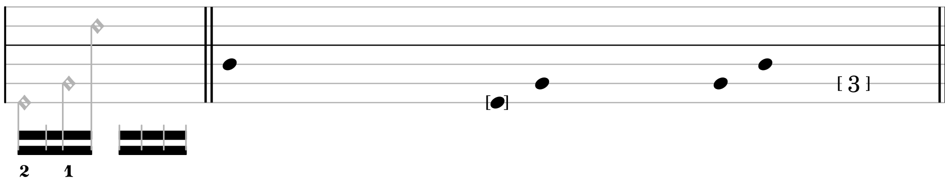
13'48"



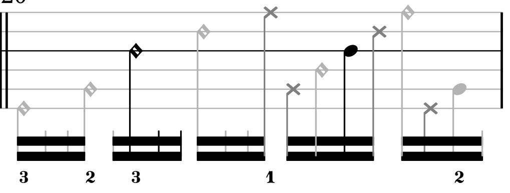
14'00"



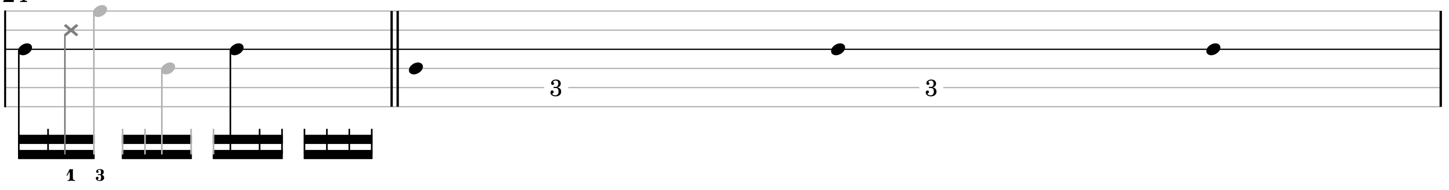
14'12"



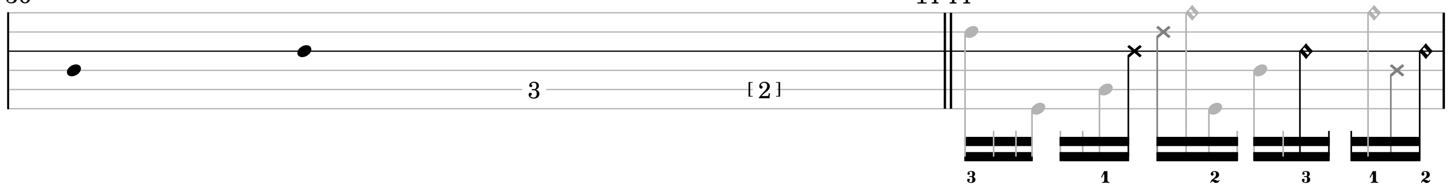
14'20"



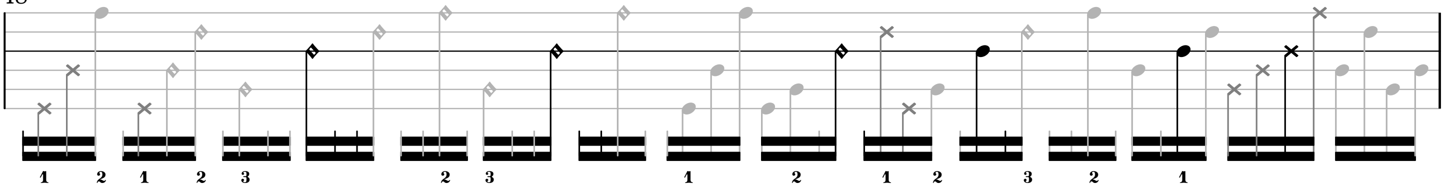
14'24"



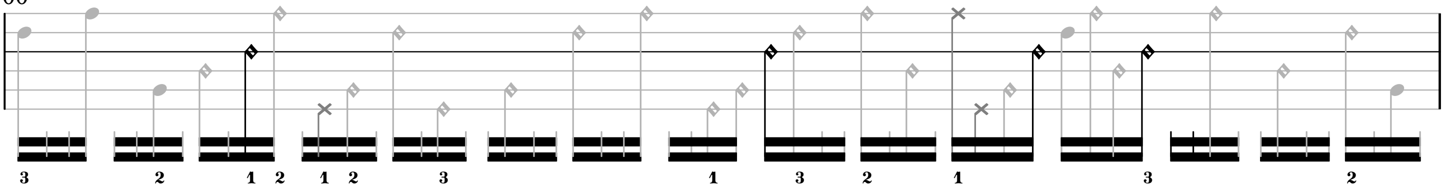
14'36" 14'44"



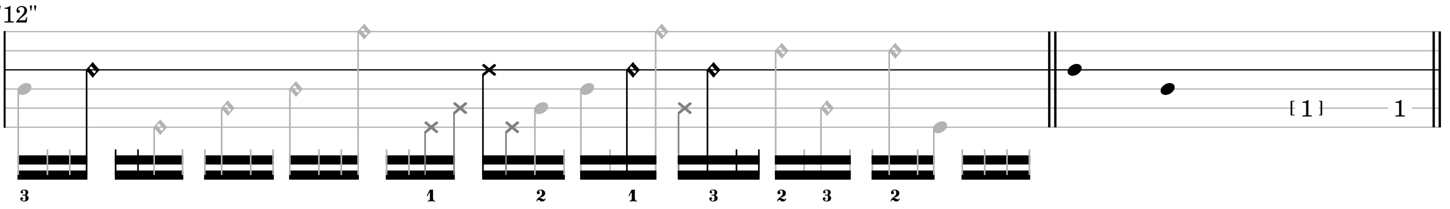
14'48"



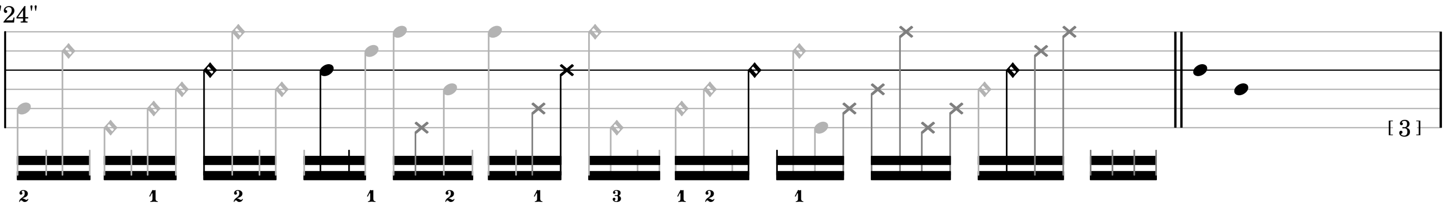
15'00"



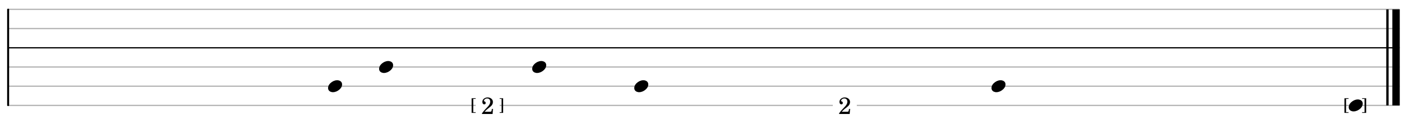
15'12"



15'24"



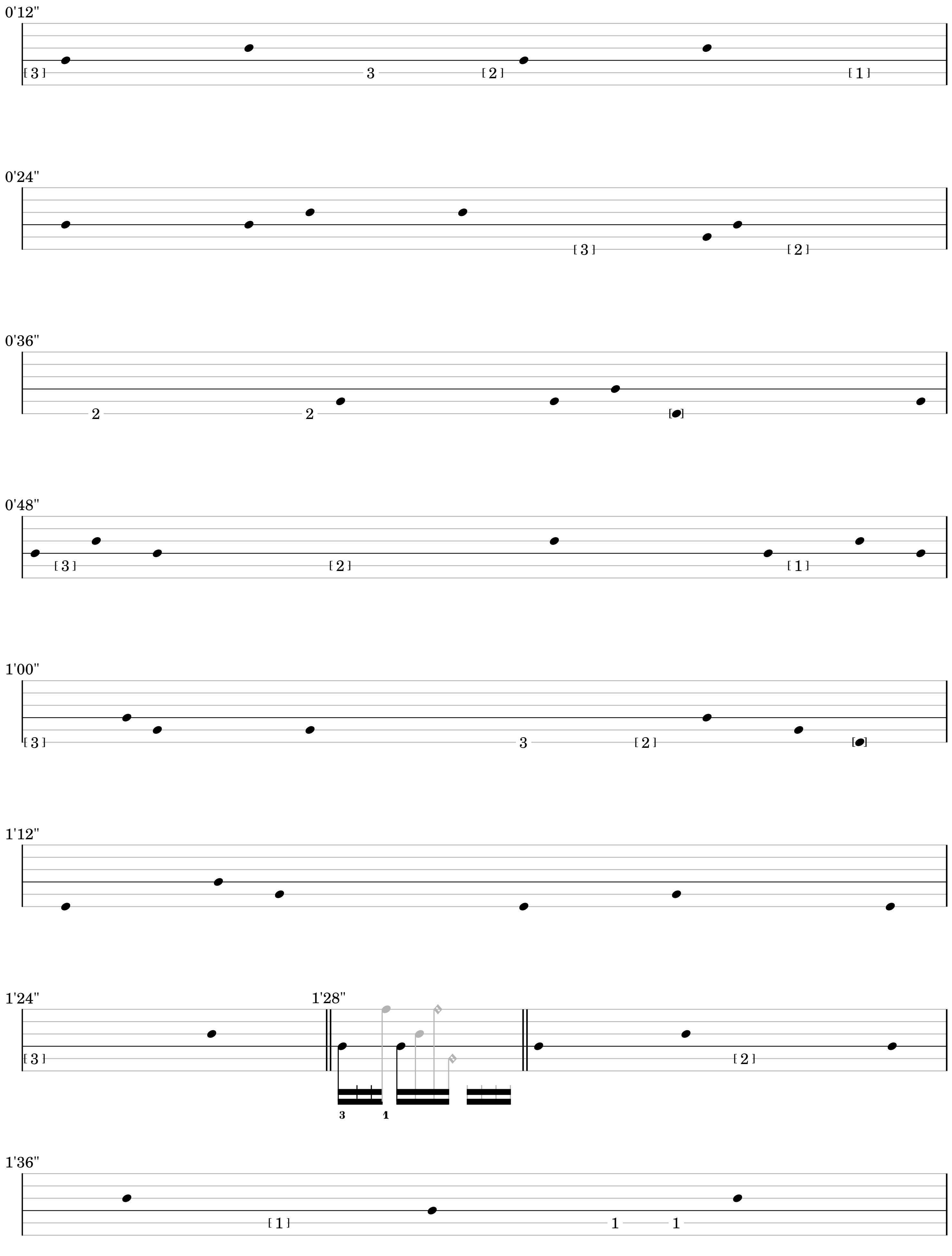
15'36"

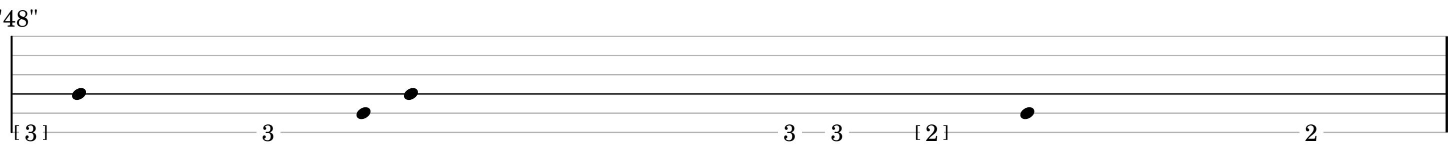


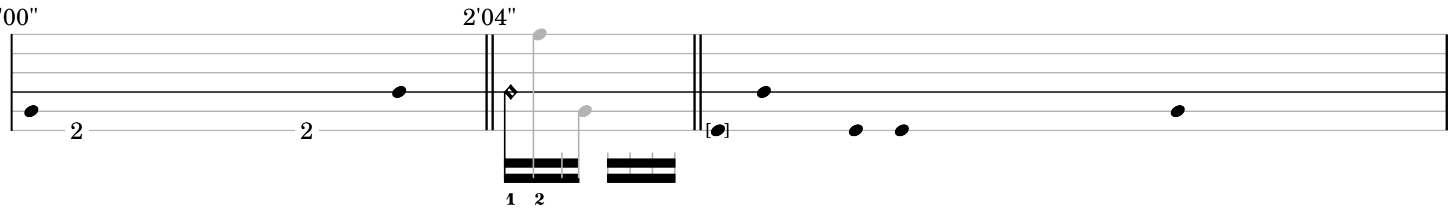
# *ostinato and interrupt*

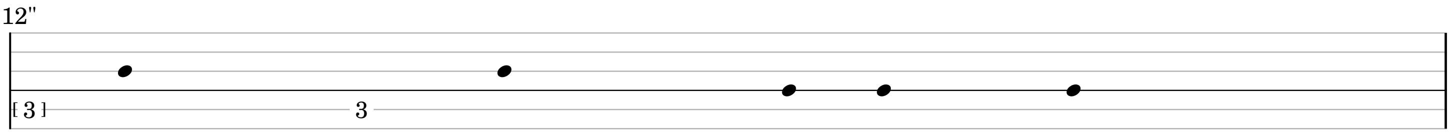
michael winter (mexico city, mx; 2017)
version generated:  2017.12.06

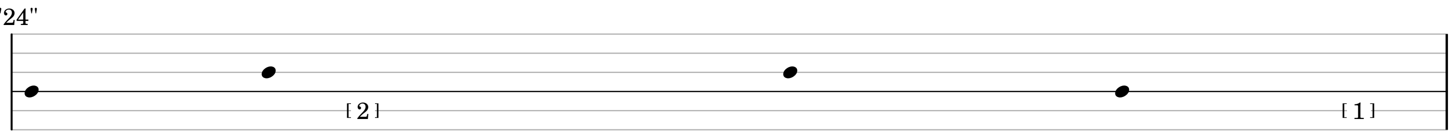
ensemble part 4

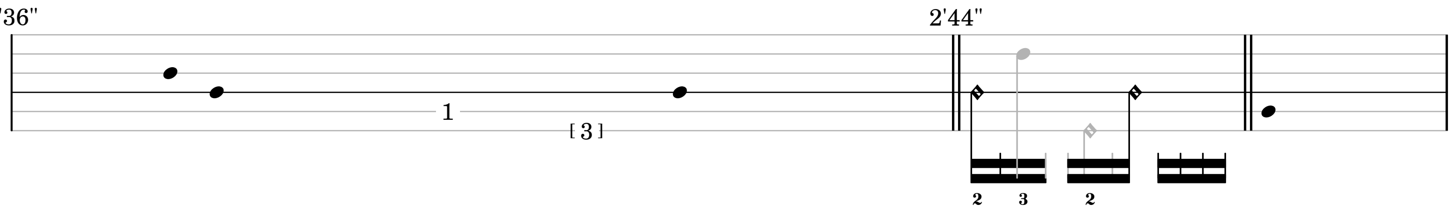


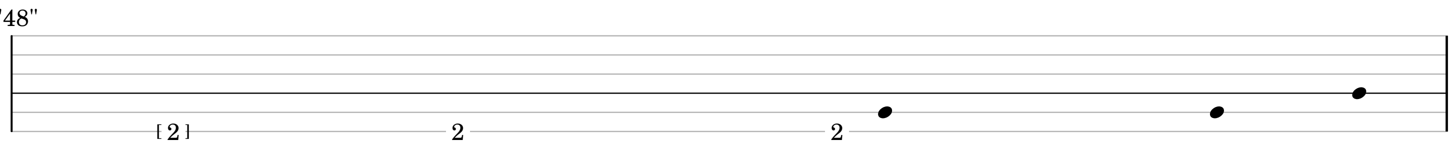

1'48"
[3]
3
3
3
[2]
2

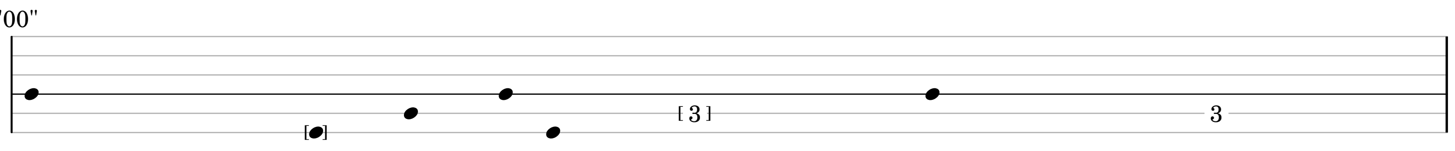

2'00"
2
2
2'04"
1
2

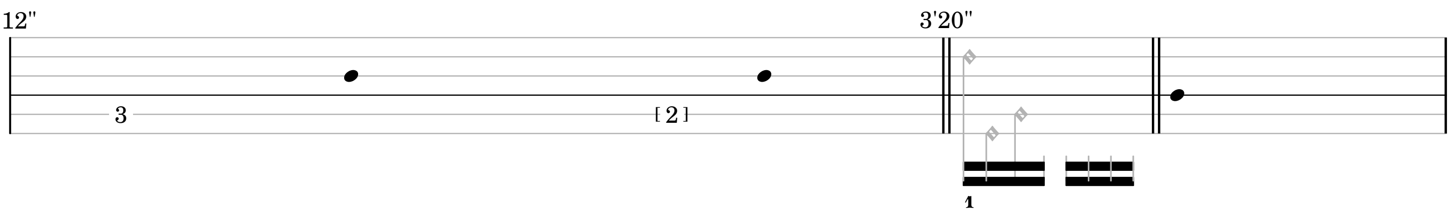

2'12"
[3]
3


2'24"
[2]
[1]

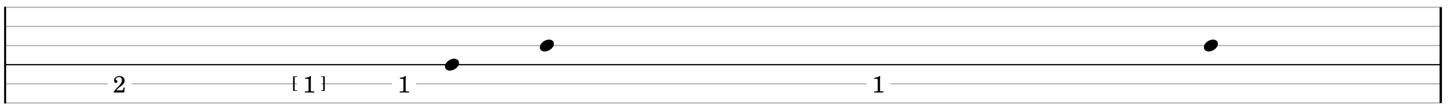

2'36"
1
[3]
2'44"
2
3
2


2'48"
[2]
2
2

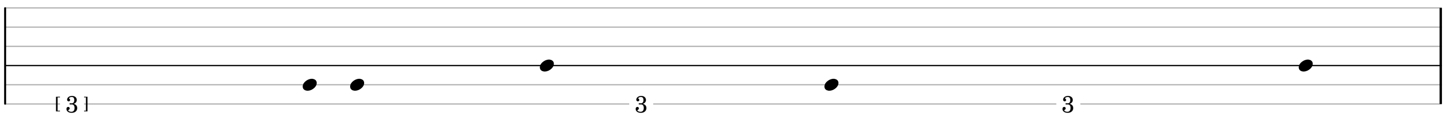

3'00"
[3]
3


3'12"
3
[2]
3'20"
1

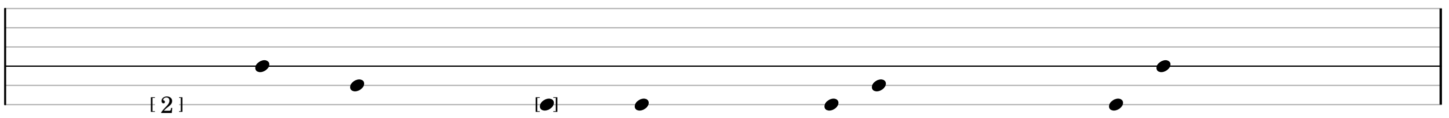
3'24"



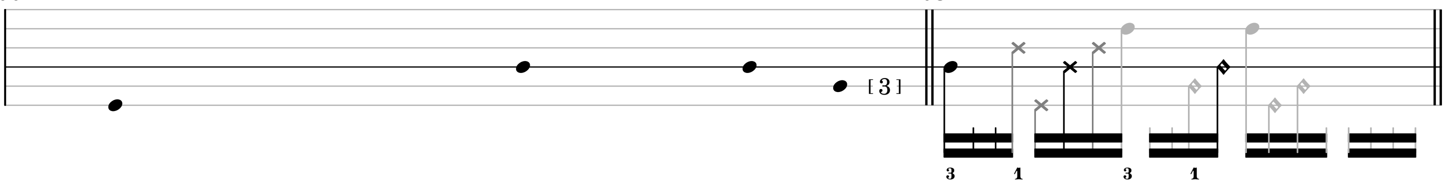
3'36"



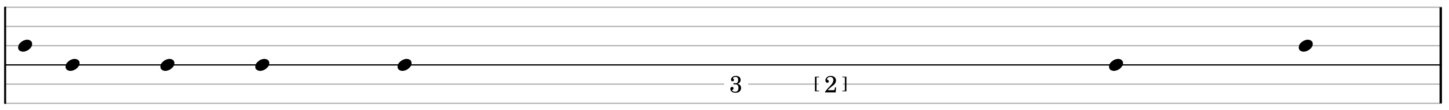
3'48"



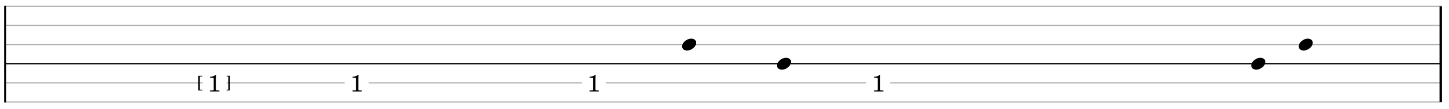
4'00"



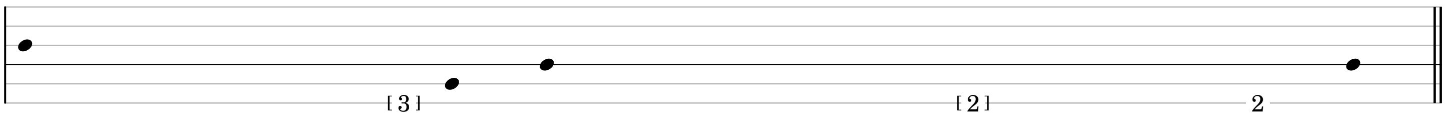
4'12"



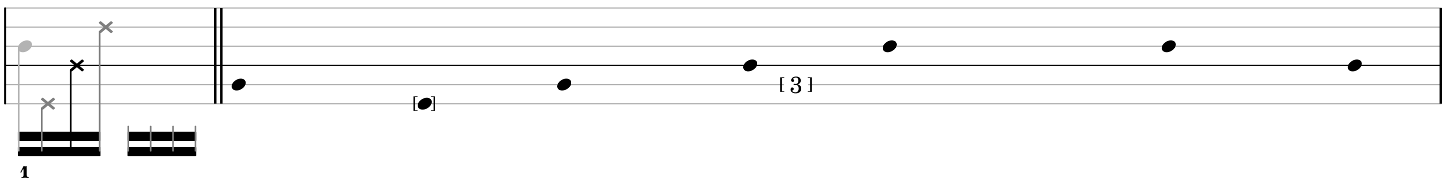
4'24"



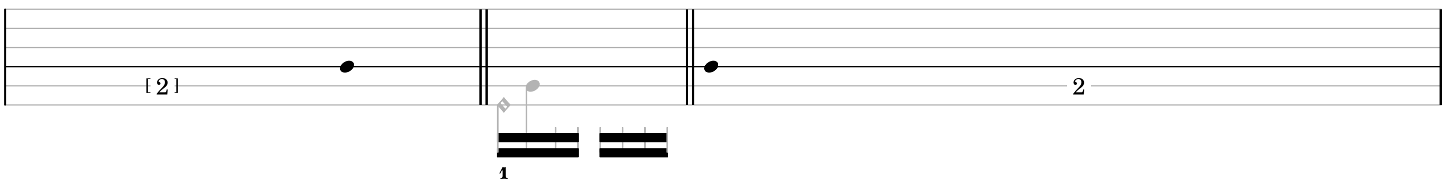
4'36"



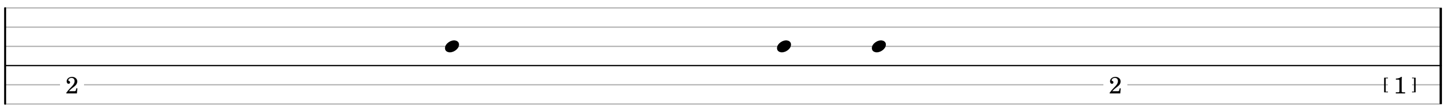
4'48"



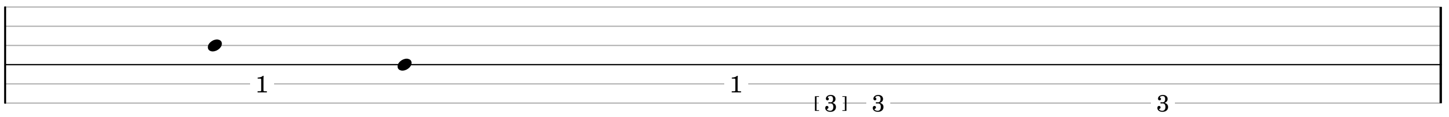
5'00" 5'04"



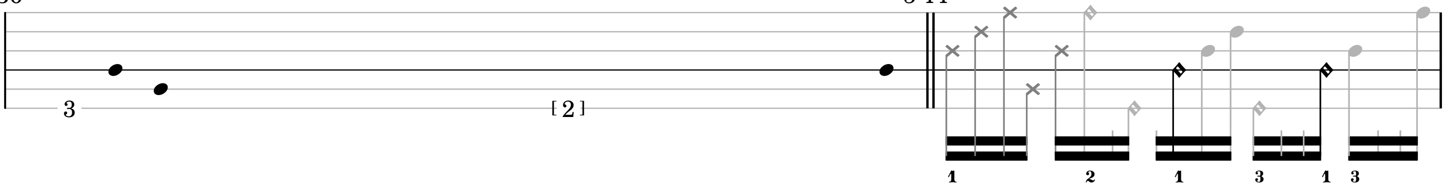
5'12"



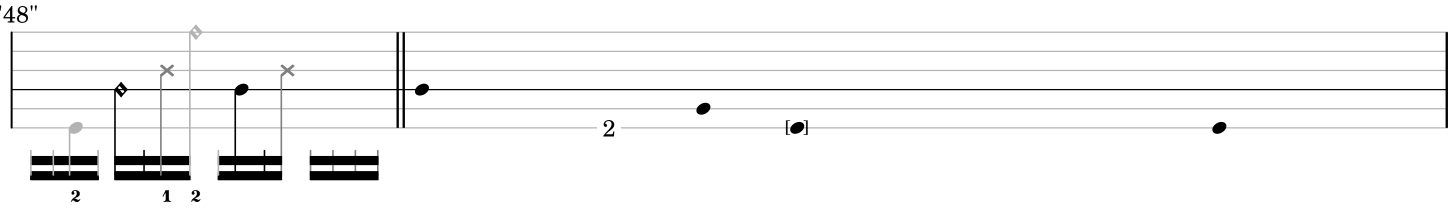
5'24"



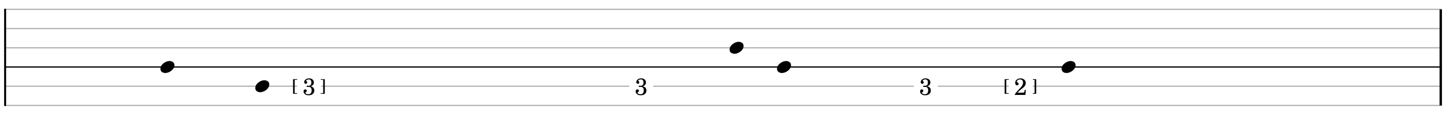
5'36" 5'44"



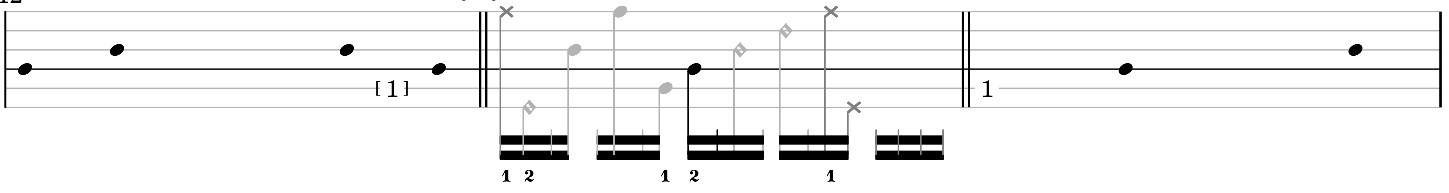
5'48"



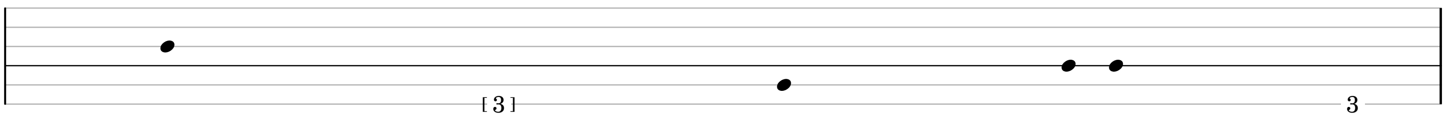
6'00"



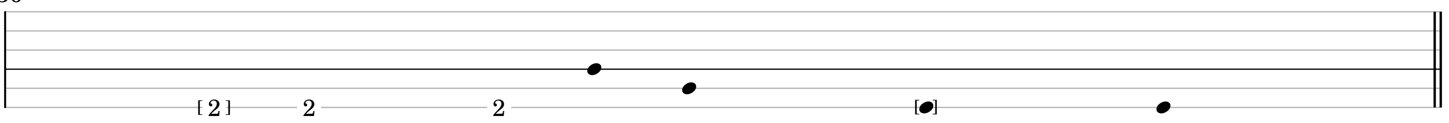
6'12" 6'16"



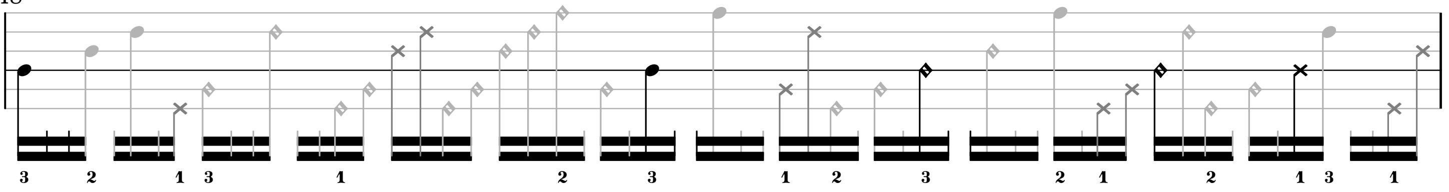
6'24"



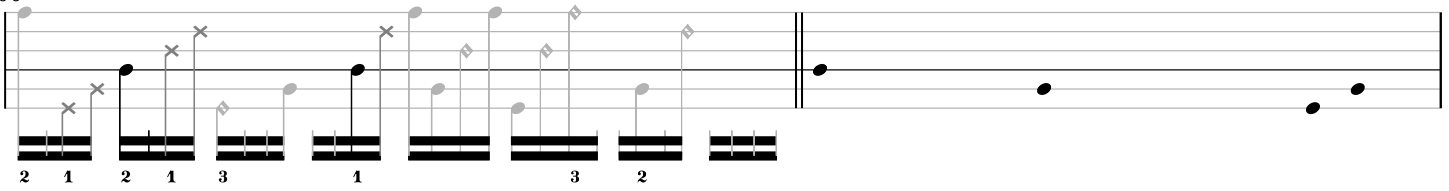
6'36"



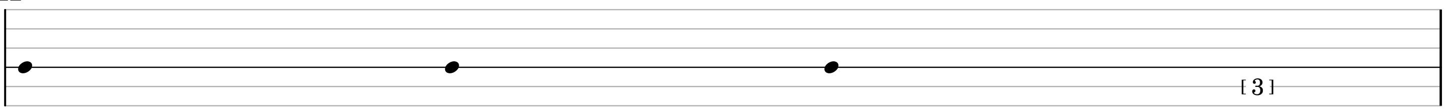
6'48"



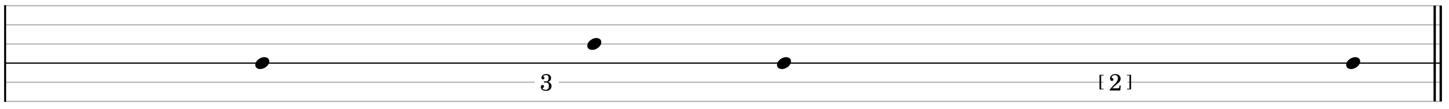
7'00"



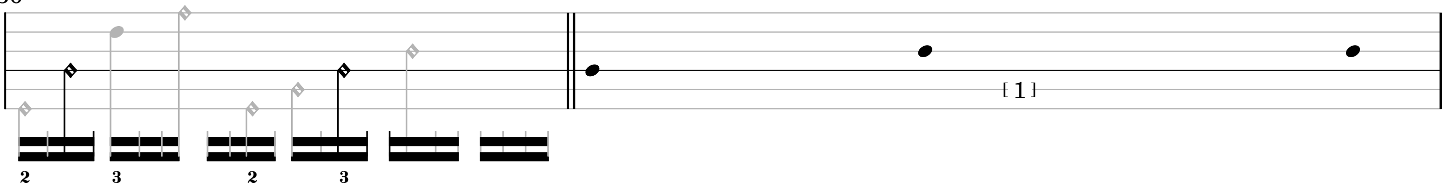
7'12"



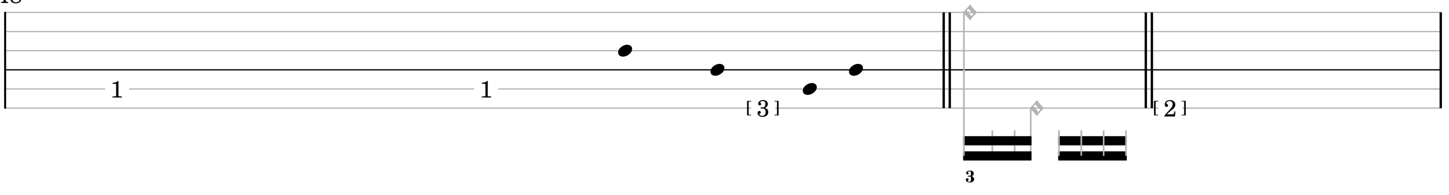
7'24"



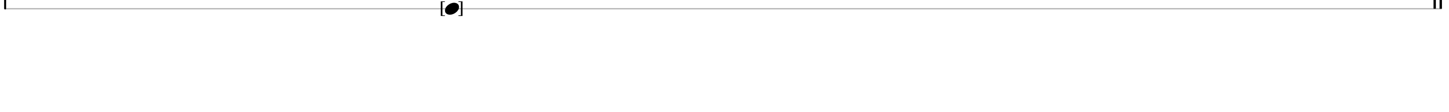
7'36"



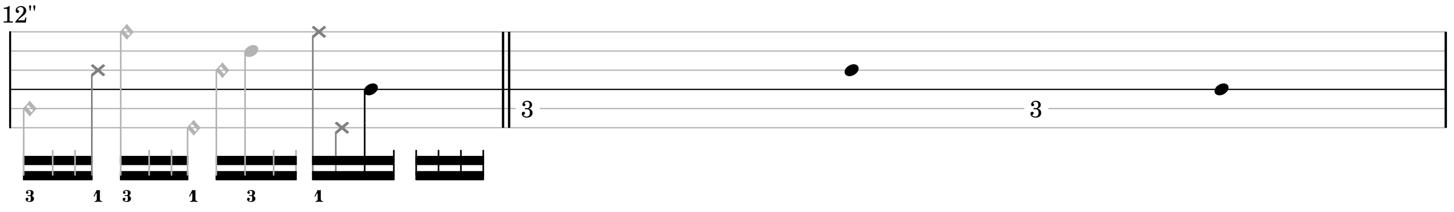
7'48"



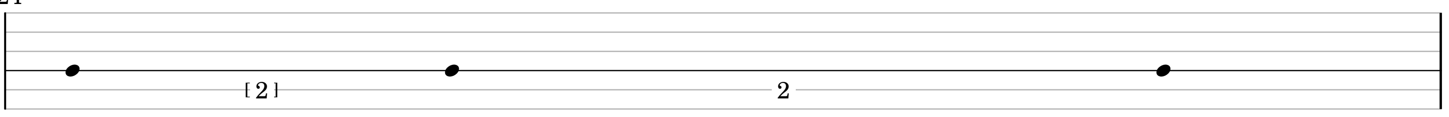
8'00"



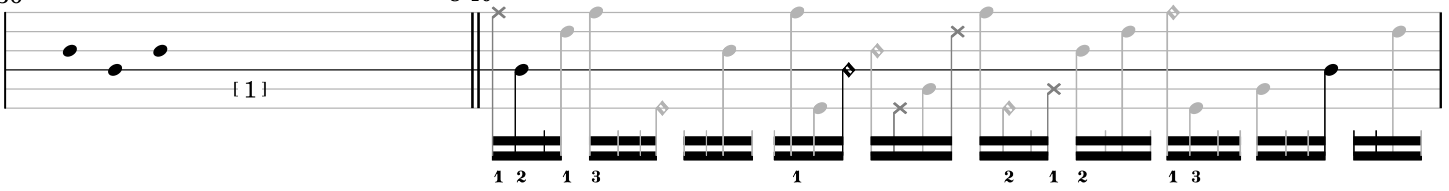
8'12"



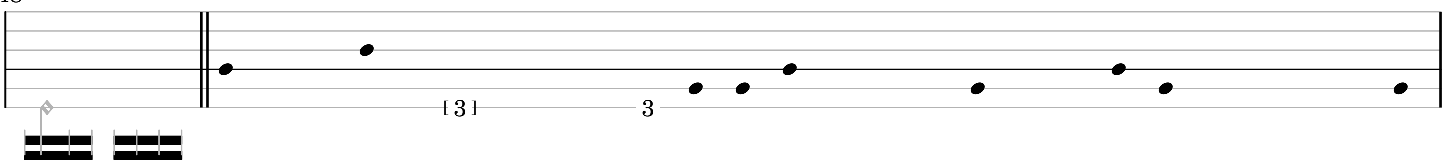
8'24"



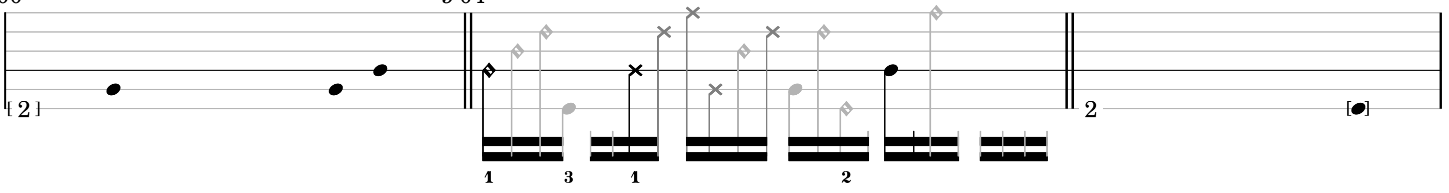
8'36" 8'40"



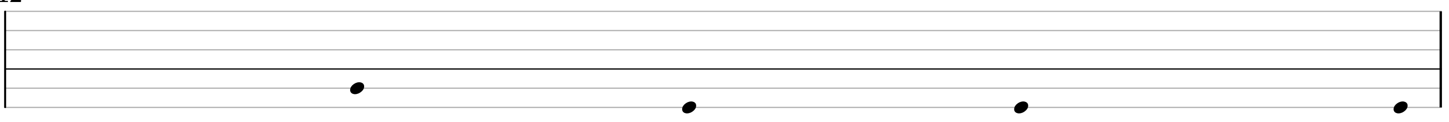
8'48"



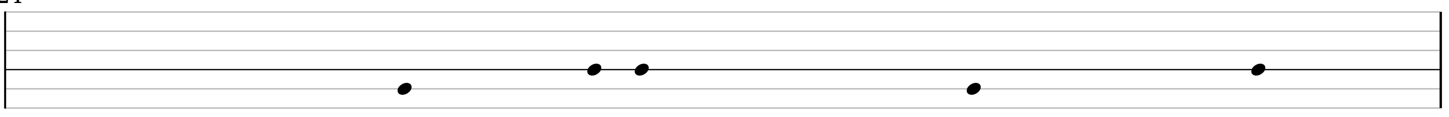
9'00" 9'04"



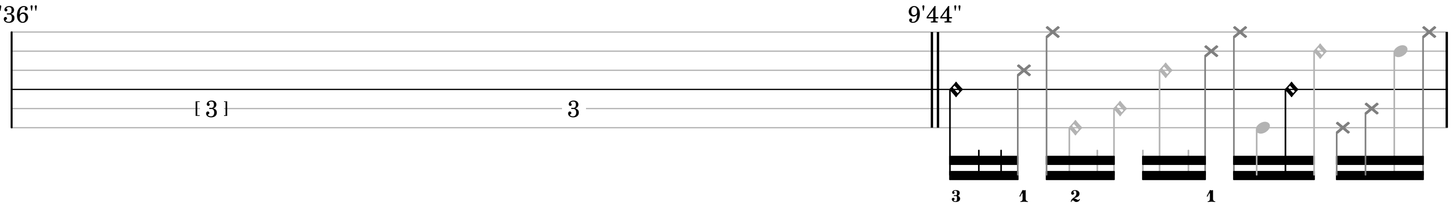
9'12"

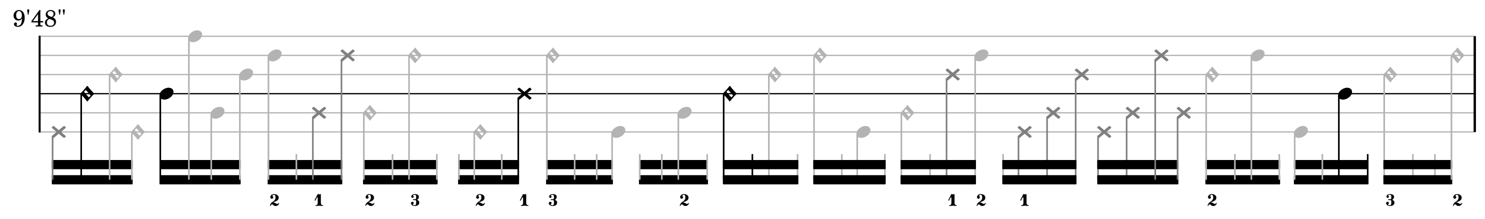


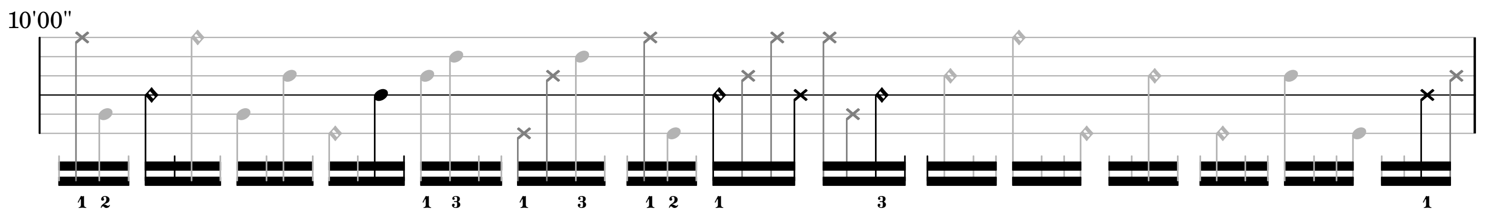
9'24"

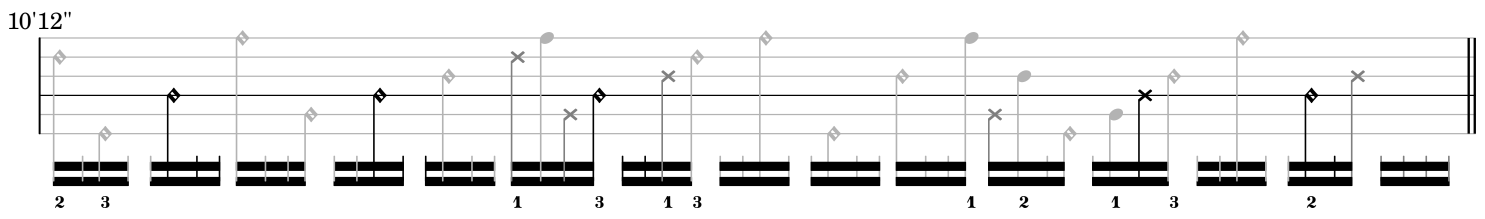


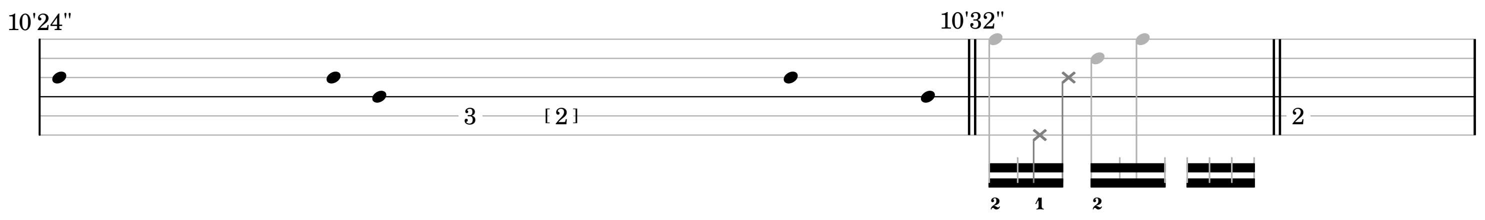
9'36" 9'44"

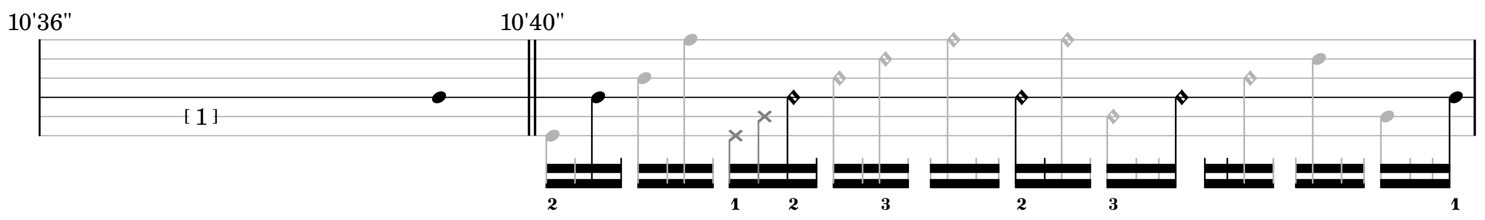


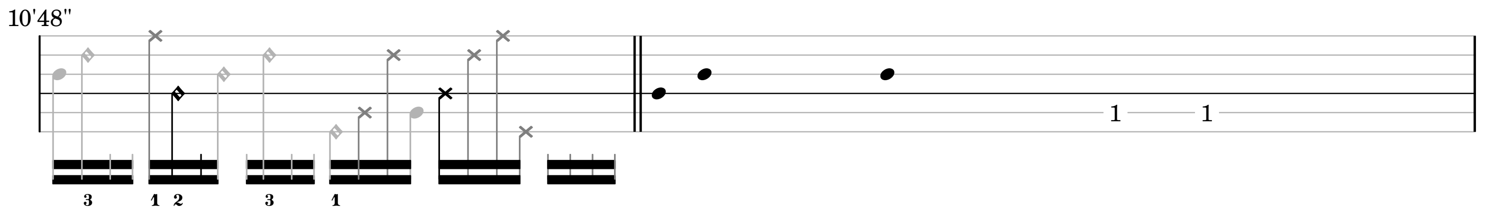

9'48"
2 1 2 3 2 1 3 2 1 2 1 2 3 2

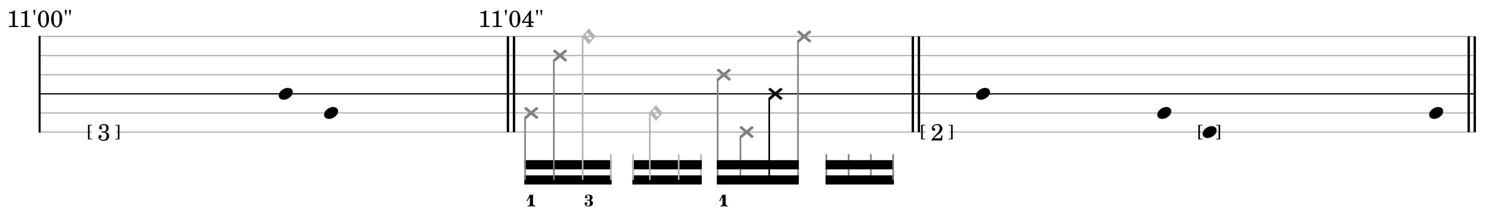

10'00"
1 2 1 3 1 3 1 2 1 3 1

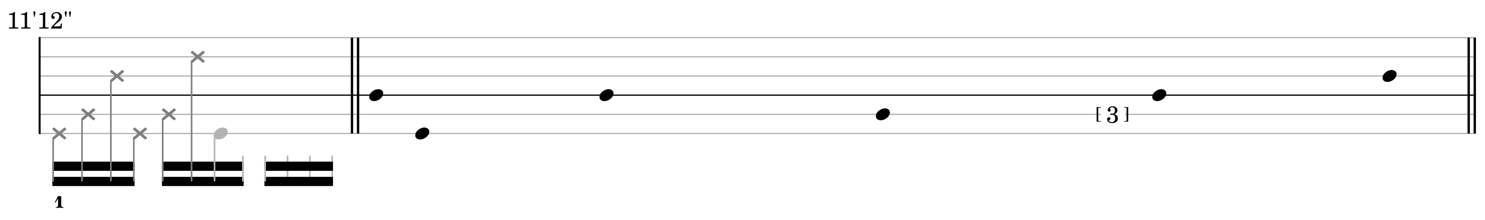

10'12"
2 3 1 3 1 3 1 2 1 3 2


10'24"
3
[ 2 ]
10'32"
2
2 1 2

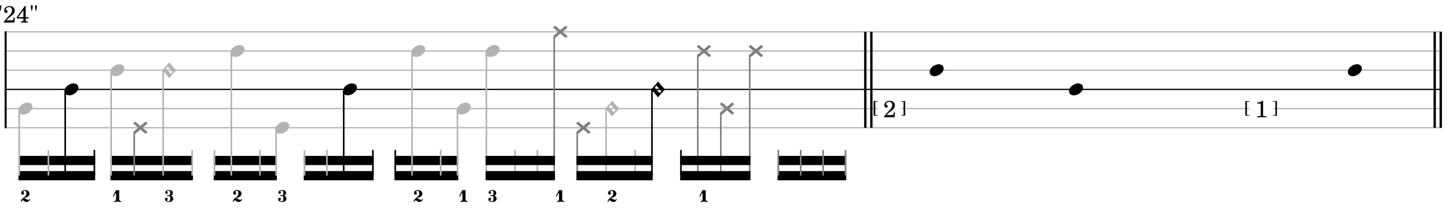

10'36"
[ 1 ]
10'40"
2 1 2 3 2 3 1


10'48"
3 1 2 3 1
1
1


11'00"
[ 3 ]
11'04"
1 3 1
[ 2 ]

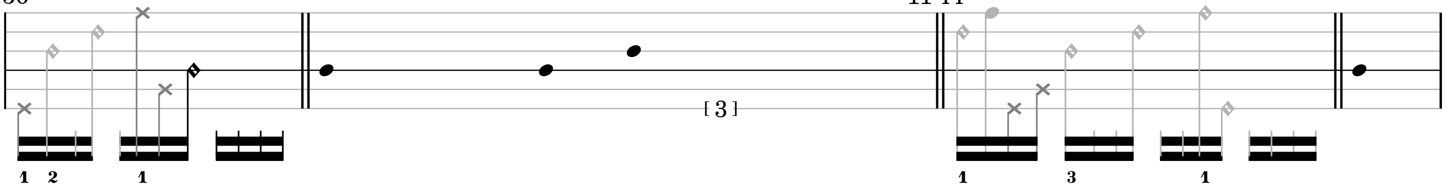

11'12"
1
[ 3 ]

11'24"

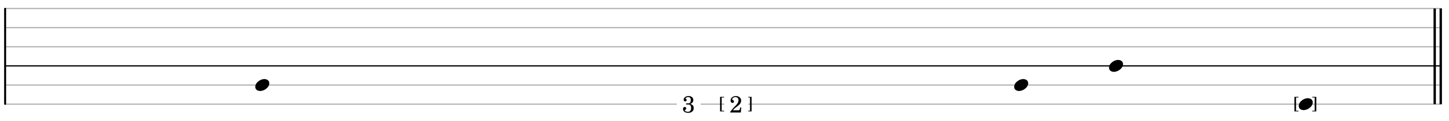


11'36"

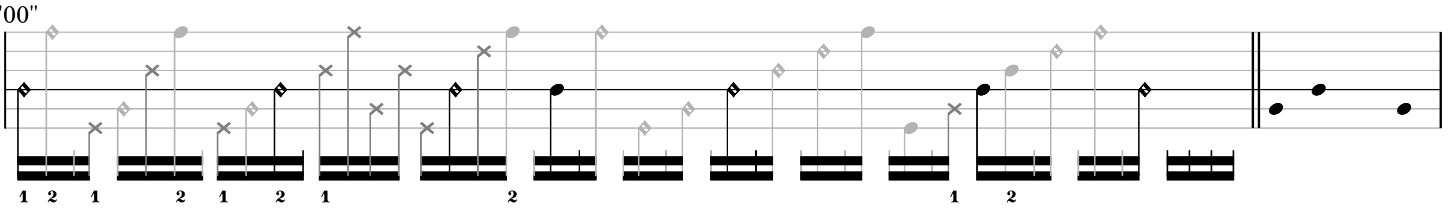
11'44"



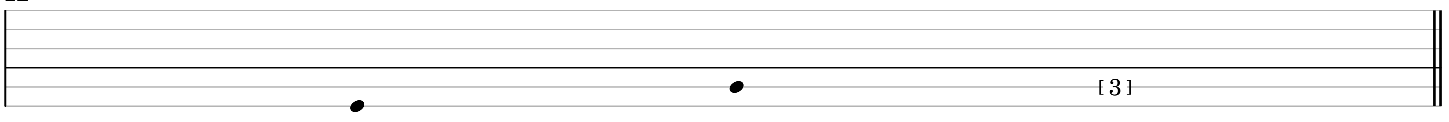
11'48"



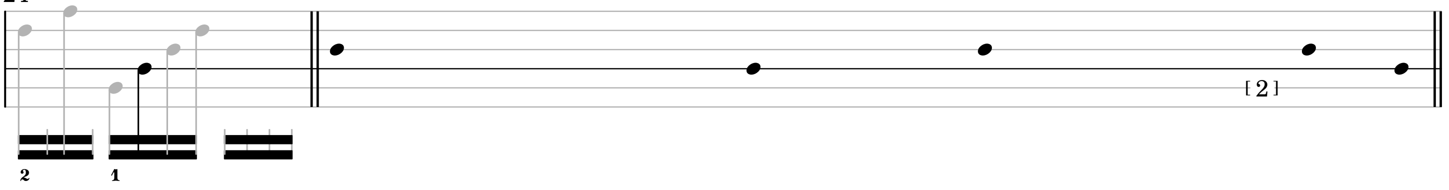
12'00"



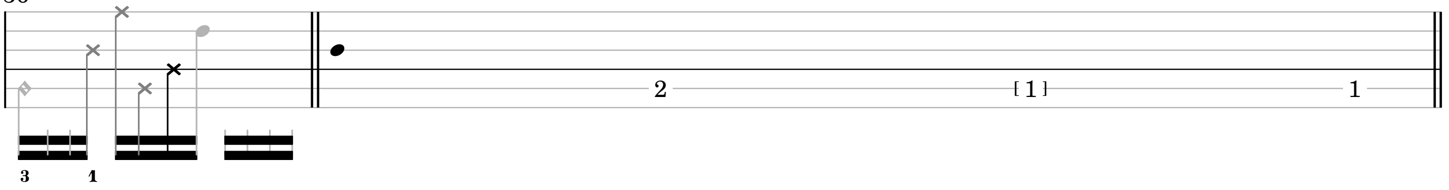
12'12"



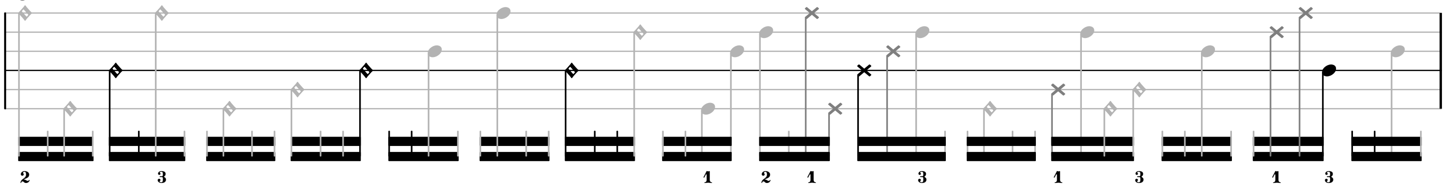
12'24"



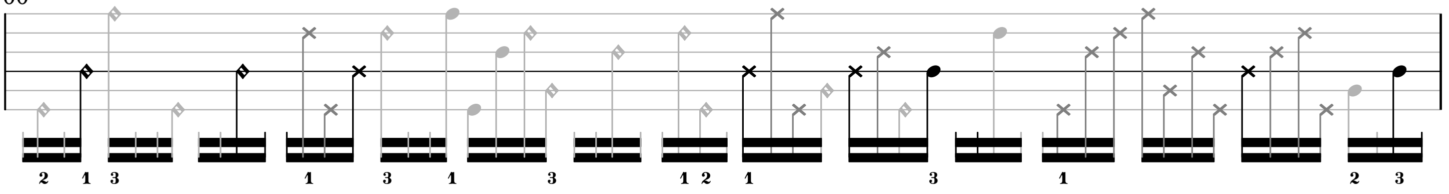
12'36"



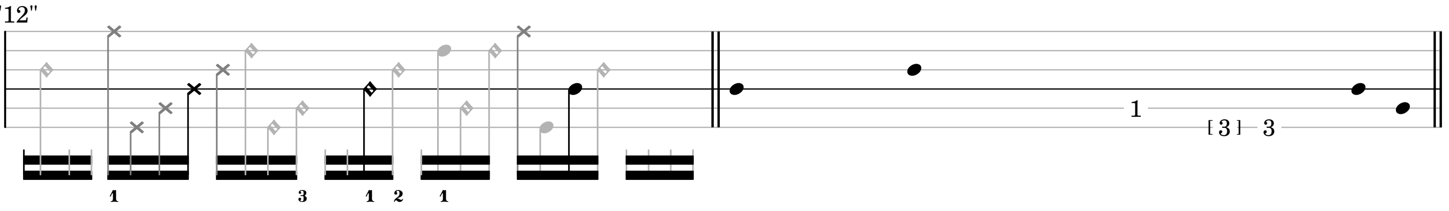
12'48"



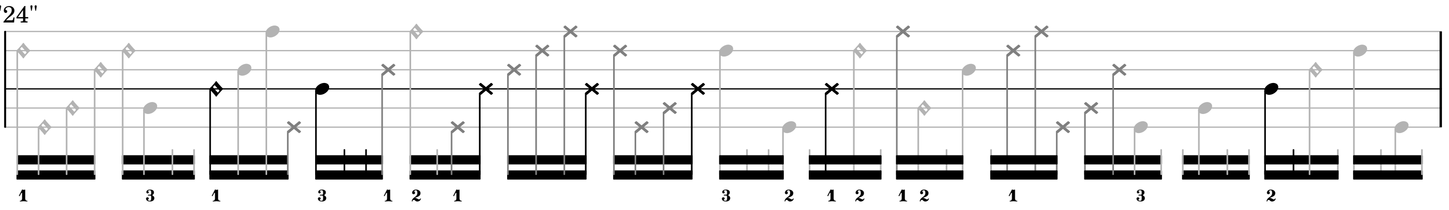
13'00"



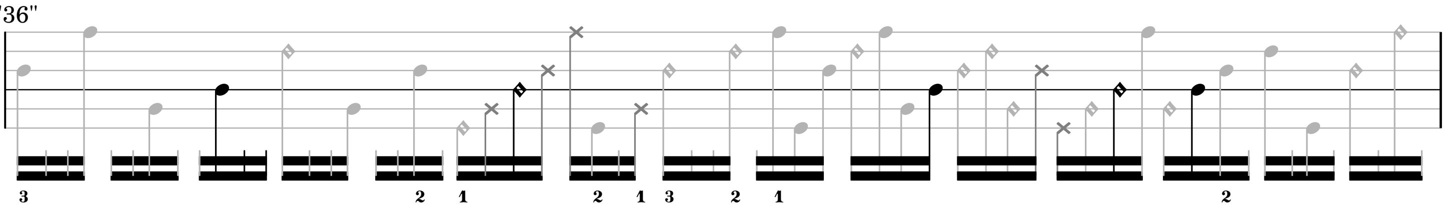
13'12"



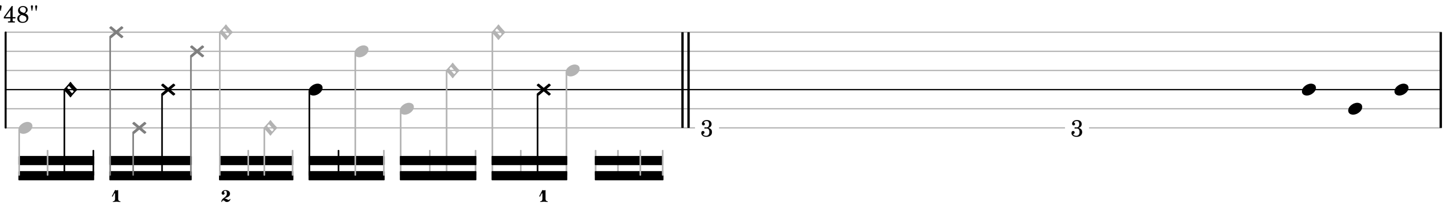
13'24"



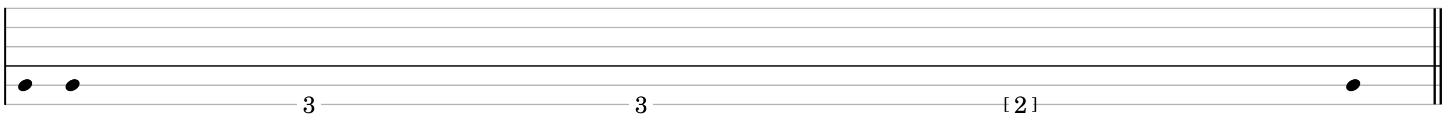
13'36"



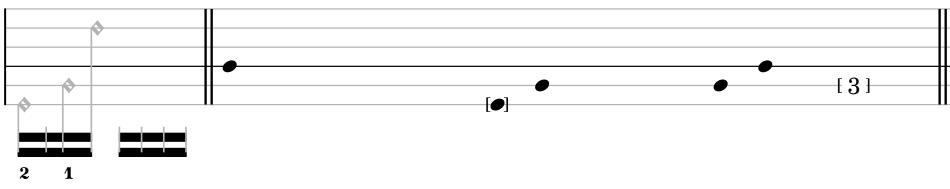
13'48"



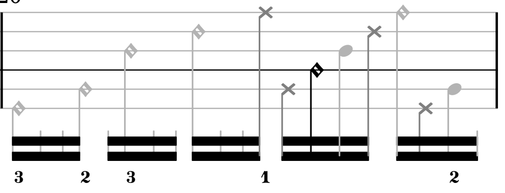
14'00"



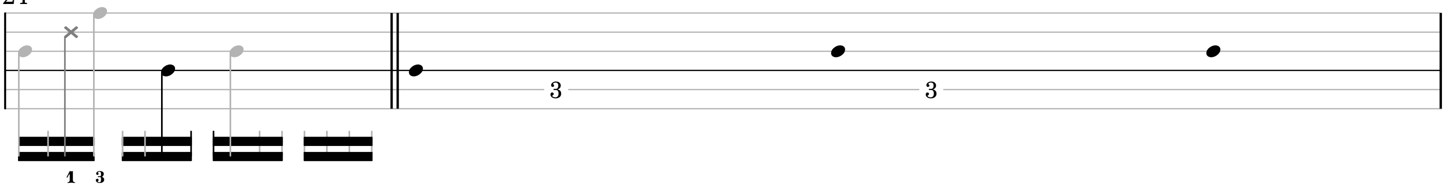
14'12"



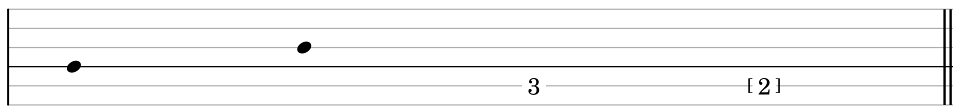
14'20"



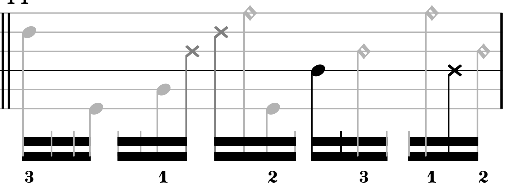
14'24"



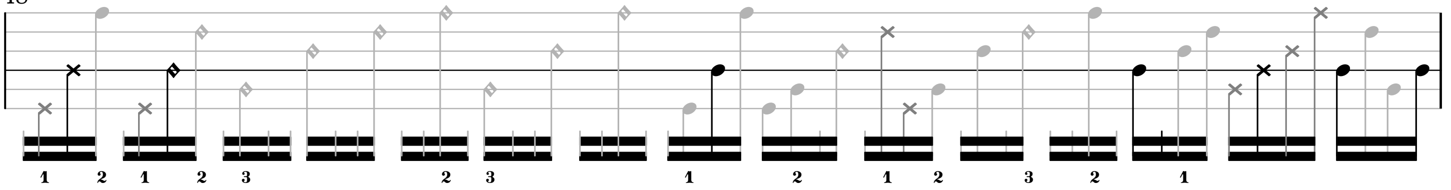
14'36"



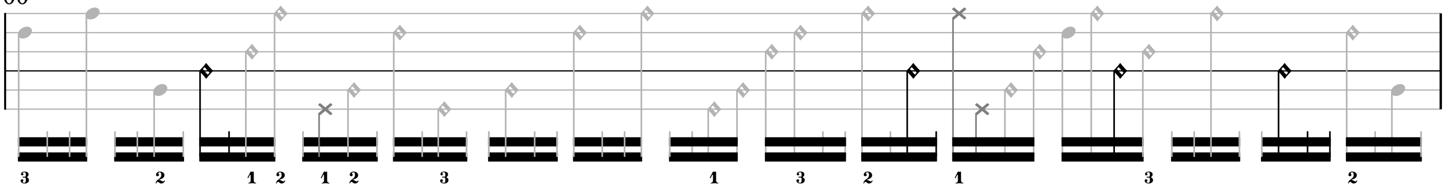
14'44"



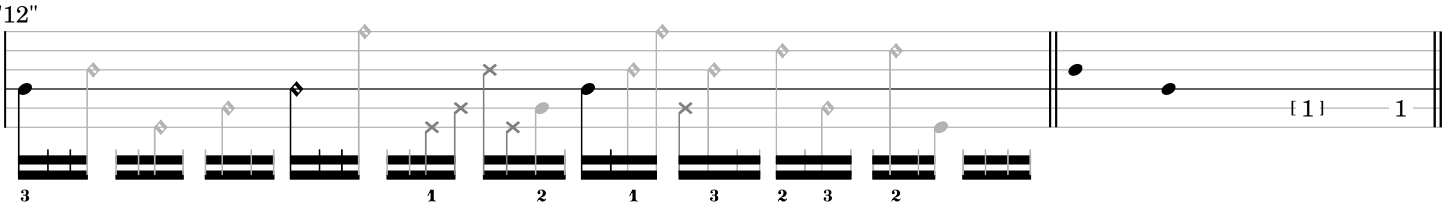
14'48"



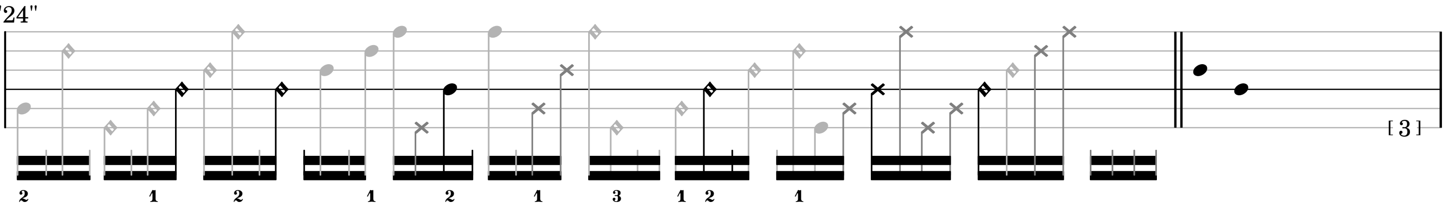
15'00"



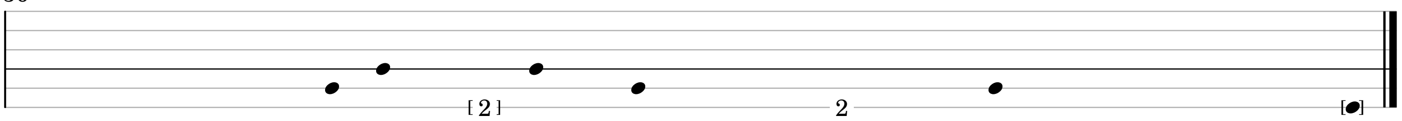
15'12"



15'24"



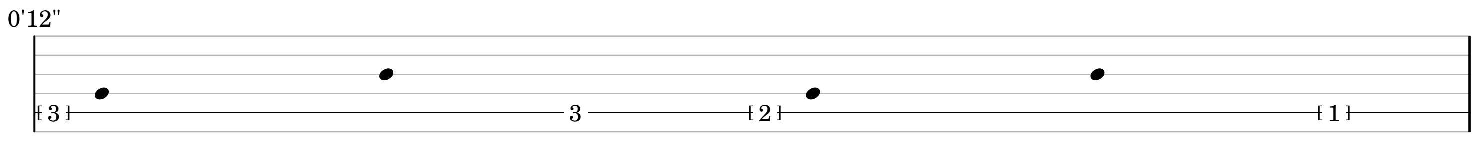
15'36"

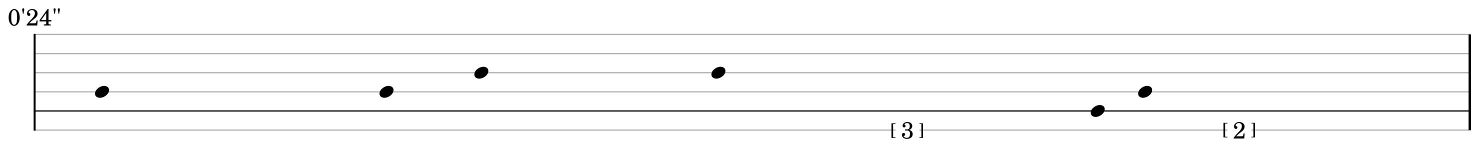


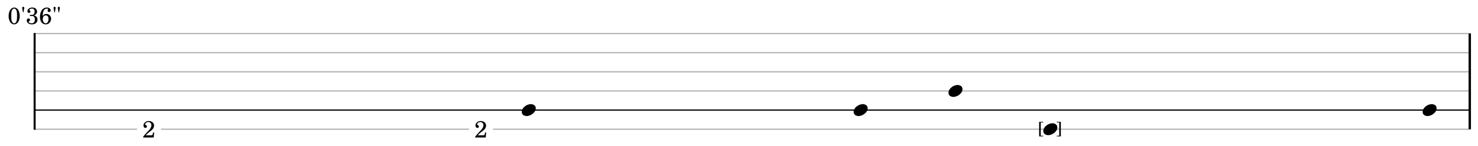
# *ostinato and interrupt*

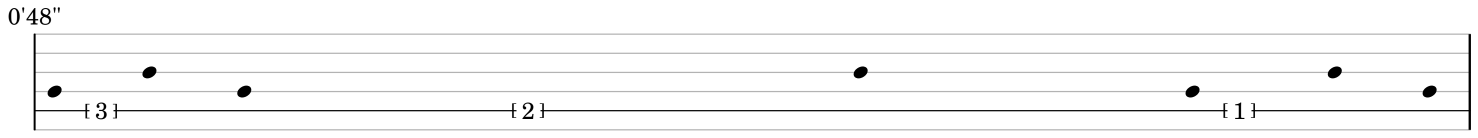
michael winter (mexico city, mx; 2017)
version generated: 2017.12.06

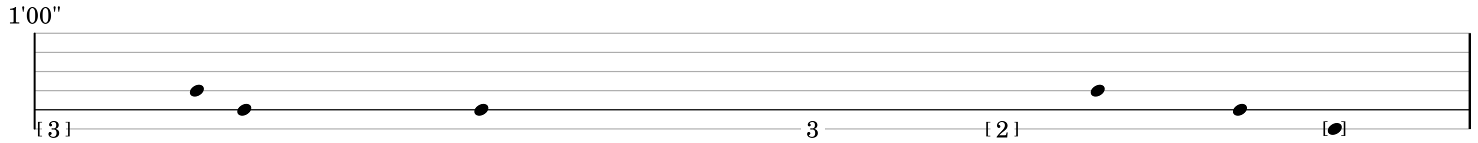
ensemble part 5

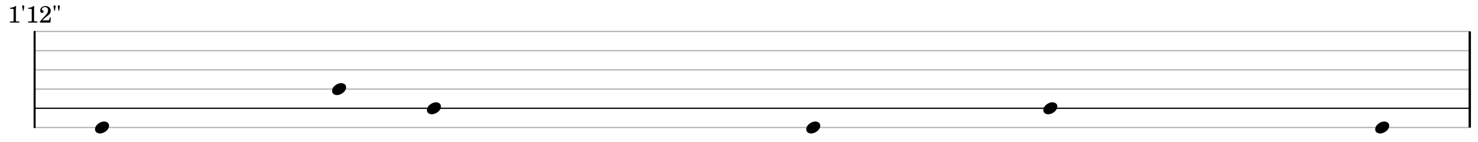


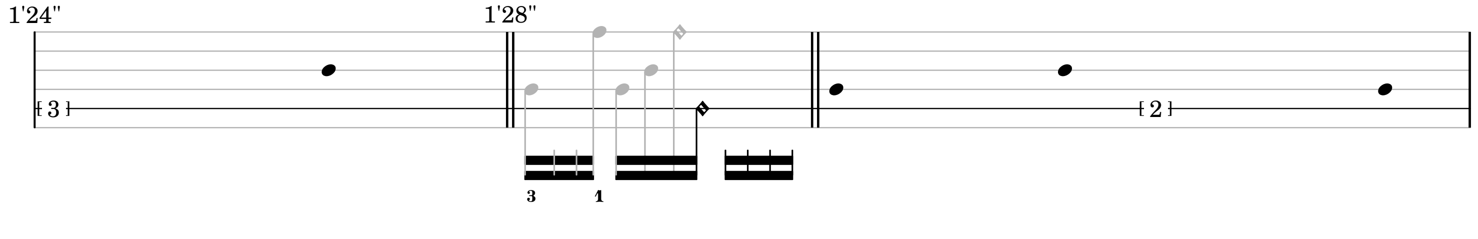


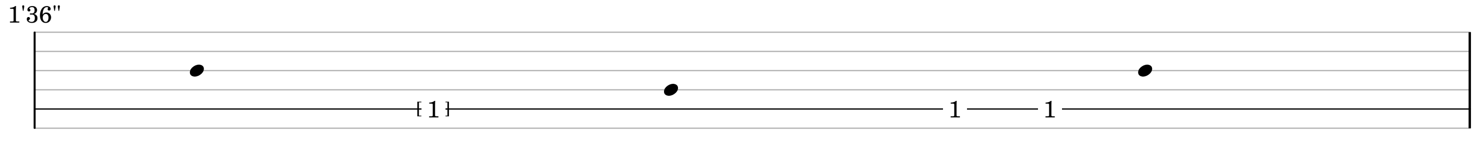


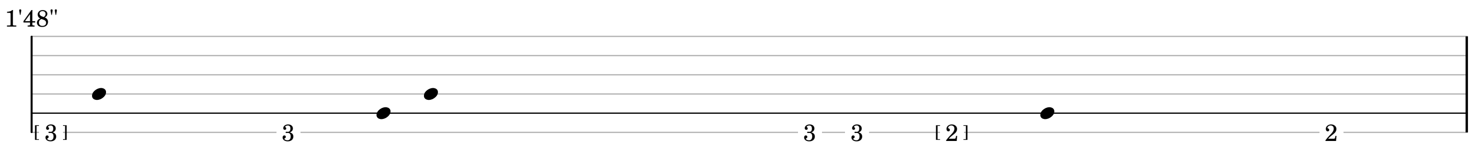




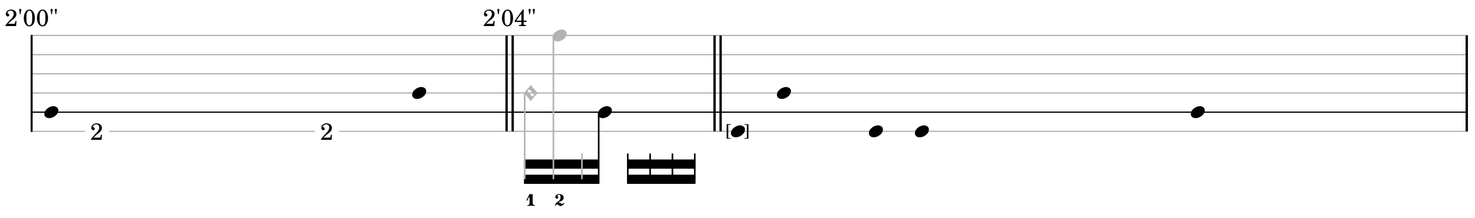




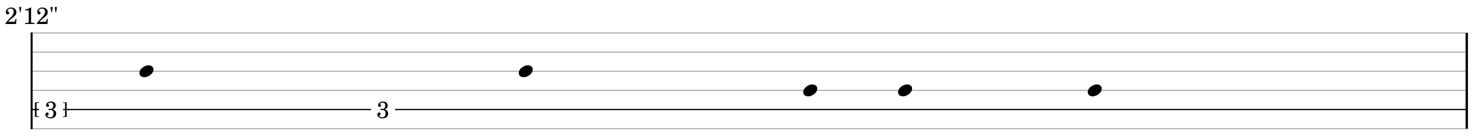




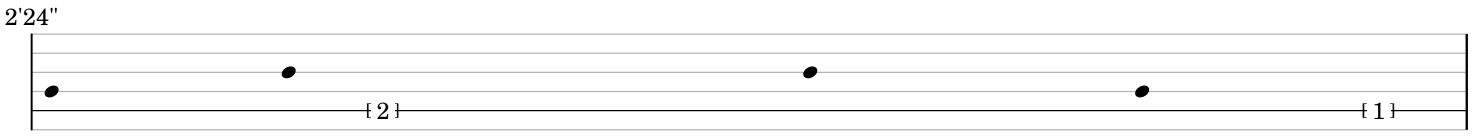
1'48"
[3]
3
3
3
[2]
2



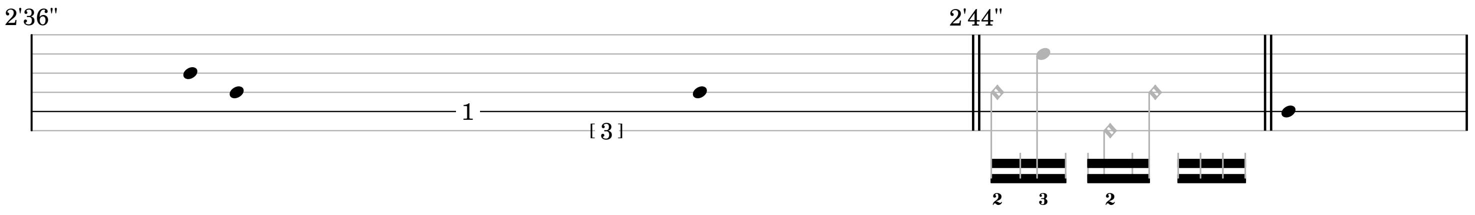
2'00"
2
2
2'04"
1
2



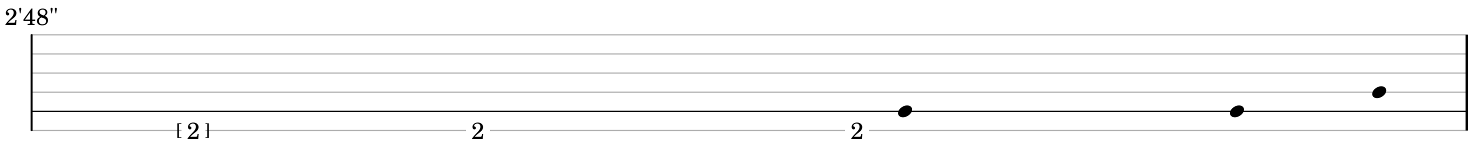
2'12"
[3]
3



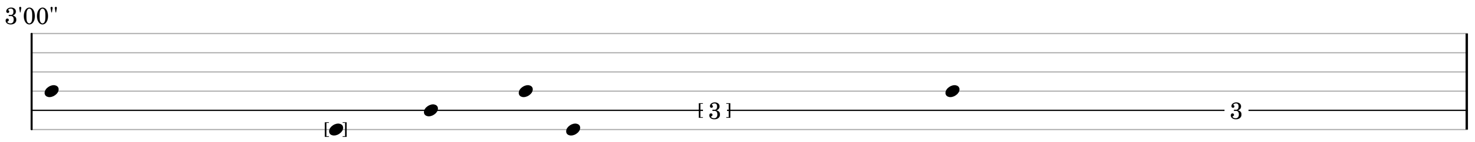
2'24"
[2]
[1]



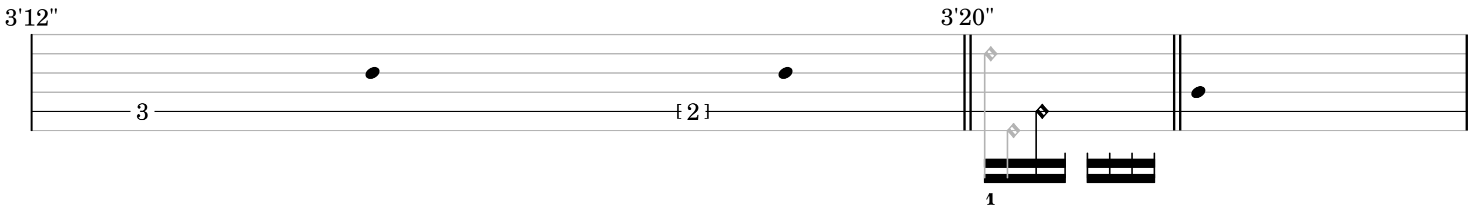
2'36"
1
[3]
2'44"
2
3
2



2'48"
[2]
2
2

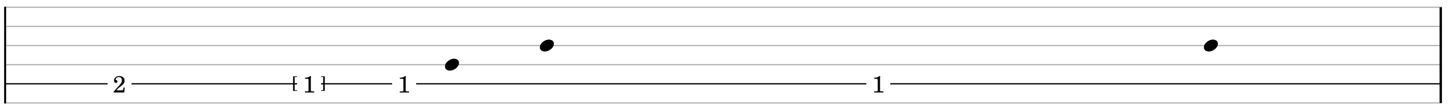


3'00"
[3]
3

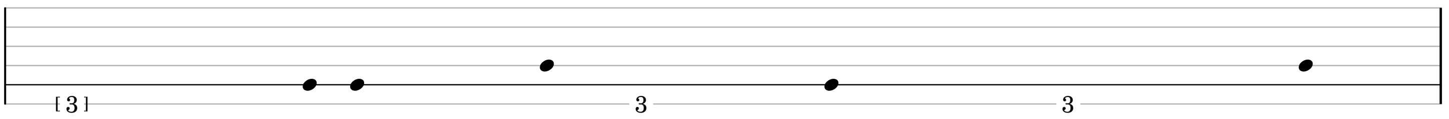


3'12"
3
[2]
3'20"
1

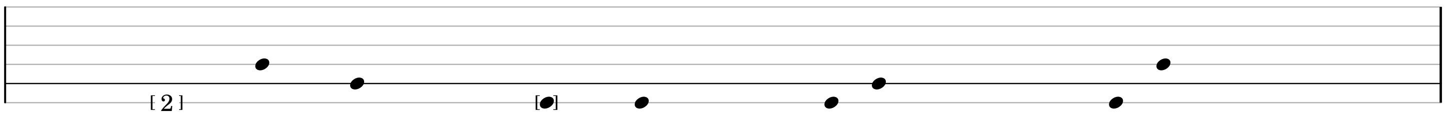
3'24"



3'36"

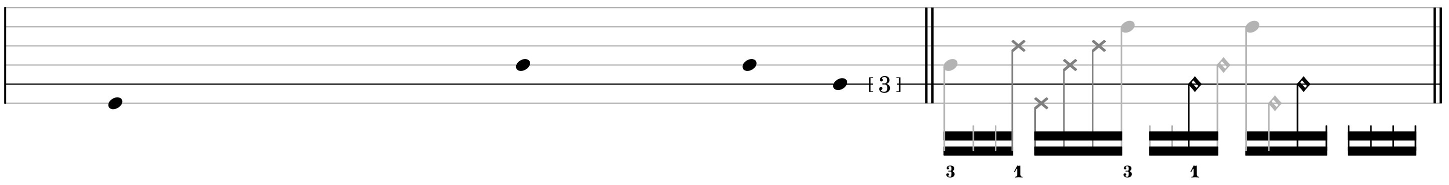


3'48"

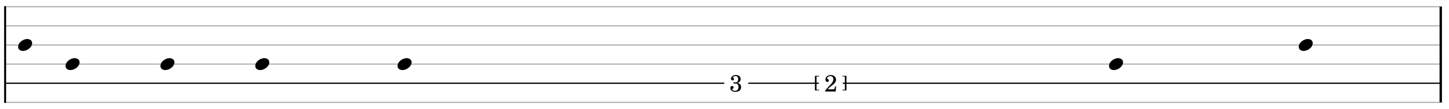


4'00"

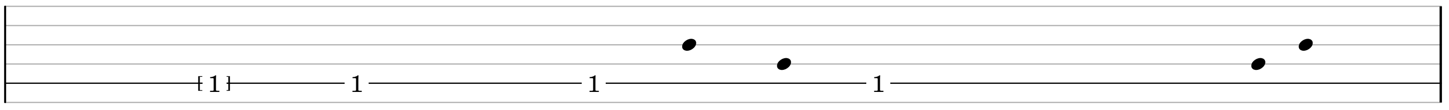
4'08"



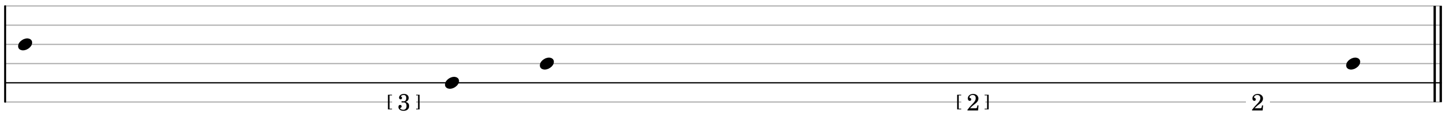
4'12"



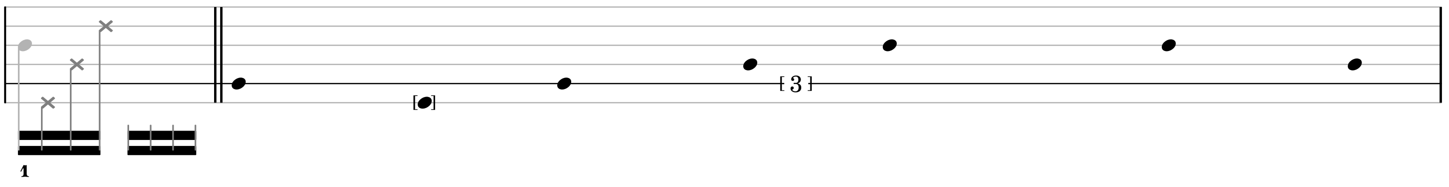
4'24"



4'36"

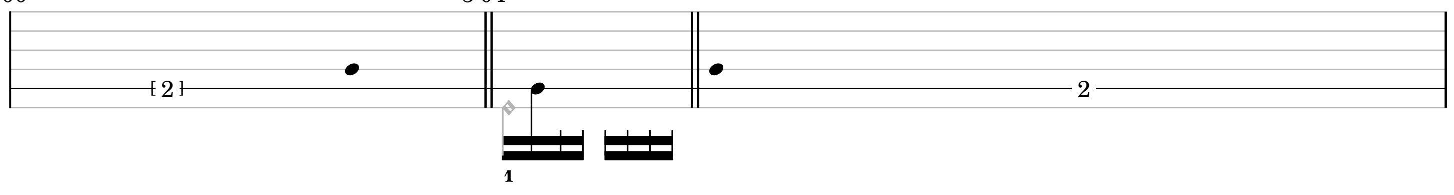


4'48"

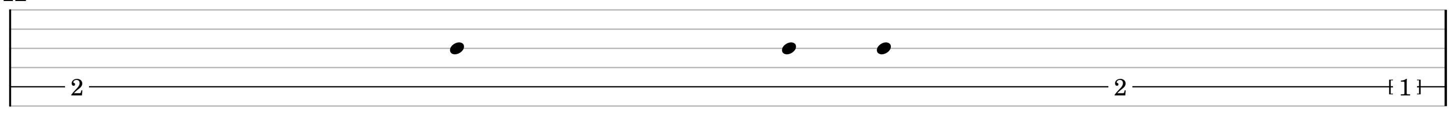


5'00"

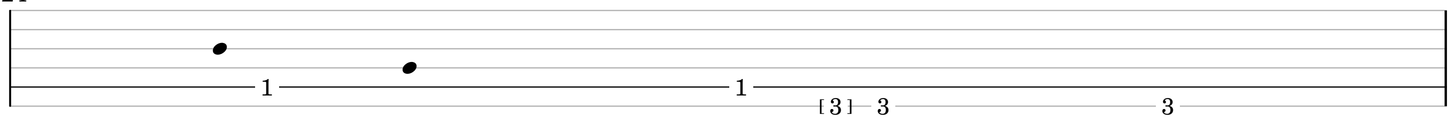
5'04"



5'12"

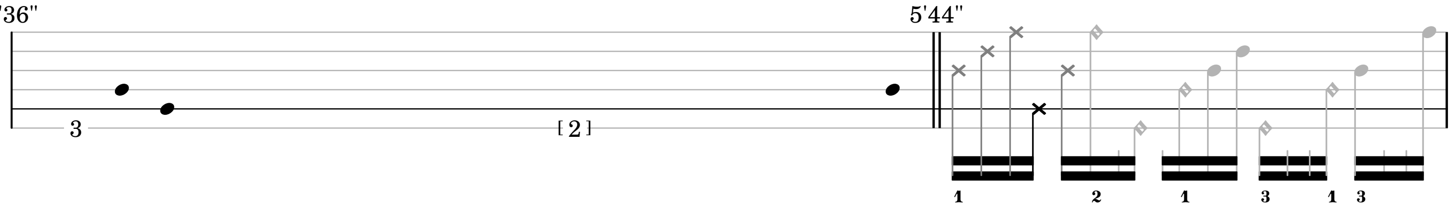


5'24"

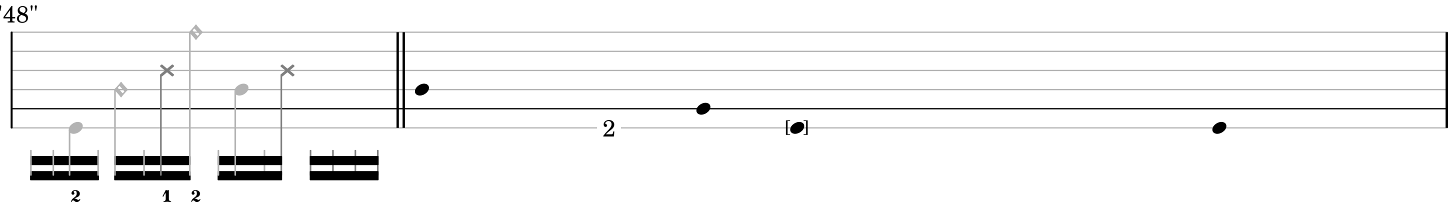


5'36"

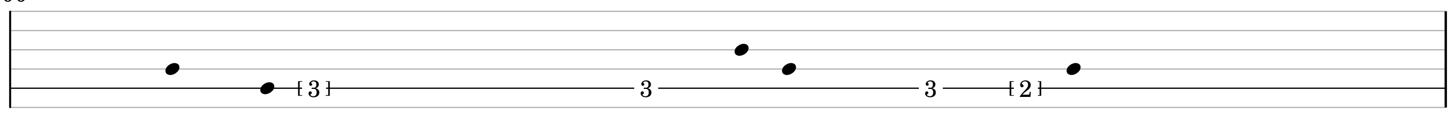
5'44"



5'48"

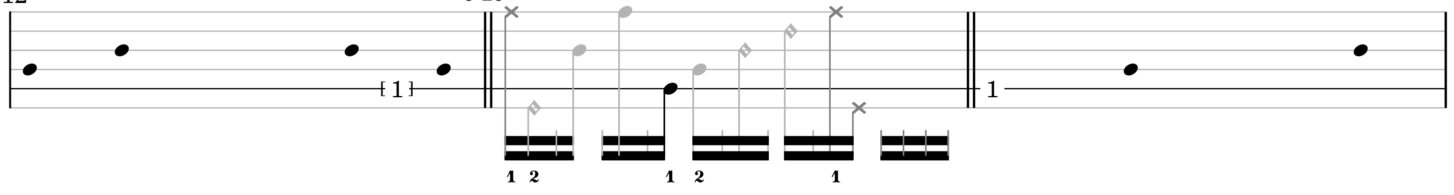


6'00"

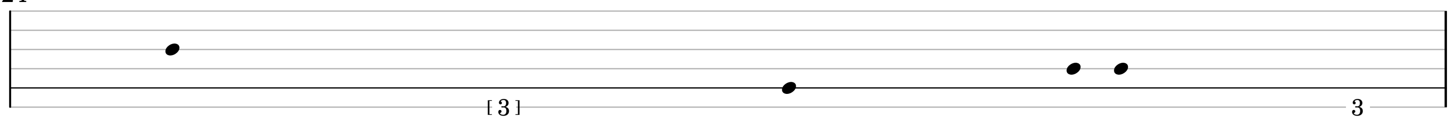


6'12"

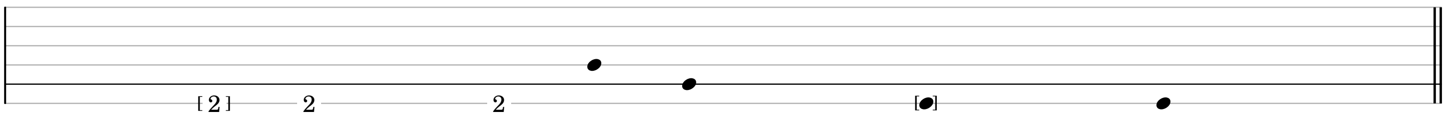
6'16"



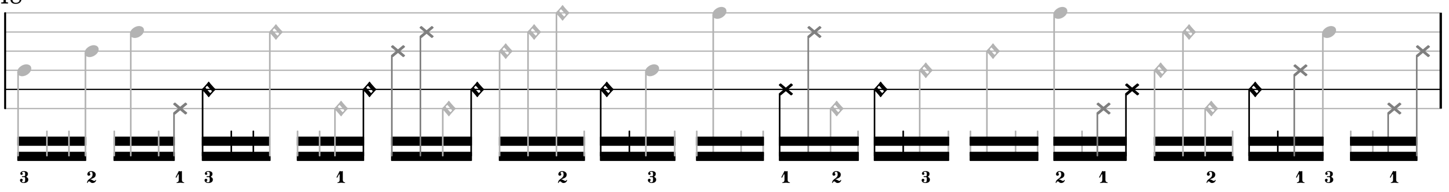
6'24"



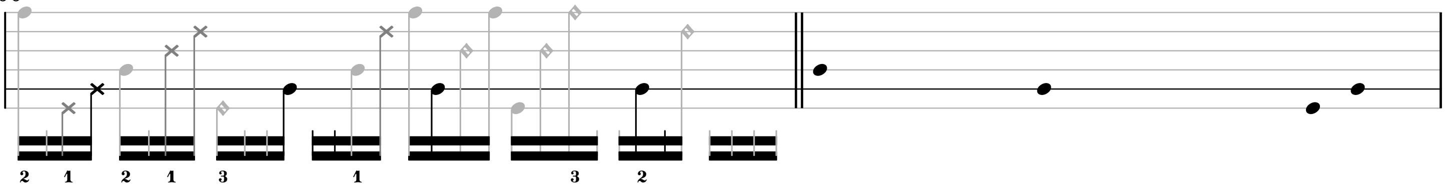
6'36"



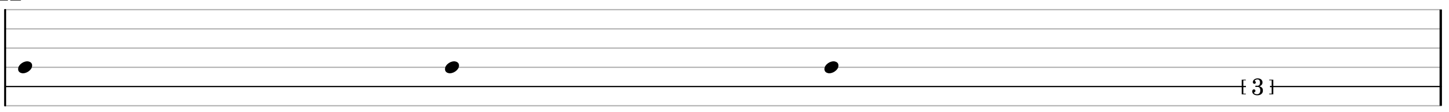
6'48"



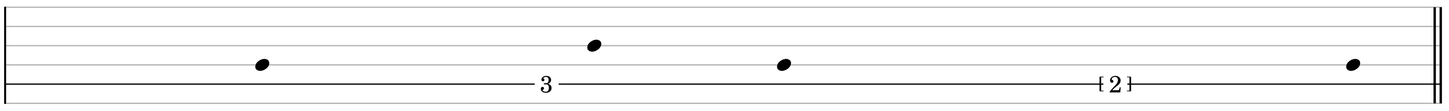
7'00"



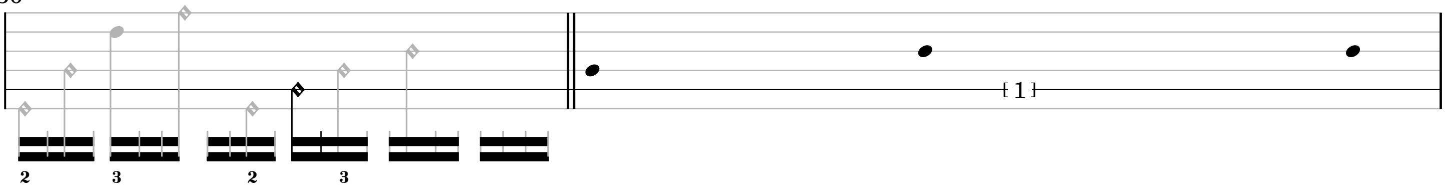
7'12"



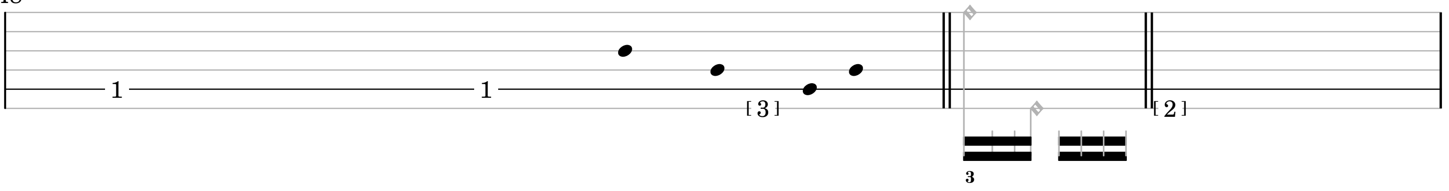
7'24"



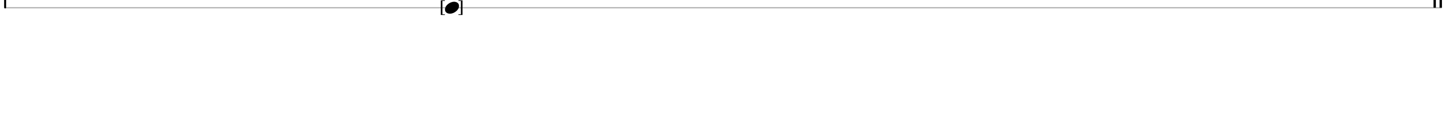
7'36"



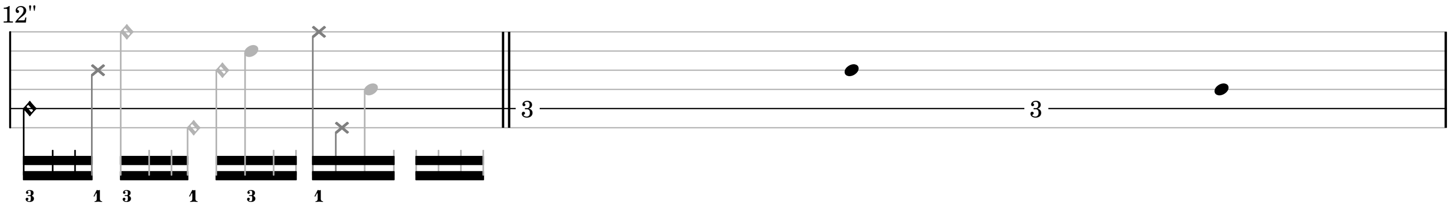
7'48"



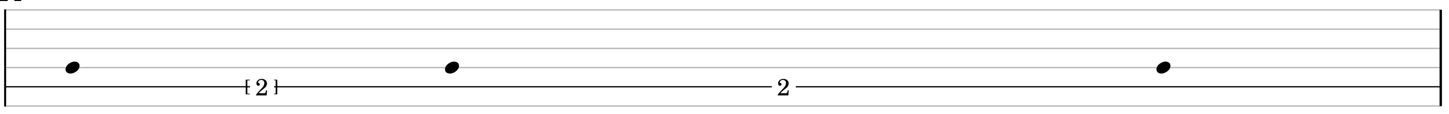
8'00"



8'12"

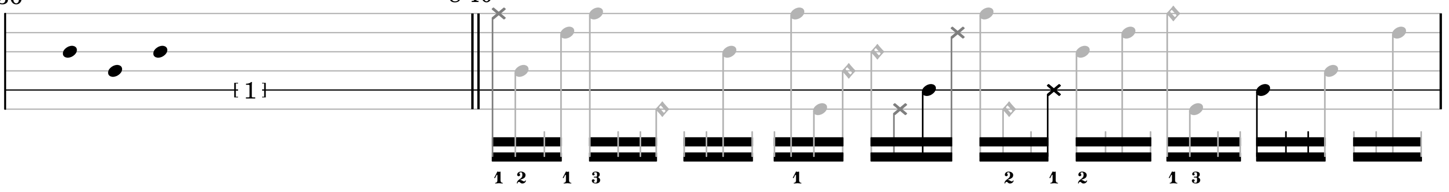


8'24"

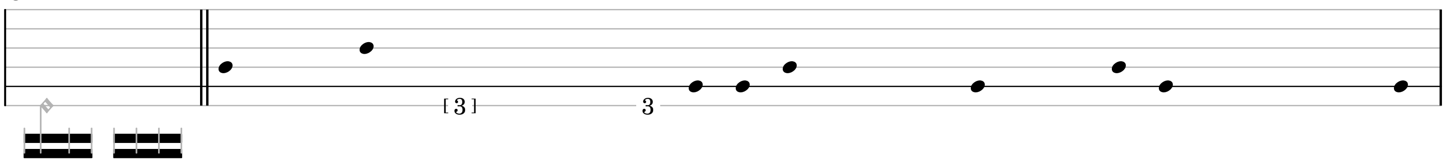


8'36"

8'40"

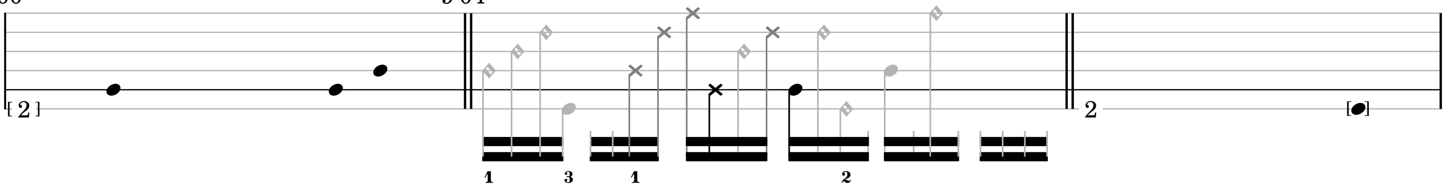


8'48"

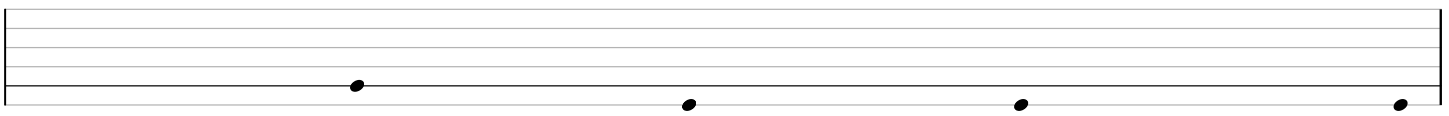


9'00"

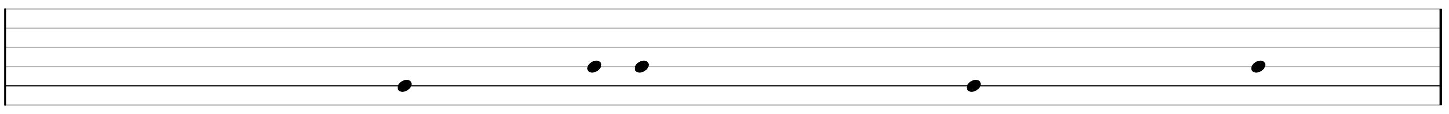
9'04"



9'12"

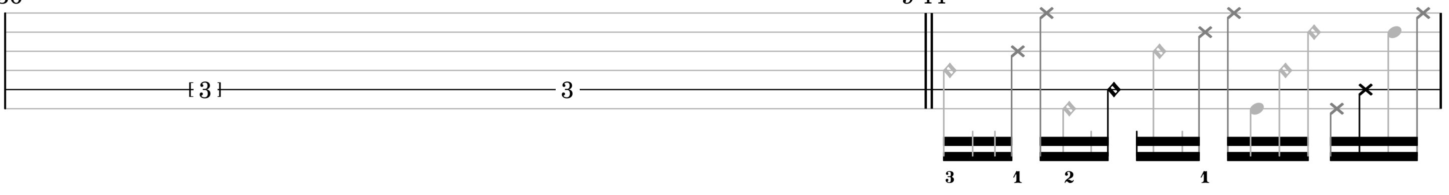


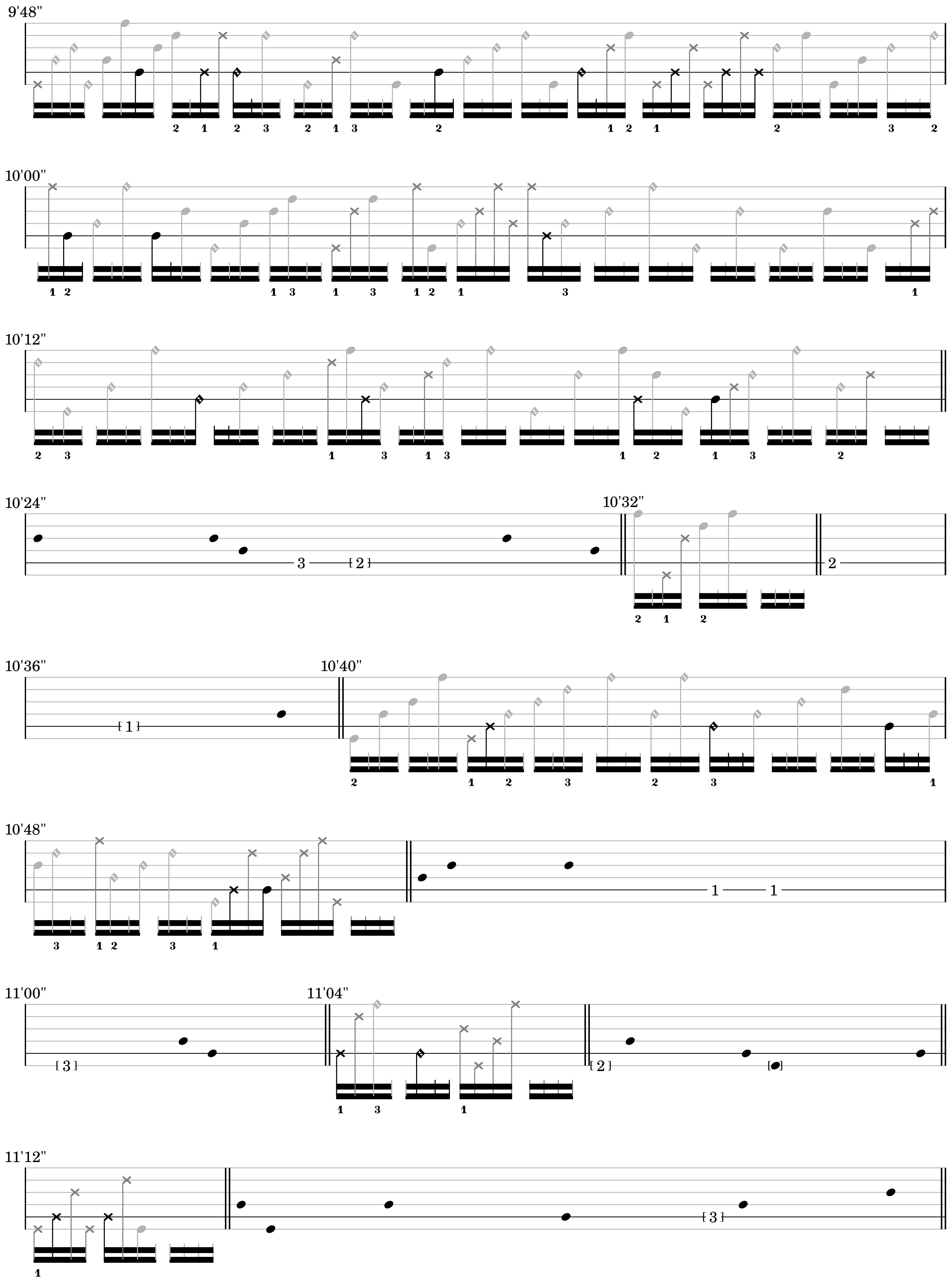
9'24"



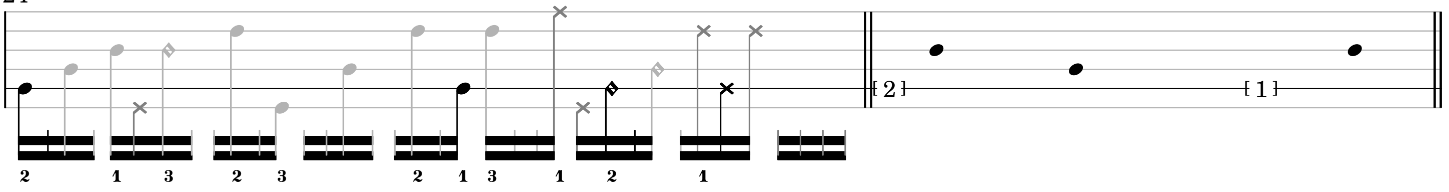
9'36"

9'44"



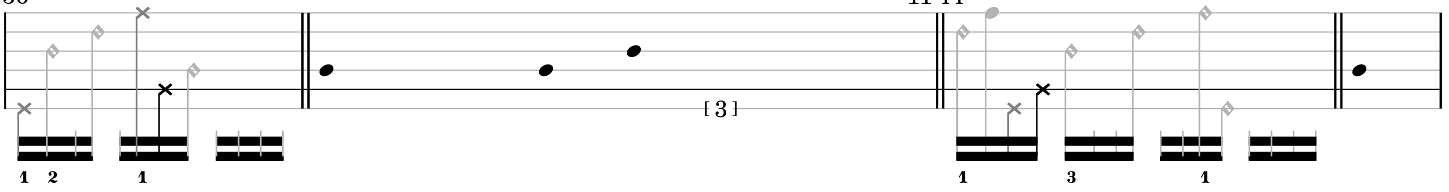

9'48"
10'00"
10'12"
10'24"
10'32"
10'36"
10'40"
10'48"
11'00"
11'04"
11'12"

11'24"

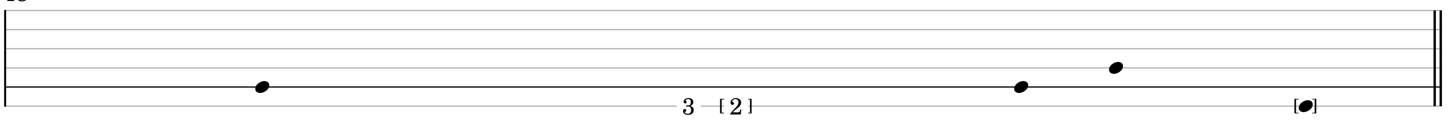


11'36"

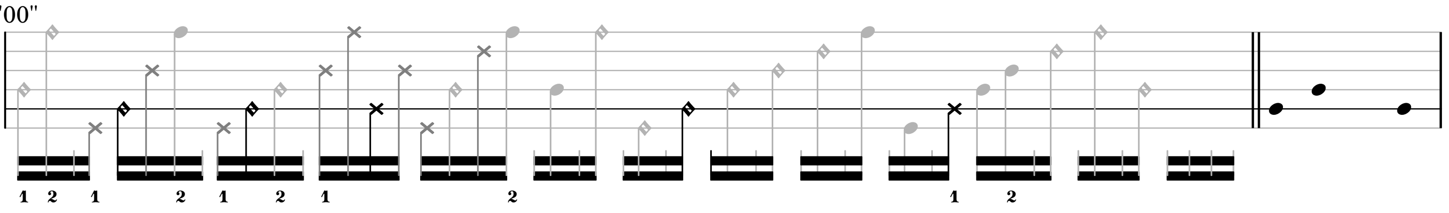
11'44"



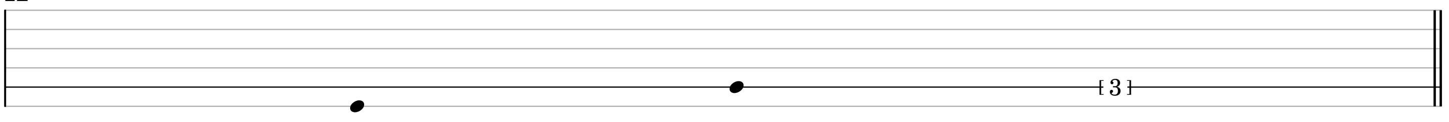
11'48"



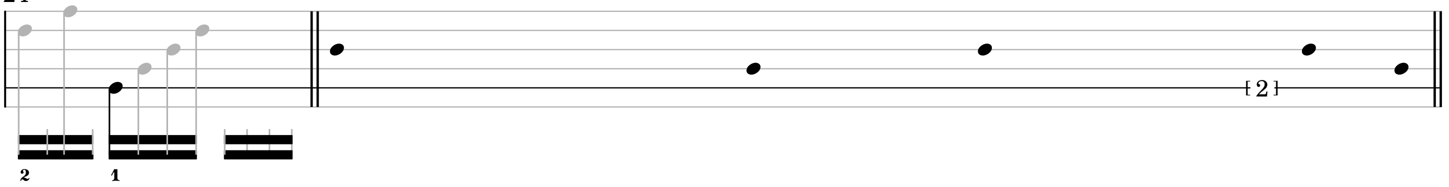
12'00"



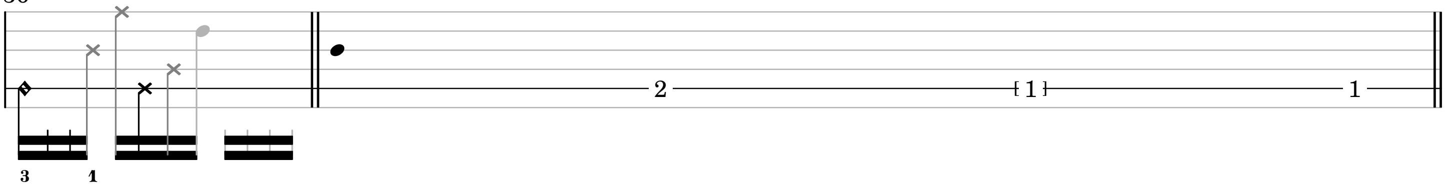
12'12"



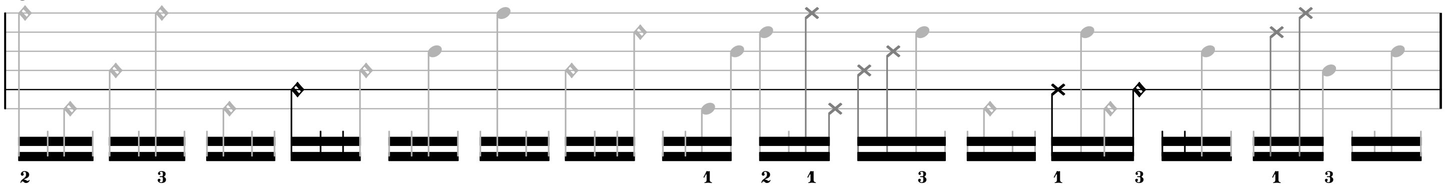
12'24"



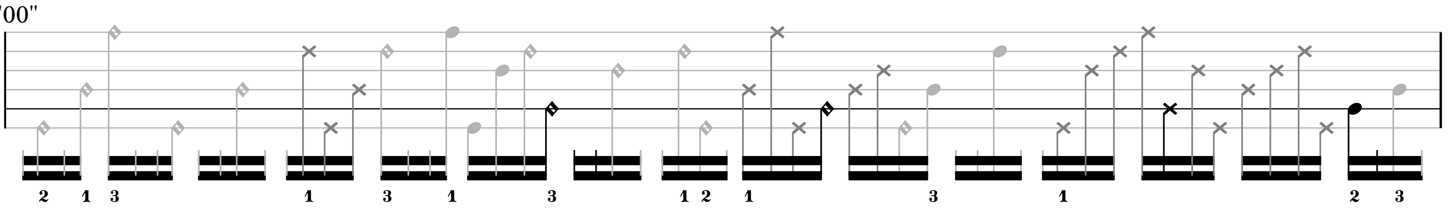
12'36"



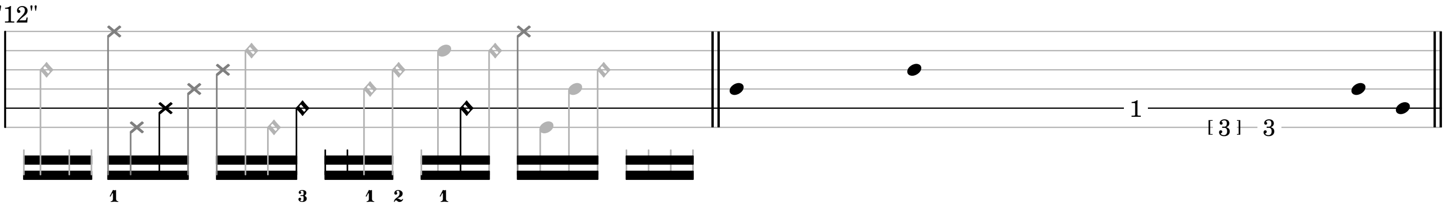
12'48"



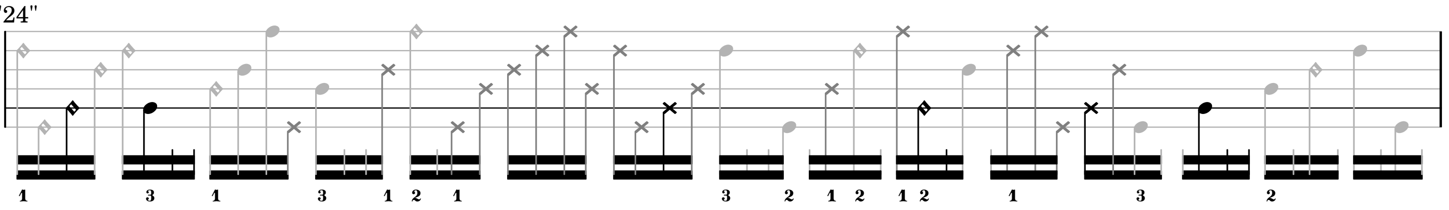
13'00"



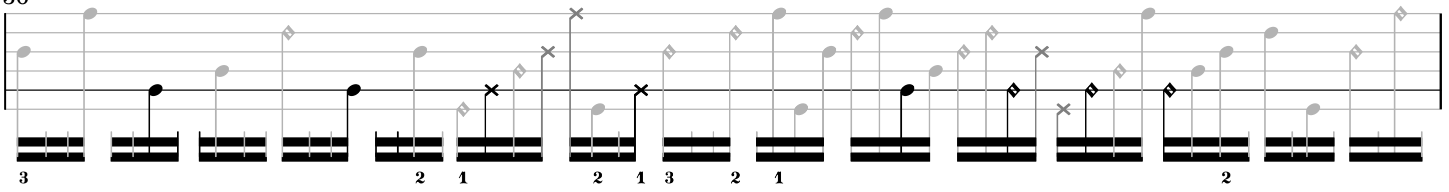
13'12"



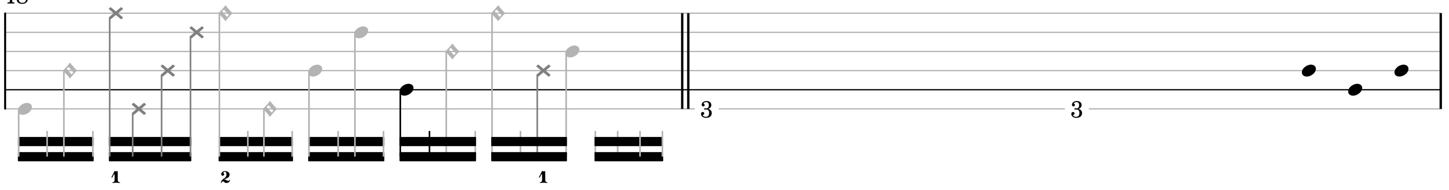
13'24"



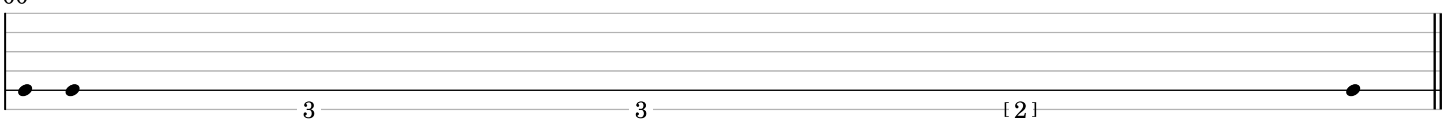
13'36"



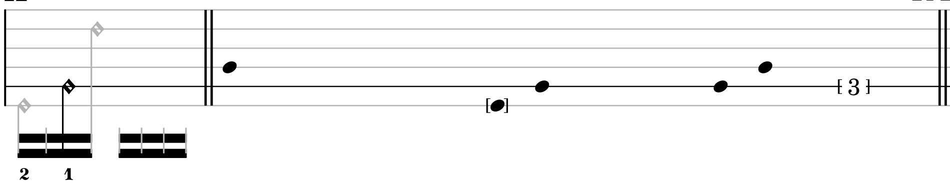
13'48"



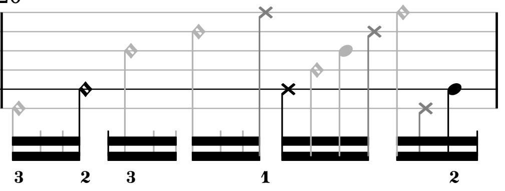
14'00"



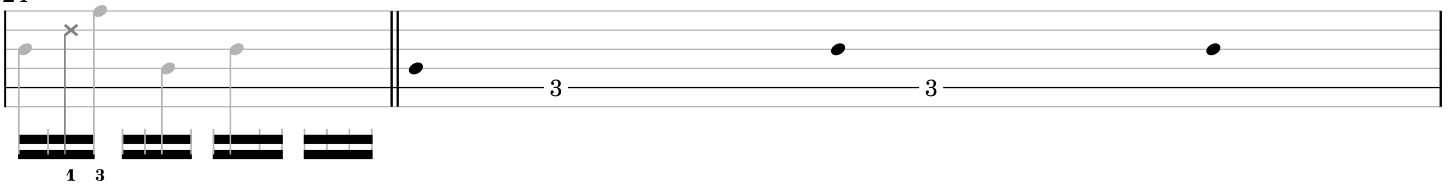
14'12"

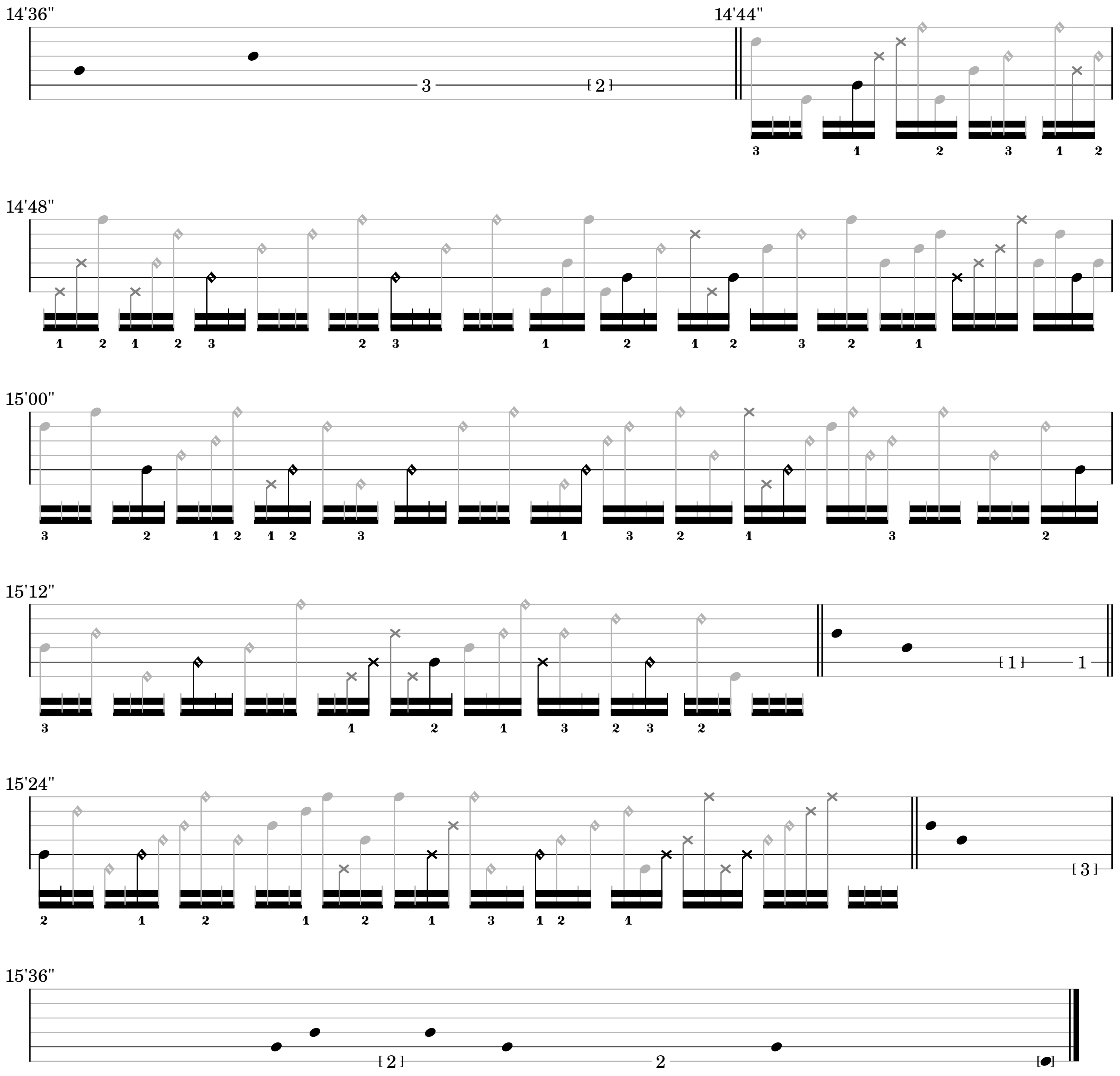


14'20"



14'24"

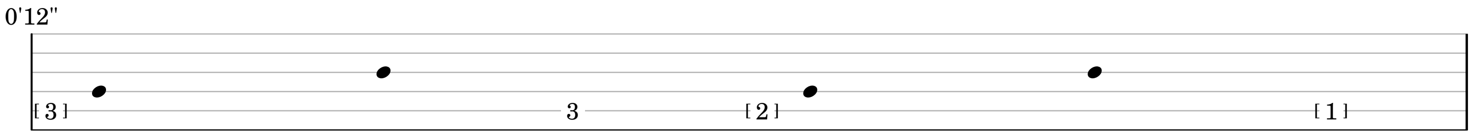


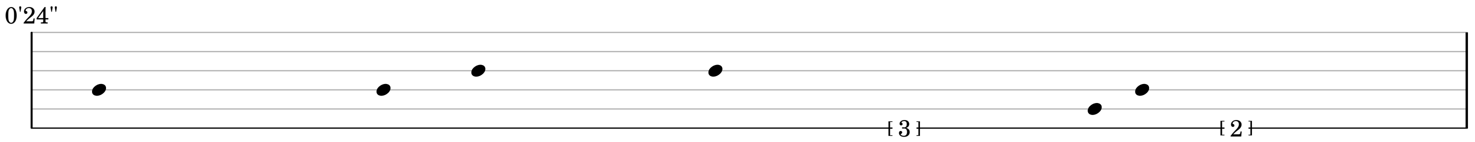

14'36"
14'44"
14'48"
15'00"
15'12"
15'24"
15'36"

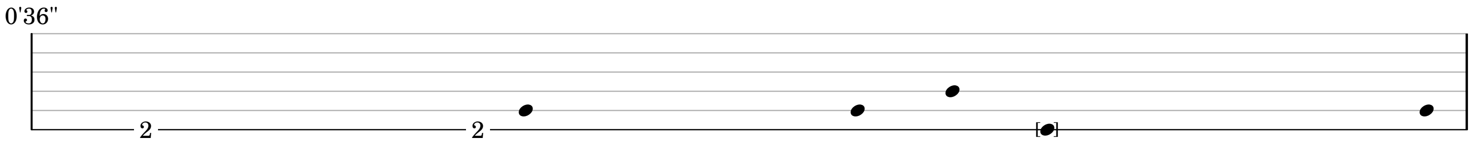
# *ostinato and interrupt*

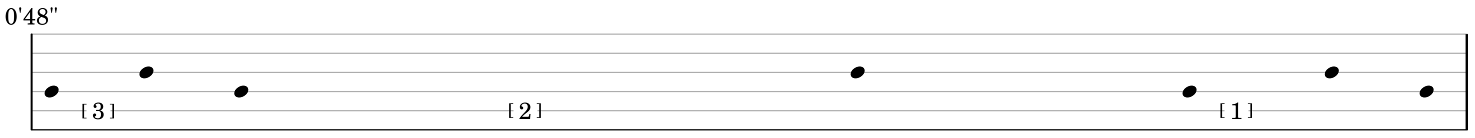
michael winter (mexico city, mx; 2017)
version generated: 2017.12.05

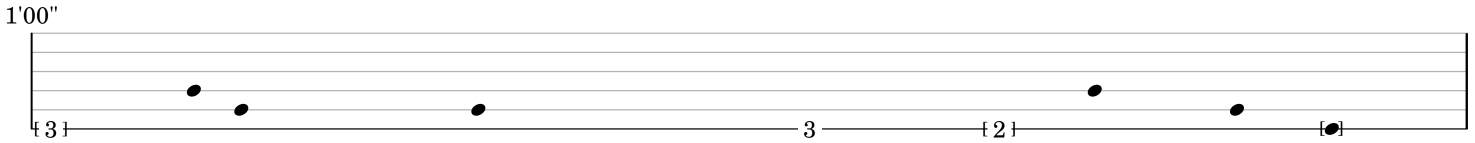
ensemble part 6

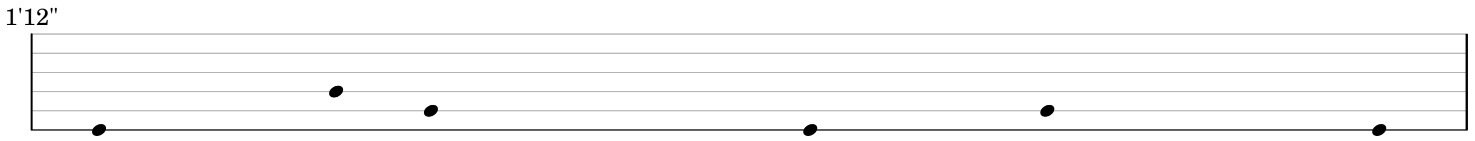


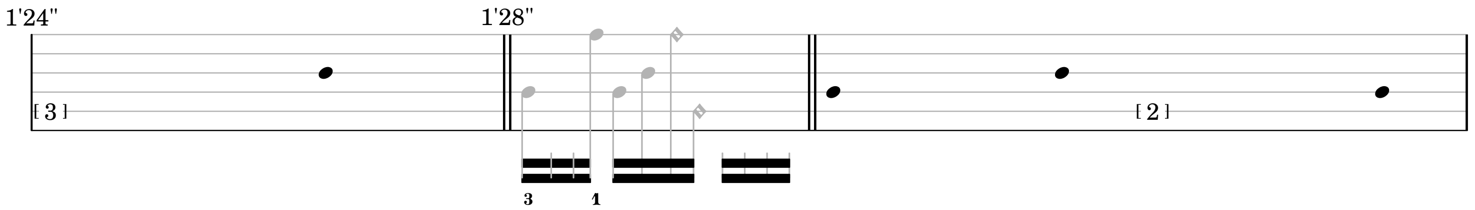


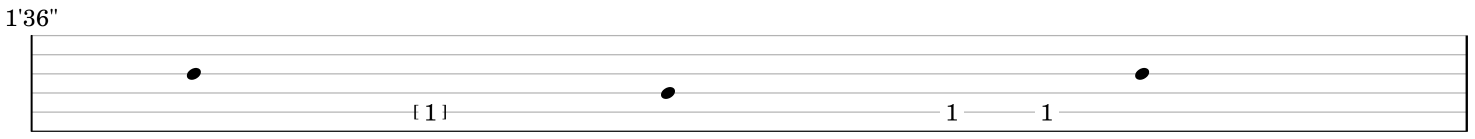




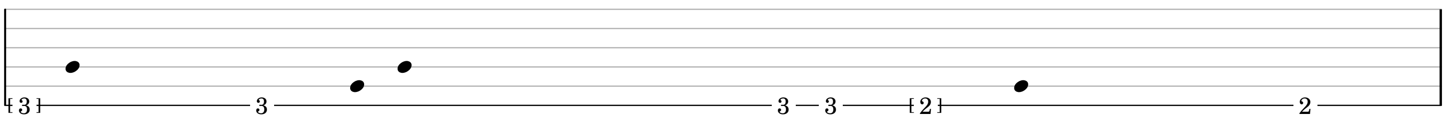






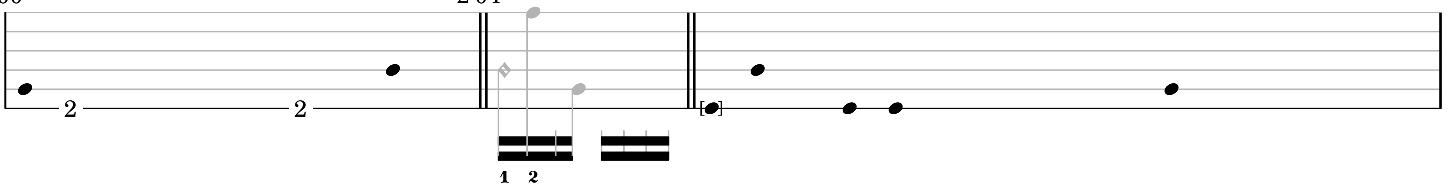


1'48"

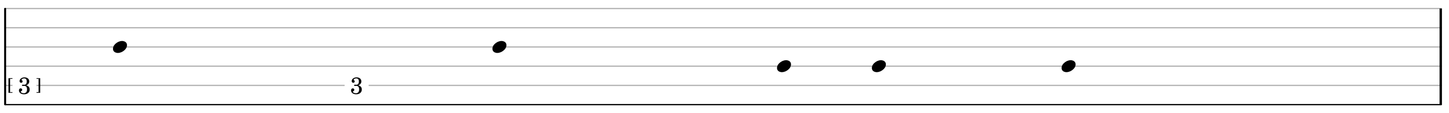


2'00"

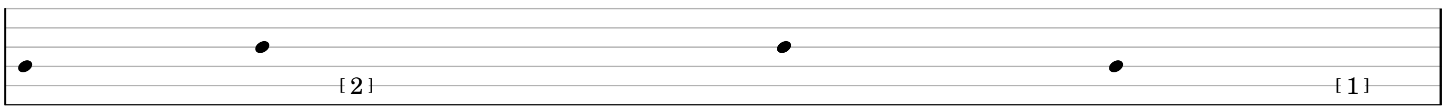
2'04"



2'12"

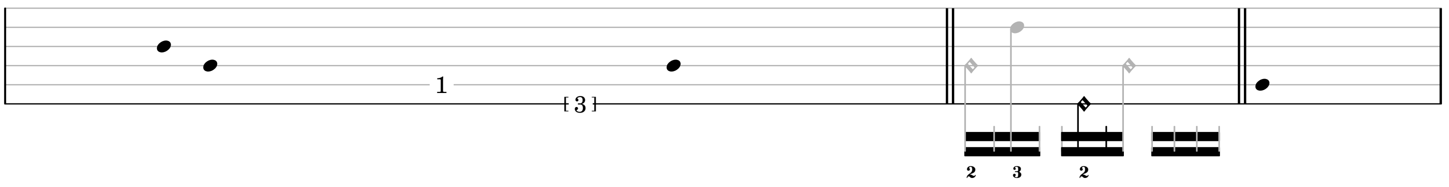


2'24"

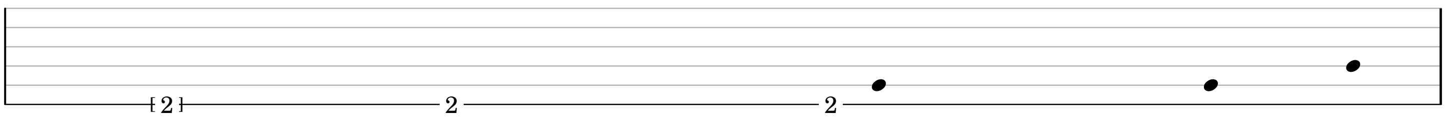


2'36"

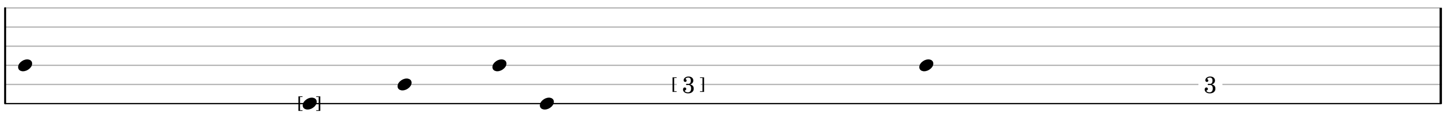
2'44"



2'48"

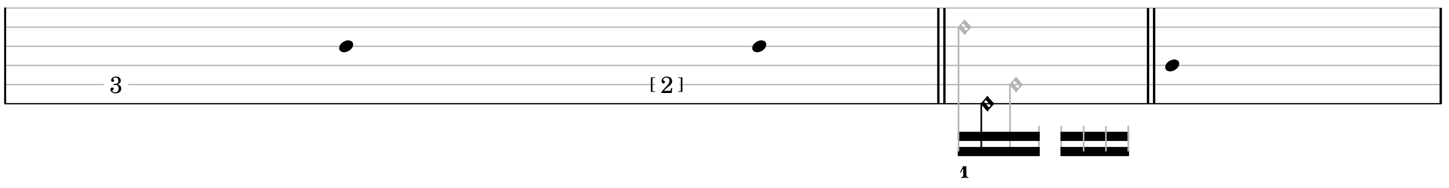


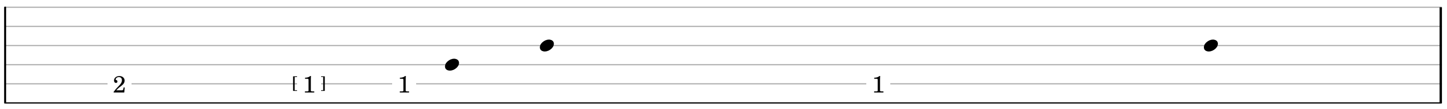
3'00"

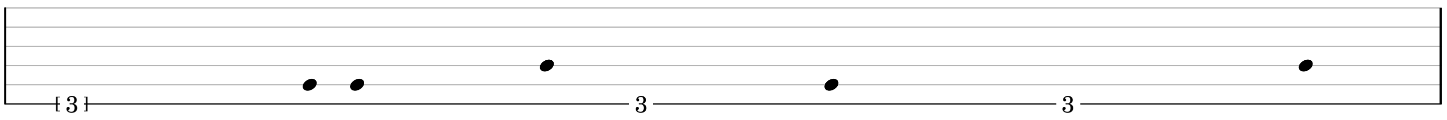


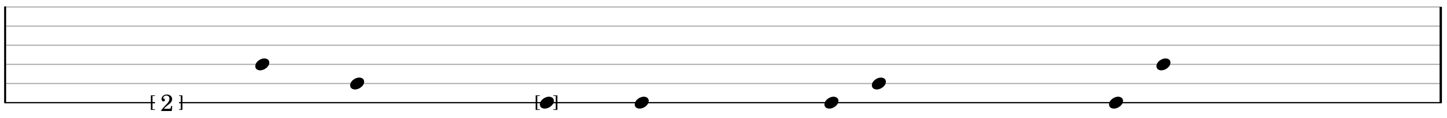
3'12"

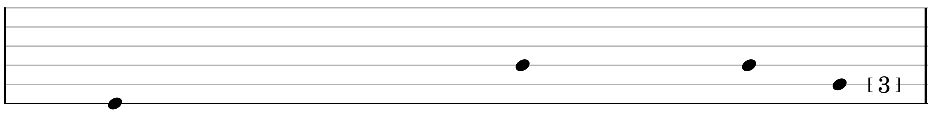
3'20"

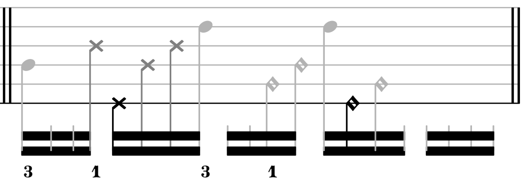


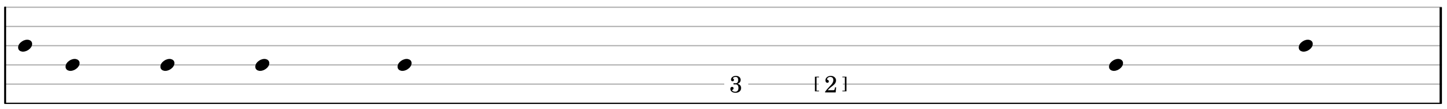
3'24"


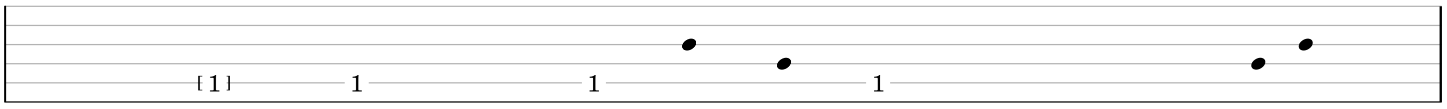
3'36"


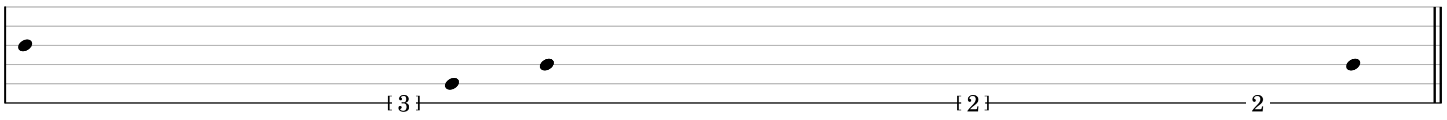
3'48"


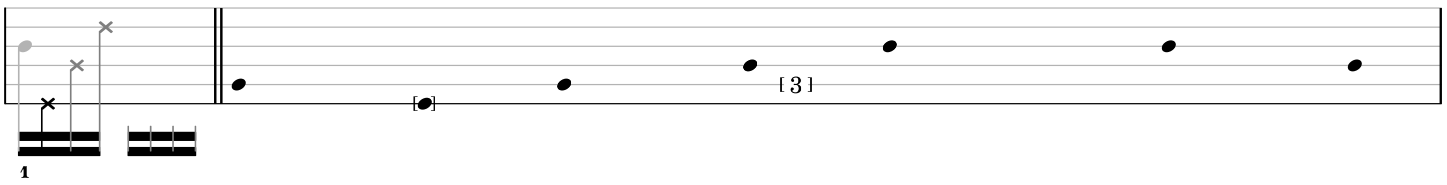
4'00"


4'08"


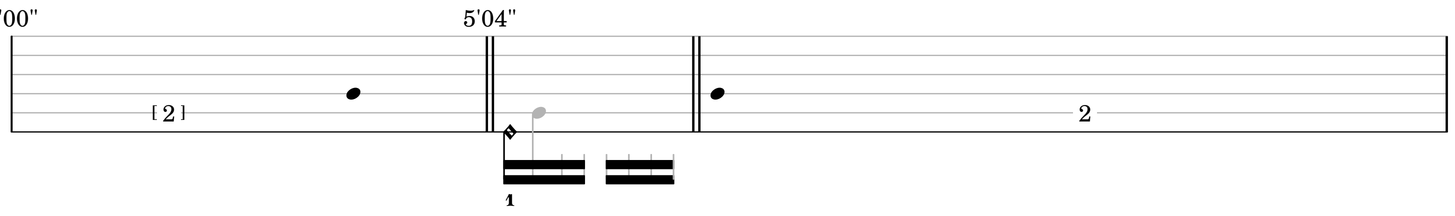
4'12"


4'24"


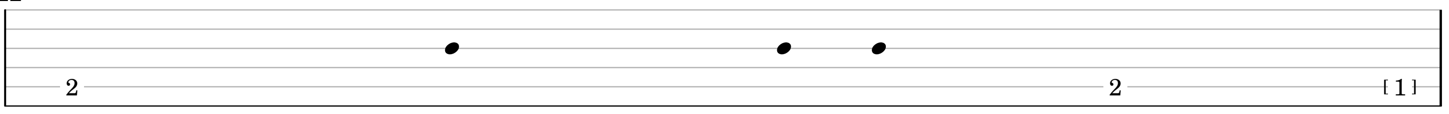
4'36"


4'48"


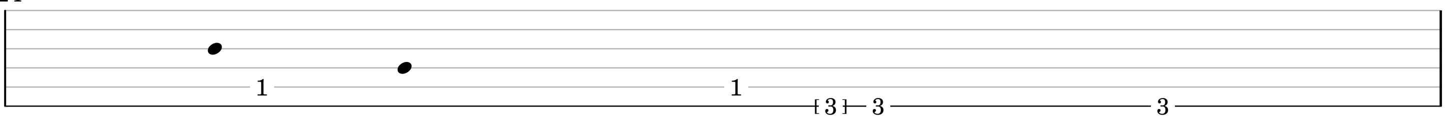
5'00"



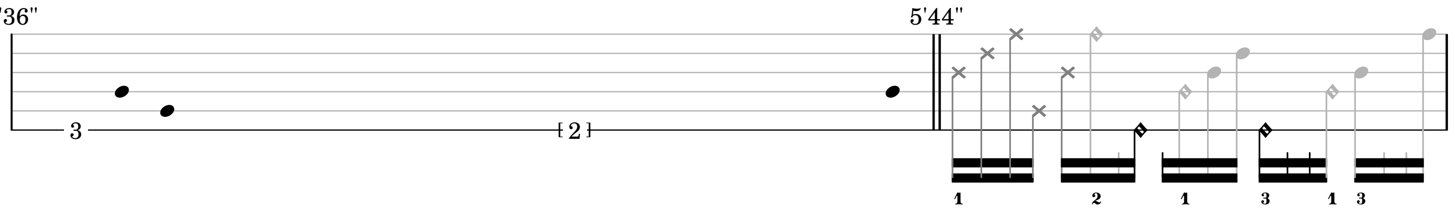
5'12"



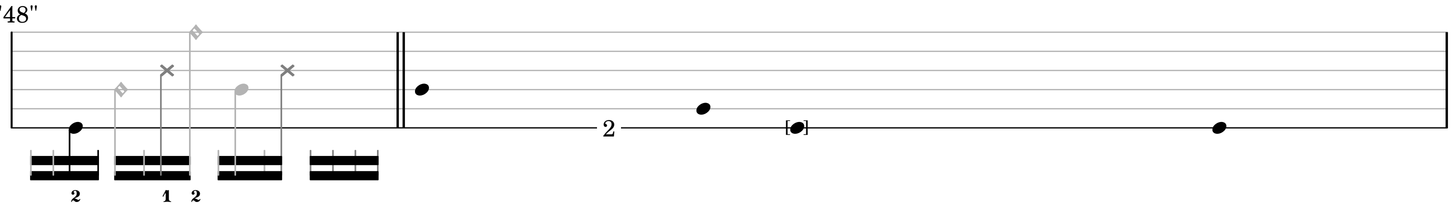
5'24"



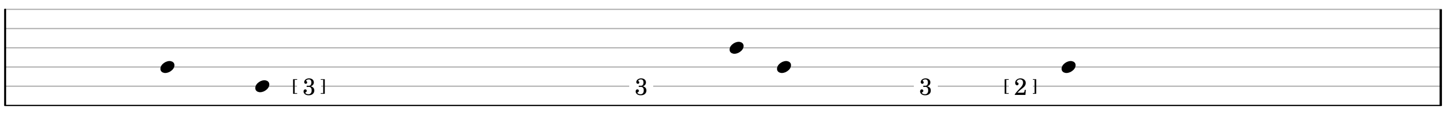
5'36"



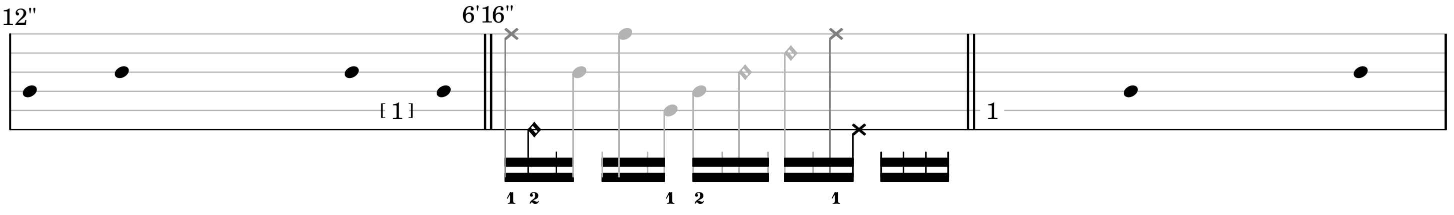
5'48"



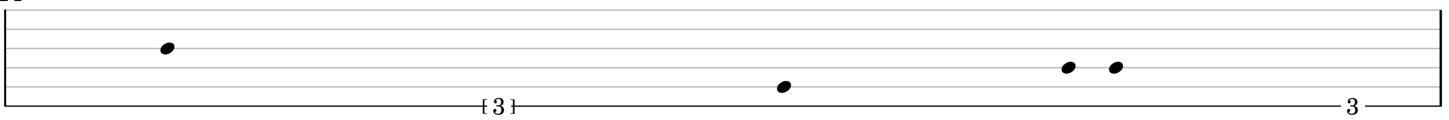
6'00"



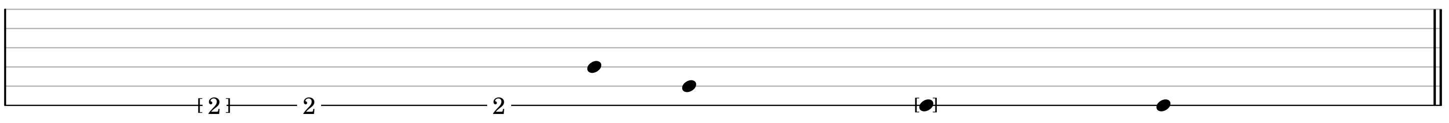
6'12"



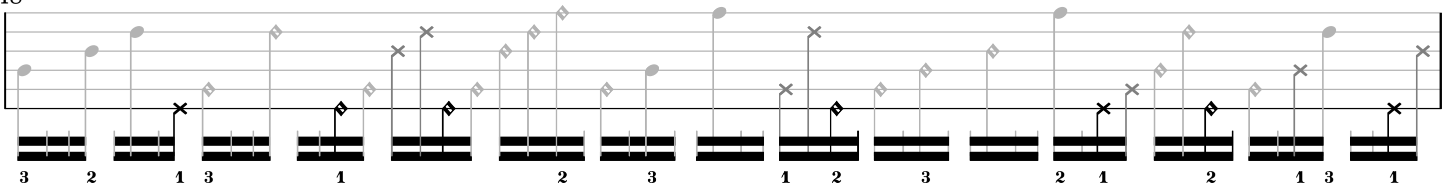
6'24"



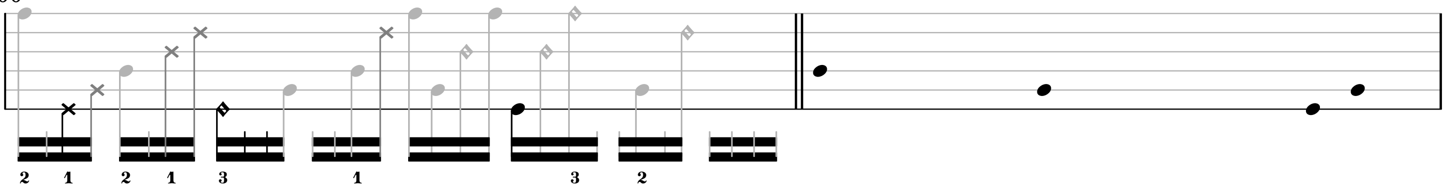
6'36"



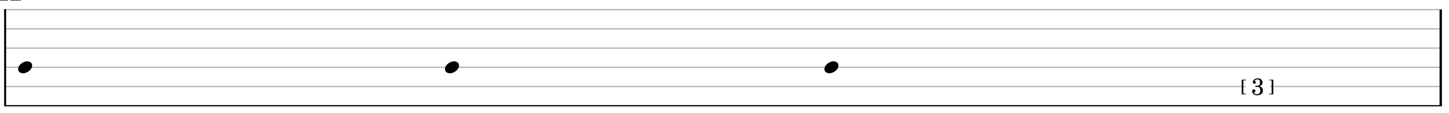
6'48"



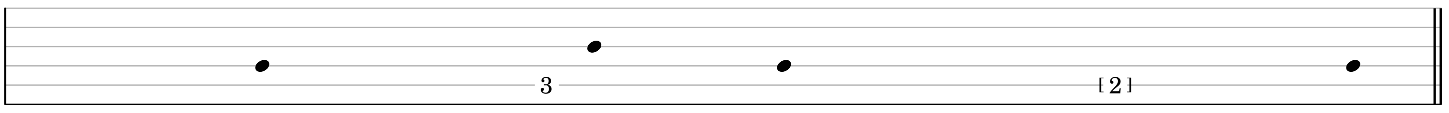
7'00"



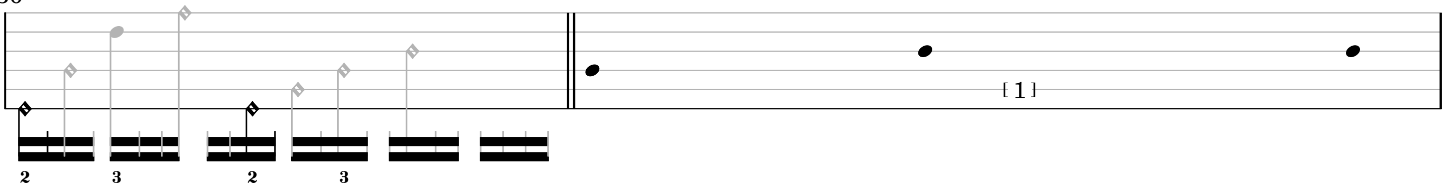
7'12"



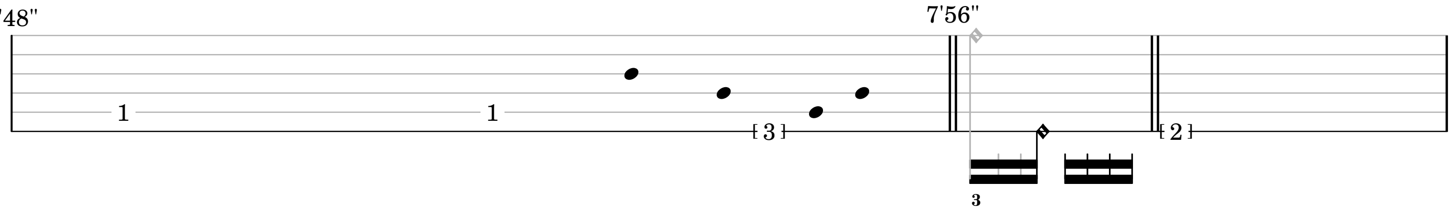
7'24"



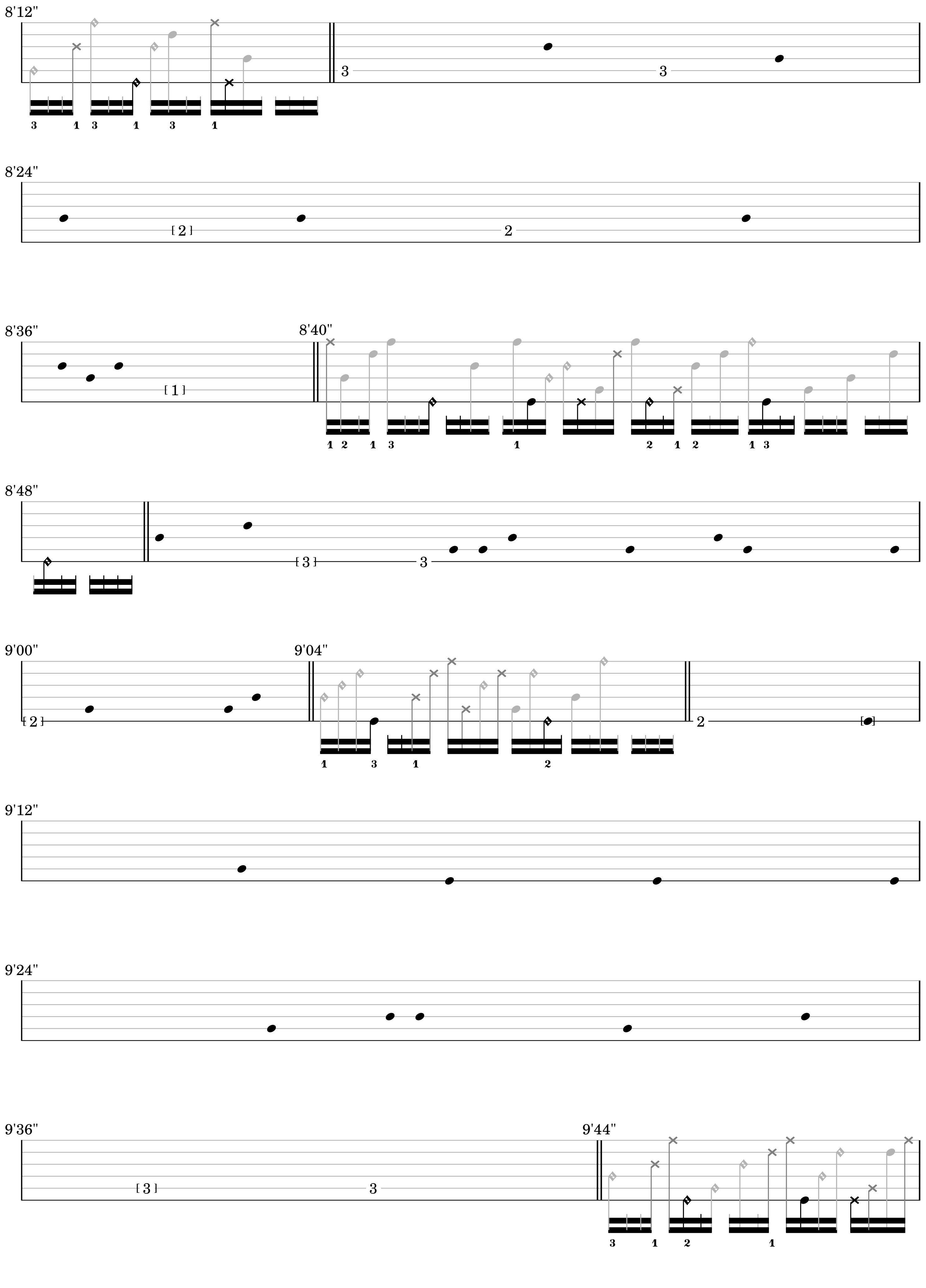
7'36"



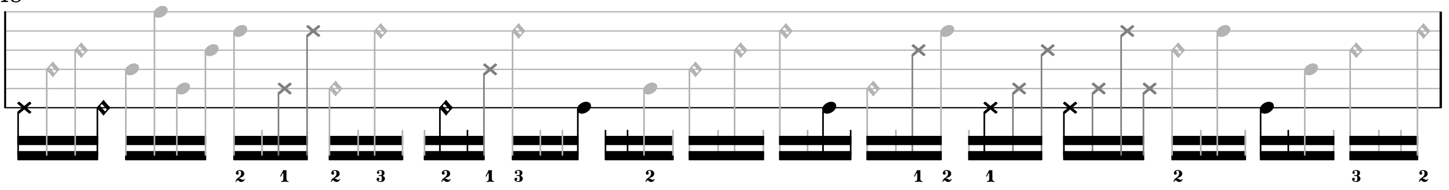
7'48"



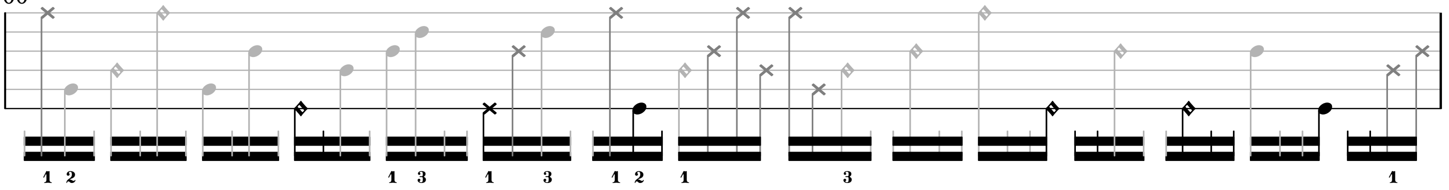
8'00"


8'12"
8'24"
8'36"
8'40"
8'48"
9'00"
9'04"
9'12"
9'24"
9'36"
9'44"

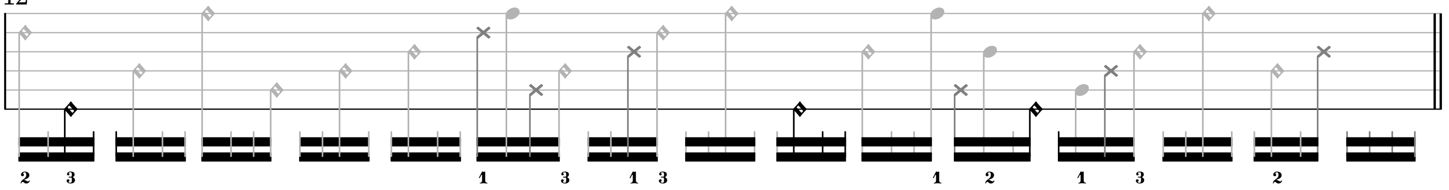
9'48"



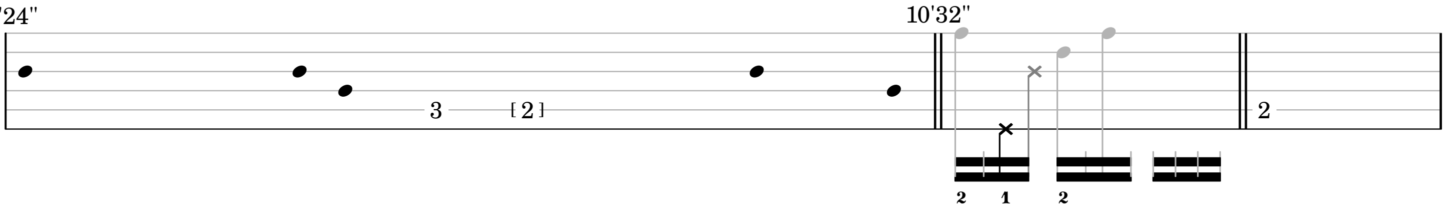
10'00"



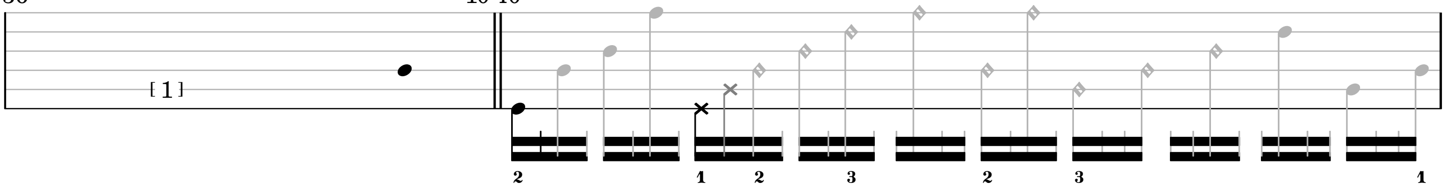
10'12"



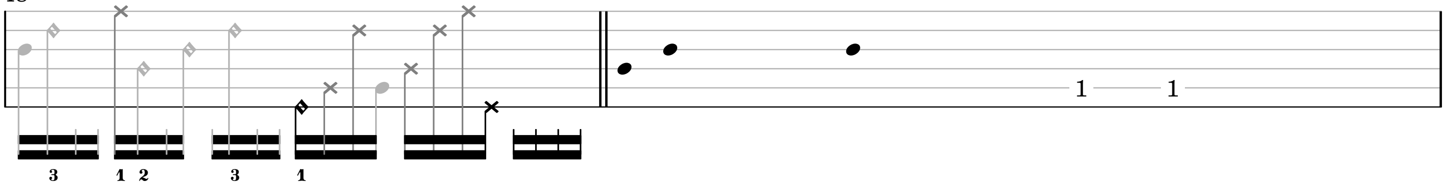
10'24"



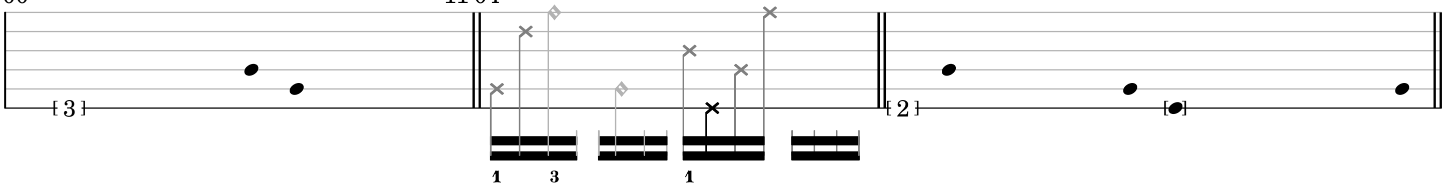
10'36"



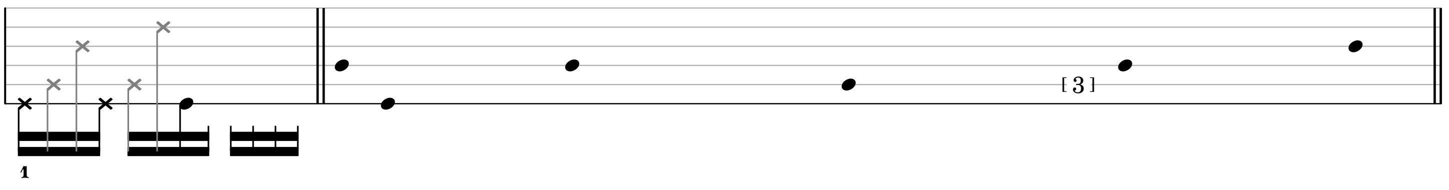
10'48"



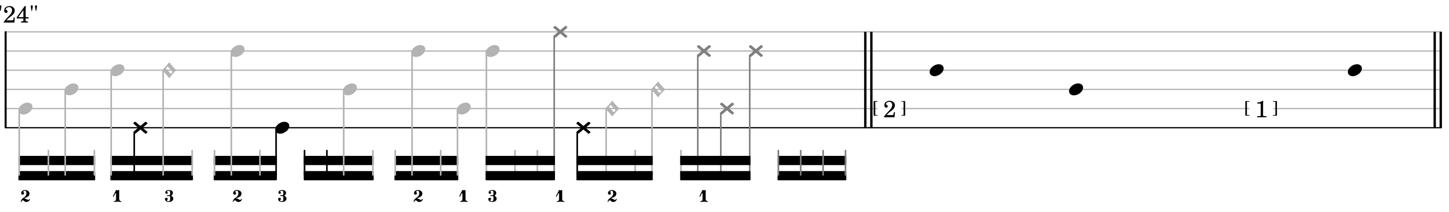
11'00"



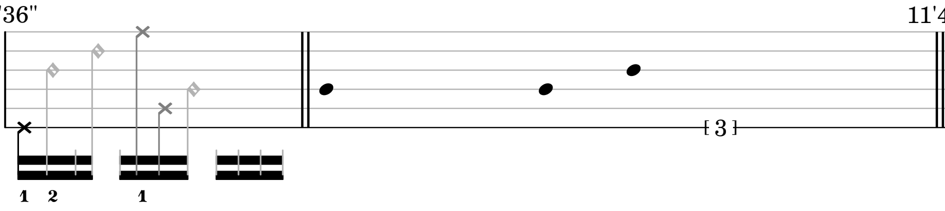
11'12"



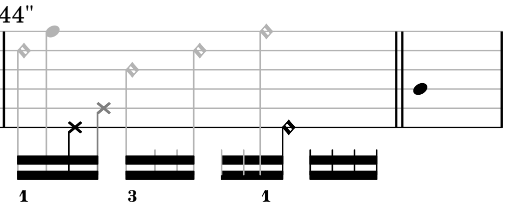
11'24"



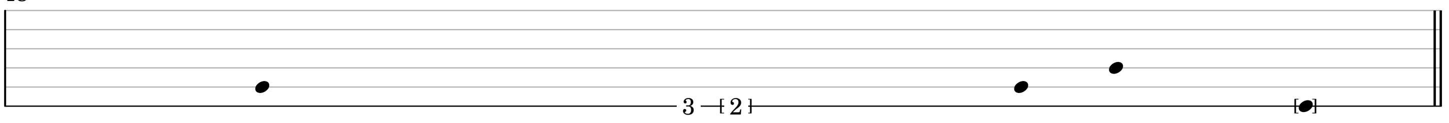
11'36"



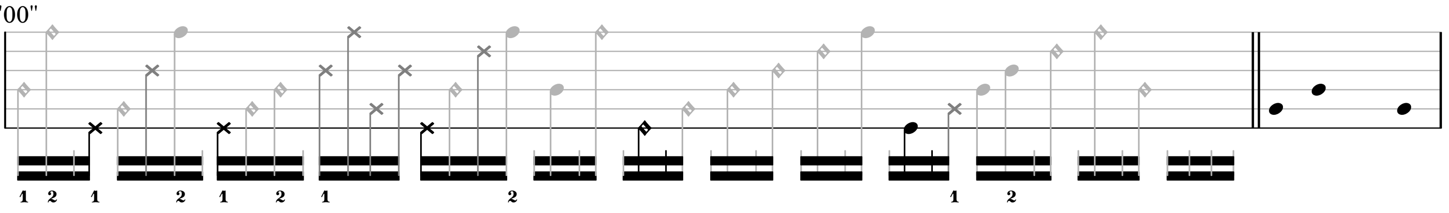
11'44"



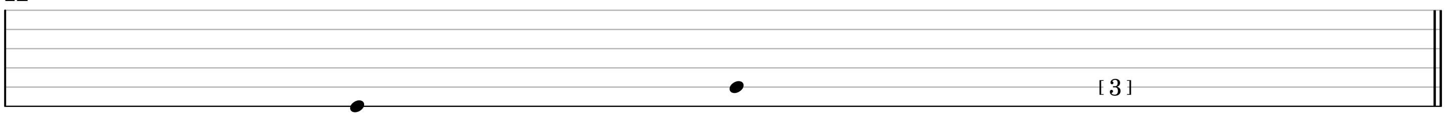
11'48"



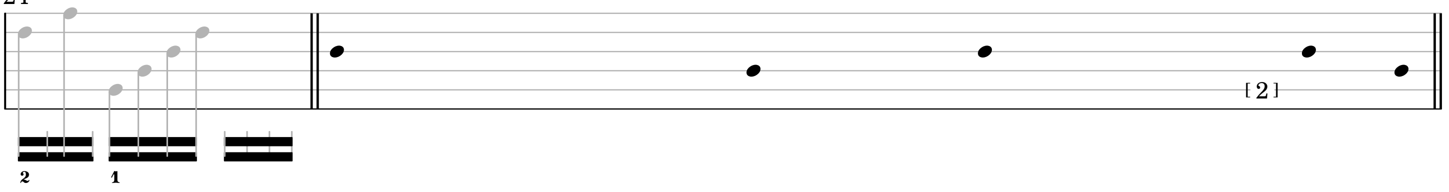
12'00"



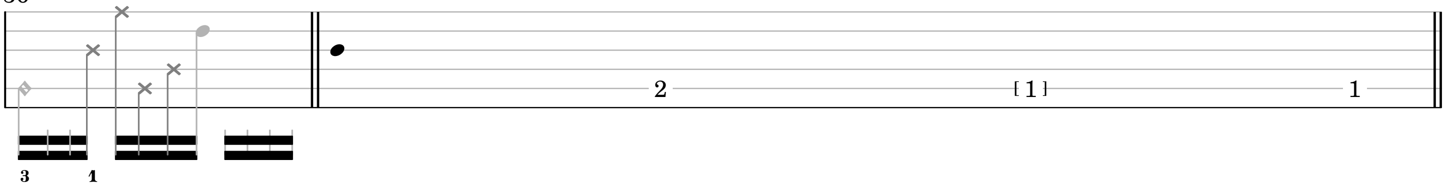
12'12"



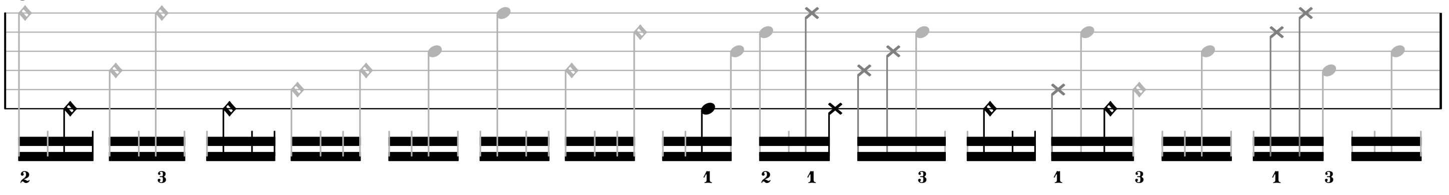
12'24"



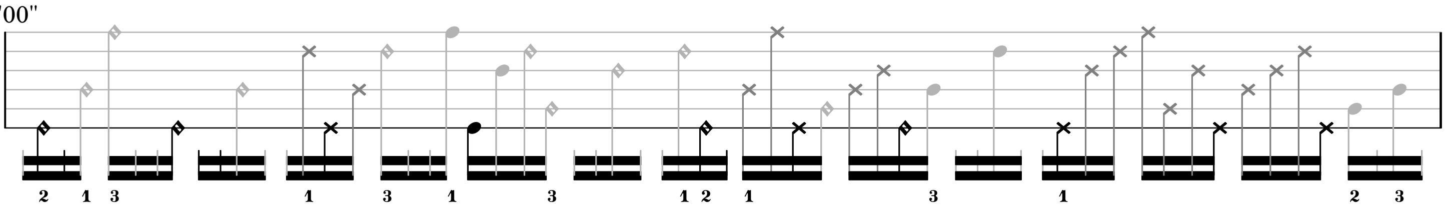
12'36"



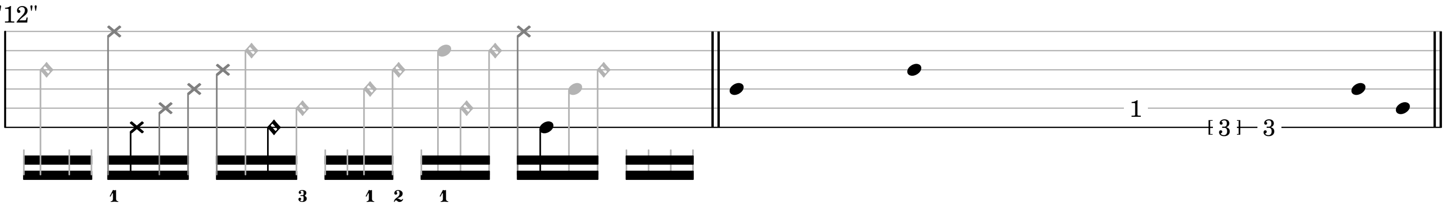
12'48"



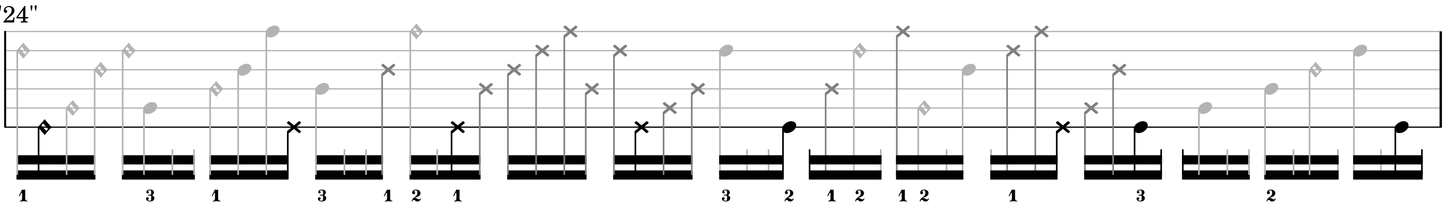
13'00"



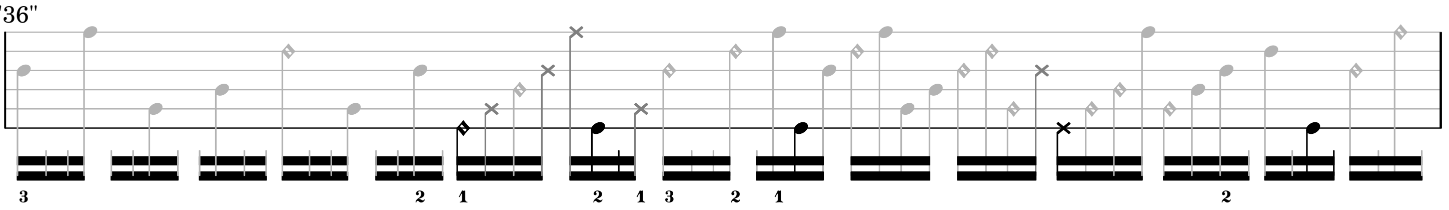
13'12"



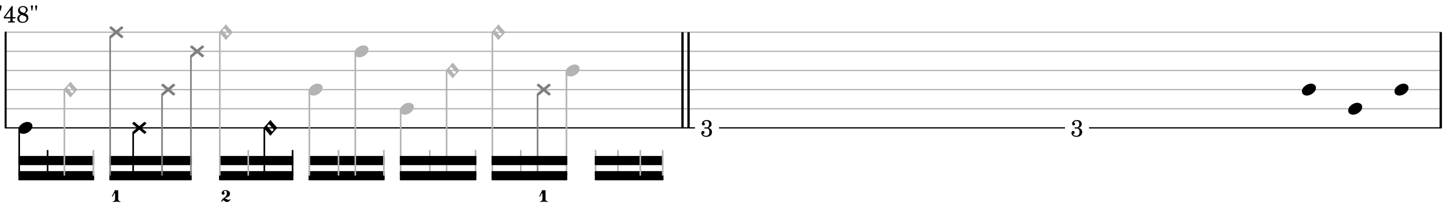
13'24"



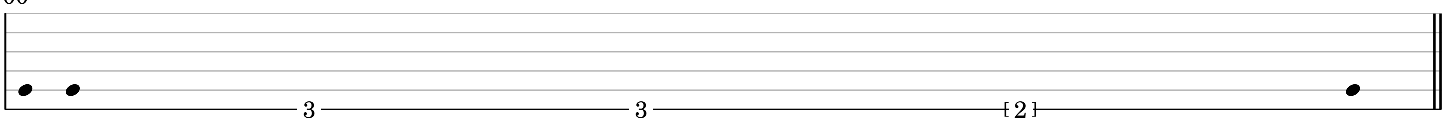
13'36"



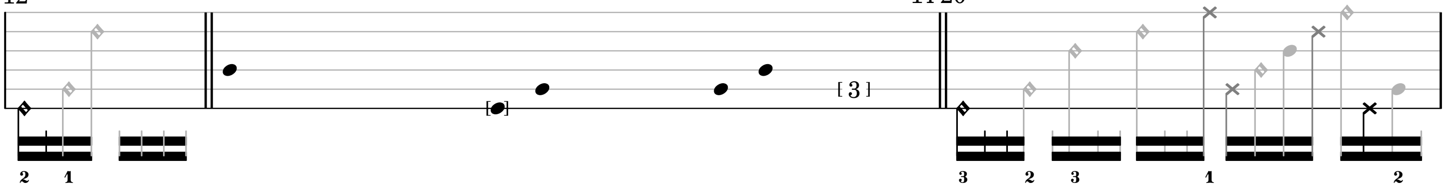
13'48"



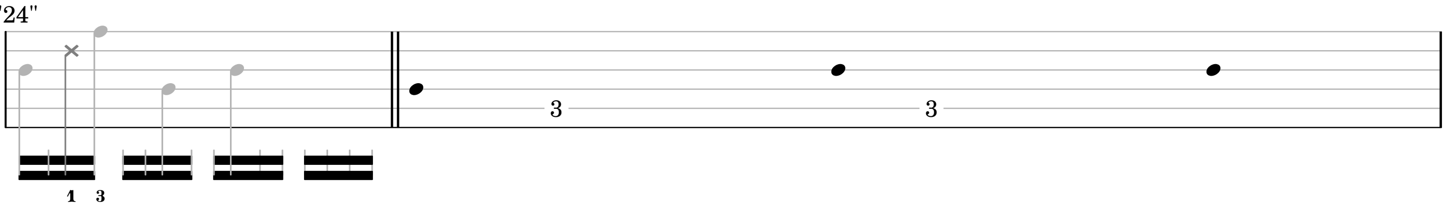
14'00"



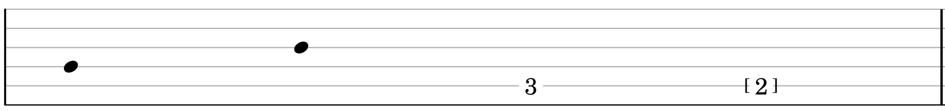
14'12" 14'20"



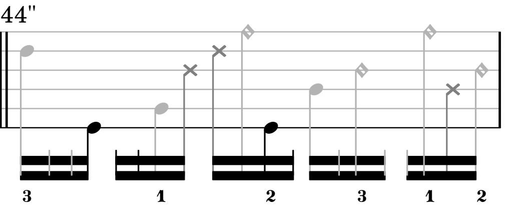
14'24"



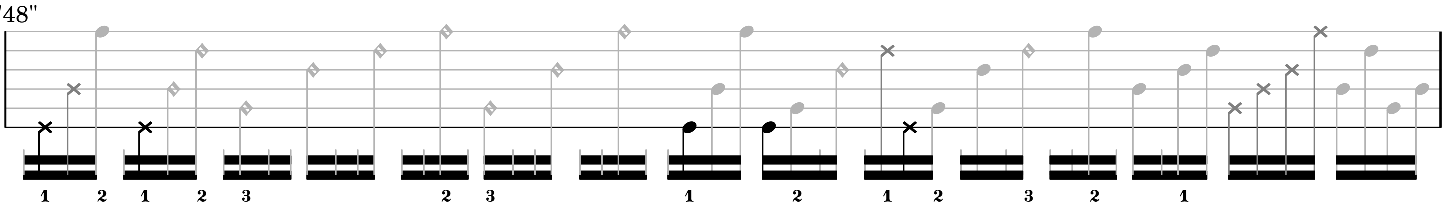
14'36"



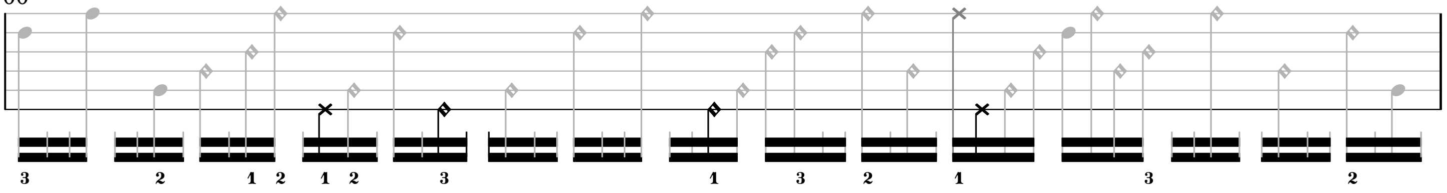
14'44"



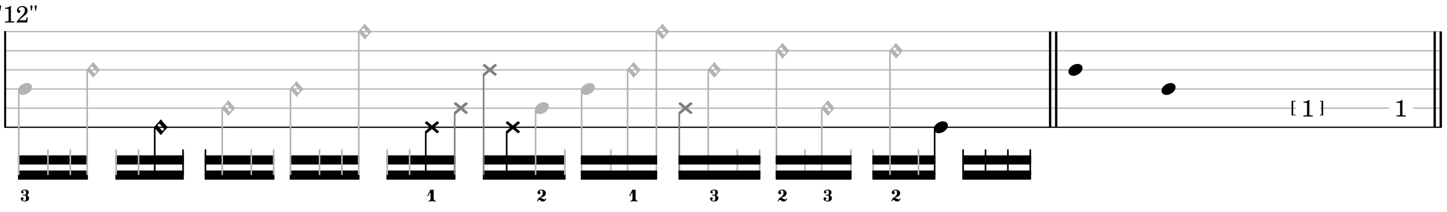
14'48"



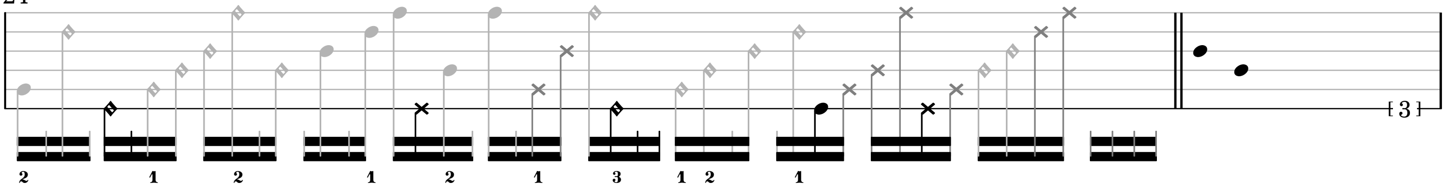
15'00"



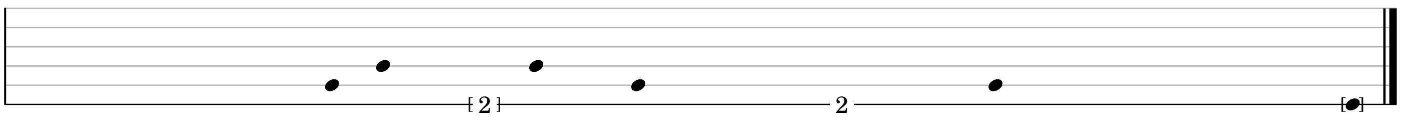
15'12"



15'24"



15'36"



ostinato_and_interrupt_main.scd

```
(
// MAIN LAUNCH - Best to reboot interpreter and server first to make sure all buffers are cleared
~dir = thisProcess.nowExecutingPath.dirname;
"ostinato_and_interrupt_generator_synthdef.scd".loadRelative(true, {
    "ostinato_and_interrupt_player_synthdef.scd".loadRelative(true, {
        "ostinato_and_interrupt_nrt_generator_function.scd".loadRelative(true, {
            "ostinato_and_interrupt_lilypond_generator_function.scd".loadRelative(true, {
                "ostinato_and_interrupt_gui_generator_function.scd".loadRelative(true, {
                    if(File.exists(~dir +/+ "../audio/ostinato_and_interrupt.wav"), {
                        ~appStatusString = "loading buffer";
                        ~generateGUI.value;
                        ~appStatusFunc.play;
                        Buffer.read(s, ~dir +/+ "../audio/ostinato_and_interrupt.wav", action: {
                            |buf| ~totalDur = buf.duration;
                            ~play = Synth.new(\play, [\buf, buf]);
                            ~appStatusFunc.stop;
                            {~appStatus.string = "ready"}.defer
                        });
                    }, {
                        ~appStatusString = "generating data";
                        ~generateGUI.value;
                        ~appStatusFunc.play;
                        ~generateData.value(true)});
})})})})});
)
```

ostinato_and_interrupt_generator_synthdef.scd

```
(
//~~SYNTHDEF THAT GENERATES THE PIECE
~ostinato_and_interrupt = SynthDef(\ostinato_and_interrupt, {

    // TODO: Replace PlayBuf with PlayBufCF

    //~~~~~~~~~~~~~~~~ARG DECLARATION~~~~~~~~~~~~~~~~
    arg scoreBuf, seed = 20170121, rt = 0, // set score buffer, random seed, and if run in realtime (sets outputs differently)
    percBufDelims = #[0, 1, 2, 3, 4, 5, 6], // set buffers for percussion samples
    guitOpenBufDelims = #[0, 1, 2, 3, 4, 5, 6], // set buffers for open strings
    guitHarmBufDelims = #[0, 1, 2, 3, 4, 5, 6], // set buffers for harmonics
    guitMuteBufDelims = #[0, 1, 2, 3, 4, 5, 6], // set buffers for muted strings
    guitBassBufDelims = #[0, 1, 2, 3, 4, 5, 6]; // set buffers for bass notes

    //~~~~~~~~~~~~~~~~VAR DECLARATION~~~~~~~~~~~~~~~~
    var fund, durUnit, tuning, oBassNotes, oOpenStringNotes; // static vars
    var li, trig, lastoString, lastoDurTemp, switchTrig, lastoStepper; // feedback in
    var env, iEnv, oEnv, currentDur, switch; // switch
    var oTrig, oStepper, oBassNote, oString, oNote, oFreq, oDurTemp, oDur, oStringCounts,
        oPluckedStrings, oSwitchTrig, oSwitchArm, oSwitchTrigDel; //ostinato
    var startState, endState, endTrig; // start and end states
    var iTrig, iSound, iDur, iString, iEnsemble, iPerc, iPluckedStrings,
        iSwitchTrig, iSwitchArm, iSwitchTrigDel; // interrupt
    var aDust, aLBuf, aFlicker, aBNoise, aWNoise, aSinBeat, aSinTrig, aFadeIn, aFadeOut; // electronic accompaniment
    var count; // score
    RandSeed.ir(1, seed); // random seed - change for different results

    //~~~~~~~~~~~~~~~~STATIC VARS~~~~~~~~~~~~~~~~
    fund = 38; // fundamental in midi (guitar low d)
    durUnit = 0.2; // minimum durational unit
    tuning = [1/1, 3/2, 2/1, 5/2, 7/2, 4/1]; // tuning of the open strings
    oBassNotes = [10, 9, 8, 3, 2, 0]; // bass notes relative to fundamental in semitones
    oOpenStringNotes = [0, 7, 12, 16]; // open string notes relative to fundamental in semitones (used only for score)

    //~~~~~~~~~~~~~~~~FEEDBACKIN~~~~~~~~~~~~~~~~
    li = LocalIn.kr(6, [0, -1, -1, 0, 0, 1]); // init vals
    trig = li[0] + TDelay.kr(Impulse.kr(0), 12); // start the guitar 12 seconds after electronic accompaniment
    lastoString = li[1]; // get last ostinato string for
    lastoDurTemp = li[2]; // get last duration value (not yet multiplied by durUnit)
    lastoStepper = li[3]; // last position of the ostinato
    switchTrig = li[4]; // trigger to change from ostinato to interrupt
    startState = li[5]; // delay before interrupts can start to occur

    //~~~~~~~~~~~~~~~~SWITCH~~~~~~~~~~~~~~~~
    env = EnvGen.kr(Env.new([0, 1], 60 * 13, \sine), 1 - startState); // env that controls the switching between ostinato and interrupt
    oEnv = 0.25 + (env * 3.75); // chance that ostinato will switch to interrupt over time
    iEnv = 1 + (env * 29); // chance that interrupt will switch to ostinato over time
    currentDur = (Line.kr(0, 60 * 20, 60 * 20) / 0.2).trunc; // time tracker
    switch = Stepper.kr(switchTrig, 0, 0, 1); // state - ostinato or interrupt
    Poll.kr(trig, currentDur, \currentDur); // poll current duration
    Poll.kr(trig, env, \env); // monitor current position of envelope

    //~~~~~~~~~~~~~~~~OSTINATO~~~~~~~~~~~~~~~~
    // mute triggers when off
    oTrig = trig * (switch <= 0);
    // update string counts
    oStringCounts = { arg i;
        var isString = (i <= lastoString) * (lastoString <= i);
        PulseCount.kr(oTrig * isString, Changed.kr(lastoStepper)) } ! 4;
    // step through the bass note of the ostinato once all notes in a cell have been played
    oStepper = Stepper.kr((oTrig * TWChoose.kr(oTrig, [0, 1], [1, 1], 1) * (Mix.new(oStringCounts > 0) >= 3)), 0, 0, 5)
```

```
        ;
    oBassNote = Select.kr(oStepper, oBassNotes);
    // select duration favoring a change for longer notes to promote flurries of shorter notes
    oDurTemp = TWChoose.kr(oTrig * TWChoose.kr(oTrig, [0, 1], [(((lastoDurTemp <= 0) * 0.75) + 1) * (lastoDurTemp > 5), 1], 1),
        [1, 2, 3, 4, 5, 6, 7], Select.kr(oStepper < 3, [[4, 3, 2, 2, 2, 1, 1], [2, 3, 3, 3, 2, 1, 1]]), 1);
    // add jitter to duration (if this is odd unfortunately a rounding error is produced)
    oDur = 2 * (oDurTemp + TIRand.kr(0, Select.kr(oDurTemp < 3, [2, 0] ), oTrig));
    // select string (always oBassNote if oBassNote has been stepped to next) and promote change on shorter notes
    oString = TWChoose.kr(oTrig,
        Select.kr(oStepper < 3, [[0, 1, 2], [1, 2, 3]]),
        Select.kr(Changed.kr(oStepper), [[4, 3, 3] * (1 / pow(Select.kr(oStepper < 3,
            [oStringCounts.drop(-1), oStringCounts.drop(1)]) + 1, Select.kr(lastoDurTemp < 2, [0.75, 2]))), [1, 0, 0]]), 1);
    // select note based on string
    oNote = Select.kr((oString - (oStepper < 3)) > 0, [oBassNote, Select.kr(oString, oOpenStringNotes)]);
    // play guitar
    oPluckedStrings = { |i| var string, isString, isStringDel, freq, snd;
        string = 5 - i; // invert string number (since oString 1 is guitar string IV)
        isString = oTrig * (string <= (5 - oString)) * ((5 - oString) <= string); // check if string is triggered
        isStringDel = TDelay.kr(isString, 0.01); // slight delay for envelope
        // play samples and select open string or bass note
        snd = PlayBuf.ar(1, Select.kr(Latch.kr((oString - (oStepper < 3)) > 0, isStringDel),
            [TIRand.kr(Select.kr(oStepper, guitBassBufDelims), Select.kr(oStepper, guitBassBufDelims) - 1, isStringDel),
                TIRand.kr(guitOpenBufDelims[string], guitOpenBufDelims[string + 1] - 1, isStringDel)
        ]), Latch.kr(1, isStringDel), isStringDel) * (1 - EnvGen.ar(Env.new([0, 1, 0], [0.01, 0.01]), isString));
        Out.ar(Select.kr(rt, [string - 2, [0, 1]]), snd * (1/3)); // ostinato guitar records to channels 0 - 3
    } ! 4;
    /* karplus strong version - legacy sonification used for auditioning the piece
    oFreq = Select.kr((oString - (oStepper < 3)) > 0, [(oNote + fund).midicps, fund.midicps * Select.kr(oString, tuning)]);
    oPluckedStrings = { arg i;
        var string, isString, snd;
        string = 5 - i;
        isString = oTrig * (string <= oString) * (oString <= string);
        snd = Pluck.ar(WhiteNoise.ar(0.1), isString, 0.2,
            Latch.kr(oFreq.reciprocal, isString), 10, 0);
        Out.ar(Select.kr(rt, [string, [0, 1]]), snd);
    } ! 4;*/
    // endState - envelope has finished but cycles through switches until the final bass note is played
    endState = Latch.kr(1, trig * (env >= 1) * (switch > 0) * (oStepper <= 2) * (oStepper >= 2));
    endTrig = trig * endState * (oStepper >= 5);
    FreeSelf.kr(TDelay.kr(endTrig, (oDur * durUnit) + 10));
    // trigger switch and allow notes to pass until next 5 second interval (so interrupt always starts on a common multiple of 4 and 5)
    oSwitchTrig = (startState <= 0) * (endState <= 0) * oTrig * (PulseCount.kr(Changed.kr(oStepper), Changed.kr(switch)) > (4 - oEnv)) *
        TWChoose.kr(oTrig, [0, 1], [1 - (env >= 1), oEnv], 1);
    oSwitchArm = PulseCount.kr(oSwitchTrig, switchTrig); // switch trig armed
    oSwitchTrigDel = 20 - (currentDur % 20); // count down to switch
    oDur = Select.kr(oSwitchArm <= 0, [Clip.kr(oDur, 0, oSwitchTrigDel), oDur]); // clip dur if past switch
    oSwitchTrig = oTrig * (oDur >= oSwitchTrigDel) * (oSwitchArm > 0); // trigger the switch
    Poll.kr(oTrig, oDur, \oDur); // monitor duration of the ostinato note

    //~~~~~~~~~~~~~~~Interrupt~~~~~~~~~~~~~~~~
    // mute triggers when off
    iTrig = trig * switch;
    // select sound type: mute, harmonic, or open string (make first interuption not have percussion)
    iSound = TWChoose.kr(iTrig * TWChoose.kr(iTrig, [0, 1], [1, 2], 1), [0, 1, 2],
        Select.kr(PulseCount.kr(Changed.kr(switch)) > 1, [[1, 1, 0], [1, 1, 1]]), 1);
    // select duration
    iDur = Select.kr(iSound < 2,
        [1, TChoose.kr(iTrig * TWChoose.kr(iTrig, [0, 1], [1, 1], 1), [1, 2, 3])]);
    // select string as a stepper occasionally changing how many strings are skipped
    iString = Stepper.kr(iTrig, 0, 0, 5, TWChoose.kr(iTrig, [1, 2, 3], [3, 2, 1], 1));
    // play ensemble
    iEnsemble ={ |i| var string, isString, harm, freq, rel, fade, snd;
        string = 5 - i; // invert string number (since iString 1 is guitar string IV)
        isString = iTrig * (string <= (5 - iString)) * ((5 - iString) <= string); // check if string is triggered
        harm = TRand.kr(1 + iSound, iSound * 5 + 1, isString); // set based on open string or harmonic
        freq = fund.midicps * tuning[i] * harm; // calculate freq
        rel = Select.kr(switch, [2, 0]); // set release
        fade = EnvGen.kr(Env.asr(releaseTime: rel), // gate if sound is mute or with switch
            Latch.kr(Select.kr(iSound < 2 * switch, [0, 1]), isString + (Changed.kr(switch) * (1 - switch))));
        snd = SinOsc.ar(Latch.kr(freq, isString), 0, fade * (1/harm)); // simple sine tone with amp 1/harmonic number
        Out.ar(Select.kr(rt, [string + 16, [0, 1]]), snd * (1/12)); // interrupt ensemble records to channels 16 - 21
    } ! 6;
    iPerc = { |i| var string, isString, isStringDel, freq, snd;
        string = 5 - i; // invert string number (since iString 1 is guitar string IV)
        isString = iTrig * (string <= (5 - iString)) * ((5 - iString) <= string) * (iSound >= 2); // check if sound is percussion / mute
        isStringDel = TDelay.kr(isString, 0.01); // slight delay for envelope
        // play samples
        snd = PlayBuf.ar(1, TIRand.kr(percBufDelims[string], percBufDelims[string + 1] - 1, isStringDel),
            Latch.kr(1, isStringDel), isStringDel) * (1 - EnvGen.ar(Env.new([0, 1, 0], [0.01, 0.01]), isString));
        Out.ar(Select.kr(rt, [string + 10, [0, 1]]), snd * (1/2)); // interrupt ensemble records to channels 10 - 15
    } ! 6;
    iPluckedStrings = { |i| var string, isString, isStringDel, snd;
        string = 5 - i; //invert string number (since iString 1 is guitar string IV)
        isString = iTrig * (string <= (5 - iString)) * ((5 - iString) <= string); // check if string is triggered
        isStringDel = TDelay.kr(isString, 0.01); // slight delay for envelope
        // play samples and select open string, harmonic, or muted string
        snd = PlayBuf.ar(1, Select.kr(Latch.kr(iSound, isStringDel),
```

```
            [TIRand.kr(guitOpenBufDelims[string], guitOpenBufDelims[string + 1] - 1, isStringDel),
            TIRand.kr(guitHarmBufDelims[string], guitHarmBufDelims[string + 1] - 1, isStringDel),
            TIRand.kr(guitMuteBufDelims[string], guitMuteBufDelims[string + 1] - 1, isStringDel)
        ]), Latch.kr(1, isStringDel), isStringDel) * (1 - EnvGen.ar(Env.new([0, 1, 0], [0.01, 0.01]), isString));
        Out.ar(Select.kr(rt, [string + 4, [0, 1]]), snd * (1/3)); // interrupt ensemble records to channels 4 - 9
    } ! 6;
    /* karplus strong version - legacy sonification used for auditioning the piece
    iPluckedStrings =
    { |i| var string, isString, freq, snd;
        string = 5 - i;
        isString = iTrig * (i <= iString) * (iString <= i) * (iSound < 2);
        freq = fund.midicps * tuning[i] *
            TRand.kr(1 + iSound, iSound * 5 + 1, isString);
        snd = Pluck.ar(WhiteNoise.ar(0.1), isString, 0.2,
            Latch.kr(freq.reciprocal, isString), 10, 0);
        Out.ar(Select.kr(rt, [string + 4, [0, 1]]), snd);
    } ! 6;*/
    // trigger switch and allow notes to pass until next interval of 4 pulses then add 4 pulses (5 pulses = 1 second)
    iSwitchTrig = iTrig * TWChoose.kr(iTrig, [0, 1], [iEnv, 1], 1);
    iSwitchArm = PulseCount.kr(iSwitchTrig, switchTrig); // switch trig armed
    iSwitchTrigDel = 4 - (currentDur % 4); // countdown to switch
    iDur = Select.kr(iSwitchArm <= 0, [Clip.kr(iDur, 0, iSwitchTrigDel), iDur]); // clip dur if past switch
    iSwitchTrig = iTrig * (iDur >= iSwitchTrigDel) * (iSwitchArm > 0); // trigger the switch
    iDur = Select.kr(iSwitchTrig, [iDur, iDur + 4]); // add 4 pulses
    Poll.kr(iTrig, iDur, \iDur); // monitor duration of interrupt note

    //~~~~~~~~~~~~~~~Electronic Accompaniment~~~~~~~~~~~~~~~
    aDust = Dust.kr(10); // random triggers
    aLBuf = LocalBuf.new((SampleRate.ir / fund.midicps).trunc); // buffer for harmonic flickering sound
    // fill buf with random bursts of noise
    RecordBuf.ar(PinkNoise.ar(), aLBuf, run: Latch.ar(TWChoose.kr(aDust, [0, 1], [5, 1], 1), aDust + TDelay.kr(aDust, 0.01)));
    // play buf back at a rate equivalent to the bass note in the ostinato, select whether on or off on every ostinato note
    aFlicker = PlayBuf.ar(1, aLBuf, 1, Impulse.ar((fund + oBassNote).midicps), loop: 1) * Latch.ar(
        TWChoose.kr(aDust, [0, 1], [1, 2], 1), aDust) * (1 - switch) * TWChoose.kr(oTrig, [0, 1], [2, 1], 1) * 0.02;
    aBNoise = BrownNoise.ar(0.007) * (1 - switch); // brown noise during ostinato
    aWNoise = WhiteNoise.ar(0.002) * switch; // white noise during interrupt
    // sine tone beating that can change rate (or not) on every ostinato note (around fundamental for ostinato and a fifth down for interrupt
    aSinTrig = TWChoose.kr(oTrig, [0, 1], [1, 1], 1) + Changed.kr(switch);
    aSinBeat = (SinOsc.ar((fund - (switch * 5)).midicps) +
        SinOsc.ar((fund - (switch * 5)).midicps + 0.5 + Latch.ar(TRand.kr(0, 2.5, aSinTrig), aSinTrig))) * 0.05;
    aFadeIn = EnvGen.kr(Env.new([0, 1], [1])); // short fade at start of piece
    aFadeOut = EnvGen.kr(Env.cutoff(10), TDelay.kr(endTrig, oDur * durUnit)); // longer fade at end of piece
    Out.ar(Select.kr(rt, [22, [0, 1]]), aBNoise * aFadeIn * aFadeOut); // brown noise records to channel 22
    Out.ar(Select.kr(rt, [23, [0, 1]]), aWNoise * aFadeIn * aFadeOut); // white noise records to channel 23
    Out.ar(Select.kr(rt, [24, [0, 1]]), aSinBeat * aFadeIn * aFadeOut); // sine beating records to channel 24
    Out.ar(Select.kr(rt, [25, [0, 1]]), aFlicker * aFadeIn * aFadeOut); // harmonic flickering records to channel 25

    //~~~~~~~~~~~~~~~Score~~~~~~~~~~~~~~~
    // write score buffer for lilypond transcription
    count = PulseCount.kr(trig);
    BufWr.kr(Select.kr(switch,
        [[switch, oString, oNote, 0, oDur], [switch, iString, 0, iSound, iDur]]), scoreBuf, Select.kr(trig, [-1, count]));
    Poll.kr(trig, count, \scoreCount);

    //~~~~~~~~~~~~~~~FeedbackOut~~~~~~~~~~~~~~~
    LocalOut.kr([
        // feedback note trigger
        TDelay.kr(trig * (1 - endTrig), Select.kr(switch, [oDur, iDur]) * durUnit - ControlRate.ir.reciprocal),
        // feedback oString, oDurTemp, and oStepper for counts
        oString,
        oDurTemp - 1,
        oStepper,
        // feedback switchTrig
        TDelay.kr(Select.kr(switch, [oSwitchTrig, iSwitchTrig]), Select.kr(switch, [oDur, iDur]) * durUnit - ControlRate.ir.reciprocal),
        // feedback whether or not in start state
        PulseCount.kr(Changed.kr(oStepper)) <= 10
    ]);
});
)


/*
(
// Uncomment to test / play everything in real time - see ~generateData for comments of synth messages
var sample_index = 0, seed = 20170121;
var perc_allocs, guit_open_allocs, guit_harm_allocs, guit_mute_allocs, guit_bass_allocs;
var allocMSGs = {
    arg inFolder;
    var bufAllocs = [], bufDelims = [];
    PathName(inFolder).folders.sort({arg a, b; a.folderName[0..1].asInteger < b.folderName[0..1].asInteger }).collect({|folder|
        bufDelims = bufDelims.add(sample_index);
        folder.files.collect({|file|
            bufAllocs  = bufAllocs.add([\b_allocRead, sample_index, file.fullPath]);
            sample_index = sample_index+1;
        });
    });
    bufDelims = bufDelims.add(sample_index);
    [bufAllocs, bufDelims]
```

```
};

~dir = thisProcess.nowExecutingPath.dirname;
perc_allocs = allocMSGs.value(~dir +/+ "../samples/percussion/");
guit_open_allocs = allocMSGs.value(~dir +/+ "../samples/strings_open/");
guit_harm_allocs = allocMSGs.value(~dir +/+ "../samples/strings_harmonics/");
guit_mute_allocs = allocMSGs.value(~dir +/+ "../samples/strings_muted/");
guit_bass_allocs = allocMSGs.value(~dir +/+ "../samples/ostinato_bass/");

s.listSendBundle(0,
    perc_allocs[0] ++ guit_open_allocs[0] ++ guit_harm_allocs[0] ++ guit_mute_allocs[0] ++ guit_bass_allocs[0] ++
    [["/b_alloc", sample_index + 1, 10000, 5], ["/d_recv", ~ostinato_and_interrupt.asBytes(s);]]);


Synth.new(\ostinato_and_interrupt, [\scoreBuf, sample_index + 1, \seed, seed, \rt, 1,
    \percBufDelims, perc_allocs[1],
    \guitOpenBufDelims, guit_open_allocs[1],
    \guitHarmBufDelims, guit_harm_allocs[1],
    \guitMuteBufDelims, guit_mute_allocs[1],
    \guitBassBufDelims, guit_bass_allocs[1]
]);
)
*/
```

ostinato_and_interrupt_nrt_generator_function.scd

```
(
//~~FUNCTION THAT GENERATES THE PIECE (calls SynthDef(\ostinato_and_interrupt))
~generateData = {
    arg isLaunch = false, seed = 20170121; // set if ran on application launch and random seed
    var sample_index = 0; // init sample index
    var perc_allocs, guit_open_allocs, guit_harm_allocs, guit_mute_allocs, guit_bass_allocs; // allocation messages

    // this function reads all the subfolders creating allocation messages and an array that tells synth the range of each type of sample
    var allocMSGs = {
        arg inFolder;
        var bufAllocs = [];
        var bufDelims = [];
        PathName(inFolder).folders.sort({arg a, b; a.folderName[0..1].asInteger < b.folderName[0..1].asInteger }).collect({
            |folder|
            bufDelims = bufDelims.add(sample_index);
            folder.files.collect({|file|
                bufAllocs  = bufAllocs.add([0, [\b_allocRead, sample_index, file.fullPath]]); // allocation message
                sample_index = sample_index+1;
            });
        });
        bufDelims = bufDelims.add(sample_index); // start and end buffer index for each subfolder
        [bufAllocs, bufDelims]
    };

    // run the above function on each folder
    perc_allocs = allocMSGs.value(~dir +/+ "../samples/percussion/");
    guit_open_allocs = allocMSGs.value(~dir +/+ "../samples/strings_open/");
    guit_harm_allocs = allocMSGs.value(~dir +/+ "../samples/strings_harmonics/");
    guit_mute_allocs = allocMSGs.value(~dir +/+ "../samples/strings_muted/");
    guit_bass_allocs = allocMSGs.value(~dir +/+ "../samples/ostinato_bass/");

    // excecute everything in non real time, generate soundfile starting with 4 seconds of silence
    Score.recordNRT(
        perc_allocs[0] ++ guit_open_allocs[0] ++ guit_harm_allocs[0] ++ guit_mute_allocs[0] ++ guit_bass_allocs[0] ++
        [
            [0, [\b_alloc, sample_index + 1, 10000, 5]],
            [0, [\d_recv, ~ostinato_and_interrupt.asBytes(s);]],
            [4, [\s_new, \ostinato_and_interrupt, 10000, 0, 0, \scoreBuf, sample_index + 1, \seed, seed] ++
                [\percBufDelims, $[,] ++ perc_allocs[1] ++ [$]] ++
                [\guitOpenBufDelims, $[,] ++ guit_open_allocs[1] ++ [$]] ++
                [\guitHarmBufDelims, $[,] ++ guit_harm_allocs[1] ++ [$]] ++
                [\guitMuteBufDelims, $[,] ++ guit_mute_allocs[1] ++ [$]] ++
                [\guitBassBufDelims, $[,] ++ guit_bass_allocs[1] ++ [$]]
            ],
            [60 * 20, [\b_write, sample_index + 1, ~dir +/+ "gen_data_resources/ostinato_and_interrupt_data.wav", "WAV", "float"]],
            [60 * 20, [\c_set, 0, 0]]],
        ~dir +/+ "gen_data_resources/ostinato_and_interrupt_osc",
        ~dir +/+ "../audio/ostinato_and_interrupt.wav",
        headerFormat: "WAV", options: ServerOptions.new.numOutputBusChannels = 26, action: {

            // trim the multichannel audio file down to the correct size
            var datasf, insf, outsf, data, durSum, n, pad, newSize;

            ~appStatusString = "writing files";
            datasf = SoundFile.openRead(~dir +/+ "gen_data_resources/ostinato_and_interrupt_data.wav");
            datasf.readData(data = FloatArray.newClear(10000));
            datasf.close;
            data = data.clump(5).drop(1);
            durSum = 0;
            n = 0;
            while( {data[n][4] != 0}, {
                durSum = durSum + data[n][4];
                n=n+1;
            });
            insf = SoundFile.openRead(~dir +/+ "../audio/ostinato_and_interrupt.wav");
            outsf = SoundFile.new.headerFormat_(insf.headerFormat).numChannels_(
                insf.numChannels).sampleRate_(insf.sampleRate).sampleFormat_(insf.sampleFormat);
            outsf.openWrite(~dir +/+ "../audio/ostinato_and_interrupt_cut.wav");
            pad = insf.sampleRate * 4 * insf.numChannels;
            insf.readData(data = FloatArray.newClear(pad));
            (((durSum + (22 * 5))/5.0).ceil).do({
                insf.readData(data = FloatArray.newClear(insf.sampleRate * insf.numChannels));
                outsf.writeData(data)});
            insf.close;
            outsf.close;
            File.delete(~dir +/+ "../audio/ostinato_and_interrupt.wav");
            File.copy(~dir +/+ "../audio/ostinato_and_interrupt_cut.wav", ~dir +/+ "../audio/ostinato_and_interrupt.wav
                ");
            File.delete(~dir +/+ "../audio/ostinato_and_interrupt_cut.wav");

            // call transcriber function
            ~appStatusString = "generating lilypond";
            ~generateLilypond.value;

            // load the buffer for playback
            ~appStatusString = "loading buffer";
            Buffer.read(s, ~dir +/+ "../audio/ostinato_and_interrupt.wav", action: {
                |buf| ~totalDur = buf.duration;
                if(isLaunch == true, {~play = Synth.new(\play, [\buf, buf])}, {~play.set(\buf, buf)});
                ~appStatusFunc.stop;
                {~appStatus.string = "ready"}.defer
            });
    });
};
)

// uncomment below to run generator without gui
/*(
"ostinato_and_interrupt_generator_synthdef.scd".loadRelative(true, {
    ~dir = thisProcess.nowExecutingPath.dirname;
    ~appStatus = StaticText();
    ~generateData.value;
});
)*/
```

ostinato_and_interrupt_lilypond_generator_function.scd

```
(
//~~FUNCTION THAT GENERATES THE LILYPOND FILES
~generateLilypond = {
    var sf, data, notes, perc, parts, noteNames, durSum, lastState, lastDur, n, beatPos, lastBassNote, lGrey, dGrey,
        lilyRepeat, lilyTime;
    var inFile, outFile, inString, outString;
    sf = SoundFile.openRead(~dir +/+ "../supercollider/gen_data_resources/ostinato_and_interrupt_data.wav");
    sf.readData(data = FloatArray.newClear(10000)); sf.close;
    data = data.clump(5).drop(1);
    notes = []; perc = []; parts = [[], [], [], [], [], [], []];
    noteNames = ["d\'", "f\'", "a\'", "c\'\'", "e\'\'", "g\'\'"];
    durSum = 60; lastState = -1; lastDur = -1; n = 0; beatPos = 0; lGrey = 70; dGrey = 50; lastBassNote = 11;
    while( {data[n][4] != 0}, {
        var state, string, note, sound, dur;
        state = data[n][0]; string = data[n][1]; note = data[n][2]; sound = data[n][3]; dur = data[n][4];
        data[n].postln; durSum = durSum + dur; durSum.postln;
        if(lastState != state, {beatPos = 0}, {});
        for(0, dur - 1,{
            arg b;
            var lilyStem, lilyRhythmMark, lilyStartBeam, lilyEndBeam, lilyBar, sec, minString, secString, lilyTime,
                 lilyBracket, lilyNote;

            lilyStem = case
            {b == 0 && lastState == 0 && state == 1} {
                " \\override Staff.Stem #\'transparent = ##f "}
            {b == 0 && ((lastState == 1 && state == 0) || (lastState == -1))} {
                " \\override Staff.Stem #\'transparent = ##t "}
            {true} {""};

            lilyRhythmMark = case
            {state == 0} {""}
            {dur > 3} {""}
            {lastState != state} {" _" ++ dur.asInteger.asString ++ " "}
            {lastDur != dur} {" _" ++ dur.asInteger.asString ++ " "}
            {true} {""};

            lilyStartBeam = if(beatPos % 4 == 0, {" [ "}, {""});
            lilyEndBeam = if(beatPos % 4 == 3, {" ] "}, {""});
            beatPos = beatPos + 1;

            lilyBar = if(b == 0 && (lastState != state), {' \\bar \"||\" '}, {""});

            sec = (((durSum - dur) + b) / 5);
            minString = (sec.trunc / 60).trunc.asInteger.asString;
            secString = (sec.trunc % 60).asInteger.asString;
            if(secString.size == 1, {secString = "0" ++ secString}, {});
            lilyTime = case
            {(b == 0 && lastState == 0 && state == 1) || (sec % 12 == 0)} {
                ' \\mark \\markup{ \\fontsize #-2 \"' ++ minString ++ "\'" ++ secString ++ '\\"\" } '}
            {b == 0 && ((lastState == 1 && state == 0) || (lastState == -1))} {""}
            {true} {""};

            lilyBracket = case
            {b == 0 && state == 0 && string == 1 && (note != 7)} {
                var res = if(lastBassNote != note, {"\\bracketify "}, {""}); lastBassNote = note; res}
            {b == 0 && state == 0 && string == 0} {
                var res = if(lastBassNote != note, {"\\bracketify "}, {""}); lastBassNote = note; res}
            {true} {lastBassNote = lastBassNote; ""};


            lilyNote = case
            {(state == 0) && (b != 0)} {"s16 "}
            {state == 0 && string == 0 && note == 0} {lilyBracket ++ noteNames[string] ++ "16 "}
            {state == 0 && string == 1 && (note - 7) == 0} {lilyBracket ++ noteNames[string] ++ "16 "}
            {(state == 0) && (string <= 1)} {
                "\\once \\override NoteHead #\'stencil = #ly:text-interface::print " ++
                "\\once \\override NoteHead #\'text = \\markup { \\translate #\'(-0 . -0.8) \\whiteout \\pad-markup #0.5 " ++ '\"' ++
                if(string == 0, {note.asInteger.asString}, {(note - 7).asInteger.asString}) ++ '\" } ' ++ lilyBracket ++ noteNames[string] ++ "16 "}
            {state == 0 && (string >= 2)} {noteNames[string] ++ "16 "}

            {(state == 1) && (b != 0)} {"r16 " ++ lilyStartBeam ++ lilyEndBeam}
            {state == 1 && sound == 0} {noteNames[string] ++ "16 " ++ lilyRhythmMark ++ lilyStartBeam ++ lilyEndBeam}
            {state == 1 && sound == 1} {noteNames[string] ++ "16 " ++ "\\harmonic " ++ lilyRhythmMark ++ lilyStartBeam + lilyEndBeam}
            {state == 1 && sound == 2} {"\\xNote " ++ noteNames[string] ++ "16 " ++ lilyRhythmMark ++ lilyStartBeam ++ lilyEndBeam};

            notes = notes.add(lilyBar ++ lilyTime ++ lilyStem ++ lilyNote);

            perc = perc.add(if(sound == 2 || (state == 0),
                {lilyBar ++ lilyTime ++ lilyStem ++ lilyNote},
                {lilyBar ++ lilyTime ++ " \\once \\override Stem.color = #(x11-color \'grey" ++ lGrey.asString ++ ") " ++ lilyStem ++
                    " \\once \\override NoteHead.color = #(x11-color \'grey" ++ lGrey.asString ++ ") " ++ lilyNote}));

            for(0, 5, {
                arg p; var grey;
                grey = if(sound == 2, {dGrey}, {lGrey});
                parts[p] = parts[p].add(if(string == p || (state == 0),
                    {lilyBar ++ lilyTime ++ lilyStem ++ lilyNote},
                    {lilyBar ++ lilyTime ++ " \\once \\override Stem.color = #(x11-color \'grey" ++ grey.asString ++ ") " ++ lilyStem ++
                        " \\once \\override NoteHead.color = #(x11-color \'grey" ++ grey.asString ++ ") " ++ lilyNote}));
            });
        });
        lastState = state;
        lastDur = dur;
        n = n + 1;
    });

    lilyRepeat = "\\repeat unfold " ++ ((durSum-60)/5/12).asInteger.asString ++ " { \\repeat unfold 59 { s16 \\noBreak } s16 \\break } ";
    lilyTime = "\\time 60/16 ";

    inFile = File(~dir +/+ "../lilypond/ostinato_and_interrupt_lilypond_score_template.ly", "r");
    inString = inFile.readAllString;
    inFile.close;

    outFile = File(~dir +/+ "../lilypond/ostinato_and_interrupt_lilypond_guitar_part.ly", "w");
    outString = "<< " ++ lilyRepeat ++ lilyTime ++
       "{ \\override Staff.Rest #\'transparent = ##t " ++ notes.join ++ ' \\bar \"|.\" } >>';
    outFile.write(inString.replace("%<<music>>", outString).replace("piece = " ++ '\"part\"', "piece = " ++ '\"guitar/all\"'));
    outFile.close;

    outFile = File(~dir +/+ "../lilypond/ostinato_and_interrupt_lilypond_percussion_part.ly", "w");
    outString = "<< " ++ lilyRepeat ++ lilyTime ++
        "{ \\override Staff.Rest #\'transparent = ##t " ++ perc.join ++ ' \\bar \"|.\" } >>';
    outFile.write(inString.replace("%<<music>>", outString).replace("piece = " ++ '\"part\"', "piece = " ++ '\"percussion\"'));
    outFile.close;

    for(0, 5, {
        arg p, staff;
        outFile = File(~dir +/+ "../lilypond/ostinato_and_interrupt_lilypond_ensemble_part_" ++ (6 - p).asString ++ ".ly", "w");
        staff = Array.fill(6, {|i| if(i == (5 - p), {"#f "}, {"(x11-color \'grey" ++ lGrey.asString ++ ") "})}).join;
        staff = "\\override Staff.StaffSymbol.stencil = #(color-staff-lines " ++ staff ++ ")";
        outString = "<< " ++ lilyRepeat ++ lilyTime ++
            "{" ++ staff ++ " \\override Staff.Rest #\'transparent = ##t " ++ (parts[p]).join ++ ' \\bar \"|.\" } >>';
        outFile.write(inString.replace("%<<music>>", outString).replace(
            "piece = " ++ '\"part\"', "piece = " ++ '\"ensemble part ' ++ (6 - p).asString ++ '\"'));
        outFile.close;
    });
};
)
// uncomment below generate lilypond files without gui (requires resouces to exist)
/*(
~dir = thisProcess.nowExecutingPath.dirname;
~generateLilypond.value
)*/
```

ostinato_and_interrupt_player_synthdef.scd

```
(
//~~SYNTHDEF THAT PLAYS THE PIECE AND ACCEPTS CONTROL FROM THE GUI
SynthDef(\play, {
    arg buf = 0, env, playRate = 0, startPos = 0, startTrig = 0, curDur,
    goVol = #[1, 1, 1, 1, 1, 1], giVol = #[1, 1, 1, 1, 1, 1], pVol = #[1, 1, 1, 1, 1, 1], eVol = #[1, 1, 1, 1, 1, 1], aVol = #[1, 1, 1, 1, 1, 1],
    goMute = #[1, 1, 1, 1, 1, 1], giMute = #[1, 1, 1, 1, 1, 1], eMute = #[1, 1, 1, 1, 1, 1], pMute = #[1, 1, 1, 1, 1, 1], aMute = #[1, 1, 1, 1, 1, 1],
    goPan = #[0, 0, 0, 0, 0, 0], giPan = #[0, 0, 0, 0, 0, 0], ePan = #[0, 0, 0, 0, 0, 0], pPan = #[0, 0, 0, 0, 0, 0], aPan = #[0, 0, 0, 0, 0, 0],
    masterVolGroups = #[1, 1, 1, 1, 1], masterMuteGroups = #[1, 1, 1, 1, 1],
    allMasterVol = 1, allMasterMute = 1;
    var phasor, player;
    var guitarOTracks, guitarITracks, percussionTracks, ensembleTracks, accompTracks;
    var guitarOTracksPanned, guitarITracksPanned, percussionTracksPanned, ensembleTracksPanned, accompTracksPanned;
    var guitarOMaster, guitarIMaster, percussionMaster, ensembleMaster, accompMaster;
    var allMaster;
    var imp, delimp;

    player = PlayBuf.ar(26, buf, playRate, startTrig, startPos * BufFrames.kr(buf));
    phasor = Phasor.ar(startTrig,
        Select.kr(playRate, [0, BufRateScale.kr(buf)]),
        0, BufFrames.kr(buf), startPos * BufFrames.kr(buf));

    guitarOTracks = { |i| var string = 5 - i; player[string - 2] * goVol[i] * goMute[i] } ! 4;
    guitarITracks = { |i| var string = 5 - i; player[string + 4] * giVol[i] * giMute[i] } ! 6;
    percussionTracks = { |i| var string = 5 - i; player[string + 10] * pVol[i] * pMute[i] } ! 6;
    ensembleTracks = { |i| var string = 5 - i; player[string + 16] * eVol[i] * eMute[i] } ! 6;
    accompTracks = { |i| player[i + 22] * aVol[i] * aMute[i] } ! 4;

    guitarOTracksPanned = { |i| Pan2.ar(guitarOTracks[i], goPan[i]) } ! 4;
    guitarITracksPanned = { |i| Pan2.ar(guitarITracks[i], giPan[i]) } ! 6;
    percussionTracksPanned = { |i| Pan2.ar(percussionTracks[i], pPan[i]) } ! 6;
    ensembleTracksPanned = { |i| Pan2.ar(ensembleTracks[i], ePan[i]) } ! 6;
    accompTracksPanned = { |i| Pan2.ar(accompTracks[i], aPan[i]) } ! 4;

    guitarOMaster = Mix.new(guitarOTracksPanned) * masterVolGroups[0] * masterMuteGroups[0];
    guitarIMaster = Mix.new(guitarITracksPanned) * masterVolGroups[1] * masterMuteGroups[1];
    percussionMaster = Mix.new(percussionTracksPanned) * masterVolGroups[2] * masterMuteGroups[2];
    ensembleMaster = Mix.new(ensembleTracksPanned) * masterVolGroups[3] * masterMuteGroups[3];
    accompMaster = Mix.new(accompTracksPanned) * masterVolGroups[4] * masterMuteGroups[4];

    allMaster = Mix.new([guitarOMaster, guitarIMaster, percussionMaster, ensembleMaster, accompMaster]) * allMasterVol * allMasterMute;
    Out.ar(0, allMaster);

    curDur = ((A2K.kr(phasor) / BufFrames.kr(buf)) * BufDur.kr(buf) * 5).trunc;
    //Optional click - uncomment and send to an output not used to give the guitarist a click track.
    //Out.ar(2, 10 * BPF.ar(WhiteNoise.ar * EnvGen.kr(Env.perc(0.01, 0.1), curDur % 4 <= 0), 440 * ((curDur % 20 <= 0) + 1), 0.02));
    SendTrig.kr(Changed.kr(curDur), 0, curDur);
    imp = Impulse.kr(10);
    delimp = Delay1.kr(imp);
    SendReply.kr(imp,
        '/allMasterLevels',
        values: [Amplitude.kr(allMaster)]);
    SendReply.kr(imp,
        '/groupMasterLevels',
        values: [
            Amplitude.kr(guitarOMaster) ++ Amplitude.kr(guitarIMaster) ++
            Amplitude.kr(percussionMaster) ++ Amplitude.kr(ensembleMaster) ++ Amplitude.kr(accompMaster)]);
    SendReply.kr(imp,
        '/groupTrackLevels',
        values: [Amplitude.kr(guitarOTracks) ++ Amplitude.kr(guitarITracks) ++
            Amplitude.kr(percussionTracks) ++ Amplitude.kr(ensembleTracks) ++ Amplitude.kr(accompTracks)]);
}).add;
)
```

ostinato_and_interrupt_gui_generator_function.scd

```
(
//~~FUNCTION THAT GENERATES THE GUI
~generateGUI = {
    var win, clockStringFunc, metronomeStringFunc, metronomeColorFunc, masterView, faderViews, tabs;
    var tabButtonReset, masterButton, guitarOButton, guitarIButton, percButton, ensembleButton, accompButton, startPos = 0;
    var groupNames = ["guitar - ostinato", "guitar - interrupt", "percussion", "interrupt highlights", "fields / beats / flicker"], groupAbbr = ["go", "gi", "p", "e", "a"];
    var accompNames = ["brown noise", "white noise", "sine beating", "flicker"];
    var goVol, giVol, pVol, eVol, aVol, goPan, giPan, pPan, ePan, aPan, goMute, giMute, pMute, eMute, aMute, volGroups, panGroups, muteGroups;
    var masterMuteGroups, masterVolGroups;

    goVol = giVol = pVol = eVol = aVol = [0.8, 0.8, 0.8, 0.8, 0.8, 0.8];
    goMute = giMute = pMute = eMute = aMute = [1, 1, 1, 1, 1, 1];
    goPan = giPan = pPan = ePan = aPan = [0, 0, 0, 0, 0, 0];
    volGroups = [goVol, giVol, pVol, eVol, aVol];
    muteGroups = [goMute, giMute, pMute, eMute, aMute];
    panGroups = [goPan, goPan, pPan, ePan, aPan];
    masterMuteGroups = [1, 1, 1, 1, 1];
    masterVolGroups = [0.8, 0.8, 0.8, 0.8, 0.8];

    clockStringFunc = {
        arg div;
        var min, sec;
        sec = (div / 5).trunc;
        min = (sec / 60).asInteger.asString;
        if(min.size == 1, {min = "0" ++ min}, {});
        sec =  (sec % 60).asInteger.asString;
        if(sec.size == 1, {sec = "0" ++ sec}, {});
        min ++ ":" ++ sec
    };
    // [-30, -105, -104] and [-30, -105, -113] are unicode inverse bullet and normal bullet, respectively
    metronomeStringFunc = { arg div; case {div % 20 < 2}
        {[-30, -105, -104].collect({arg int; int.asAscii}).as(String)} {div % 4 < 2}
        {[-30, -105, -113].collect({arg int; int.asAscii}).as(String)} {true} {" "} };
    metronomeColorFunc = { arg div; case {div % 20 < 2} {Color.red} {div % 4 < 2} {Color.blue} {true} {Color.black} };

    ~appStatusFunc = Task({
        loop {
            {~appStatus.string = ~appStatusString ++ "*"}.defer;
            0.5.wait; {~appStatus.string = ~appStatusString ++ "* *"}.defer;
            0.5.wait; {~appStatus.string = ~appStatusString ++ "* * *"}.defer;
            0.5.wait; {~appStatus.string = ~appStatusString ++ "* * * *"}.defer;
            0.5.wait; {~appStatus.string = ~appStatusString ++ "* * * * *"}.defer;
            0.5.wait;
        }
    });

    win = Window("ostinato and interrupt", Rect(500, 500, 1100, 500), false).front;
    masterView = {
        var view, masterIndicators, master, generator, transport, ranSeed, startPosText, pauseButton, clock, metronome;

        OSCFunc({ arg msg, time; {
                clock.string = clockStringFunc.value(msg[3]);
                metronome.stringColor = metronomeColorFunc.value(msg[3]);
                metronome.string = metronomeStringFunc.value(msg[3])}.defer;
        },'/tr', s.addr);
        OSCFunc.new({arg msg; {
            {|i| masterIndicators[i].value = msg[3 + i].ampdb.linlin(-40, 0, 0, 1)} ! 2}.defer},
        '/allMasterLevels', s.addr);

        view = View(win);
        masterIndicators = [LevelIndicator(), LevelIndicator()];
        master = HLayout(
            VLayout(
                HLayout(
                    Slider(view).value_(0.8).action_({|v| ~play.set(\allMasterVol, v.value * 1.25)}),
                    masterIndicators[0], masterIndicators[1]),
                Button(view).states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_(
                    {|v| ~play.set(\allMasterMute, (1 - v.value).abs)}),
                StaticText(view).string_("          master          ").align_(\center),
                StaticText(view).string_("(all)").align_(\center)),
            nil);
        generator = HLayout(
            Button(view).states_([["generate"]]).action_({
                ~appStatusString = "generating data";
                ~appStatusFunc.start;
                ~generateData.value(seed: ranSeed.string.asInteger);
            }),
            ranSeed = TextField(view).string_("20170121"),
            Button(view).states_([["reset seed"]]).action_({ ranSeed.string = "20170121"}),
            Button(view).states_([["random seed"]]).action_({ ranSeed.string = 50000000.rand.asString}),
            [~appStatus = StaticText(view).string_("status: ready"), stretch: 1],
            nil);
        transport = HLayout(
            Button(view).states_([["play", Color.black], ["stop", Color.black, Color.grey]]).action_(
                {| pState |
                    pauseButton.value = 0;
                    if(pState.value == 0, {~play.set(\playRate, 0, \startTrig, 0);
                        clock.string = clockStringFunc.value((startPos * ~totalDur * 5).asInteger)},
                        {~play.set(\startPos, startPos, \playRate, 1, \startTrig, 1)})}),
            pauseButton = Button(view).states_([["pause", Color.black], ["pause", Color.black, Color.grey]]).action_(
                {| pState |
                    if(pState.value == 1, {~play.set(\playRate, 0)},{~play.set(\playRate, 1)})}),
            StaticText(view).string_("start time"),
            [Slider(view, Rect(0, 0, 30, 5)).action_(
                {|pos|
                    var min, sec;
                    startPosText.string = clockStringFunc.value((pos.value * ~totalDur * 5).asInteger);
                    startPos = pos.value;
            }), stretch: 1],
            startPosText = StaticText(win).string_("00:00").font_(Font("Liberation Mono", 15)),
            nil);
        view.layout_(HLayout(master,
            [VLayout(generator, nil,
                HLayout(clock = StaticText(win).string_("00:00").font_(Font("Liberation Mono", 200)),
                StaticText(win).string_("|").font_(Font("Liberation Mono", 200)),
                metronome = StaticText(win).string_([-30, -105, -104].collect({arg int; int.asAscii}).as(String)).font_
                     (Font("Liberation Mono", 300)).stringColor_(Color.red)),
                nil, transport
    ), alignment: \top])) };
    faderViews = { |group|
        var view, masterIndicators, trackIndicators, master, tracks;
        view = View(win);
        masterIndicators = {LevelIndicator()} ! 10;
        trackIndicators = {LevelIndicator()} ! 26;

        OSCFunc.new({arg msg; {
            {|i| masterIndicators[i].value = msg[3 + i].ampdb.linlin(-40, 0, 0, 1)} ! 10}.defer},
        '/groupMasterLevels', s.addr);
        OSCFunc.new({arg msg; {
            {|i| trackIndicators[i].value = msg[3 + i].ampdb.linlin(-40, 0, 0, 1)} ! 26}.defer},
        '/groupTrackLevels', s.addr);

        master = HLayout(
            VLayout(
                [HLayout(
                    Slider(view).value_(0.8).action_(
                        {|v| masterVolGroups[group] = v.value * 1.25; ~play.set(\masterVolGroups, masterVolGroups)}),
                    masterIndicators[group * 2],
                    masterIndicators[group * 2 + 1]), stretch: 2],
                Button(view).states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_(
                    {|v| masterMuteGroups[group] = (1 - v.value).abs; ~play.set(\masterMuteGroups, masterMuteGroups)}),
                StaticText(view).string_("             master             ").align_(\center),
                StaticText(view).string_("("++groupNames[group]++")").align_(\center)
            ),
            nil);
        tracks = { |part|
            HLayout(
                VLayout(
                    HLayout(
                        Slider(view).value_(0.8).action_(
                            {|v| volGroups[group][part] = v.value * 1.25; ~play.set(groupAbbr[group] ++ "Vol",
                                volGroups[group])}),
                        trackIndicators[group * 6 + part - if(group > 0, {2}, {0})]),
                    Button(view).states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_(
                        {|v| muteGroups[group][part] = (1 - v.value).abs; ~play.set(groupAbbr[group] ++ "Mute",
                            muteGroups[group])}),
                    StaticText(view).string_("pan").align_(\center),
                    Knob(view).value_(0.5).action_(
                        {|v| panGroups[group][part] = v.value * 2 - 1; ~play.set(groupAbbr[group] ++ "Pan", panGroups[
                            group])}),
                    StaticText(view).string_(
                        if(group == 4, {accompNames[part]}, {(6 - part).asString})).align_(\center)
                ),
                nil)
        } ! if((group == 0) || (group == 4), {4}, {6});
        view.layout_(HLayout(master, nil, *tracks))} ! 5;
    tabButtonReset = {masterButton.value = 1;
        guitarOButton.value = 1; guitarIButton.value = 1; percButton.value = 1; ensembleButton.value = 1; accompButton.
            value = 1;};
    win.layout = VLayout(
        HLayout(
            masterButton = Button().states_([["master controls", Color.white, Color.grey], ["master controls", Color.
                black]]).action_(
                {tabButtonReset.value; masterButton.value = 0; tabs.index = 0 }).value_(0),
            guitarOButton = Button().states_([["guitar (ostinato)", Color.white, Color.grey], ["guitar (ostinato)",
                Color.black]]).action_(
                {tabButtonReset.value; guitarOButton.value = 0; tabs.index = 1 }).value_(1),
            guitarIButton = Button().states_([["guitar (interrupt)", Color.white, Color.grey], ["guitar (interrupt)",
                Color.black]]).action_(
                {tabButtonReset.value; guitarIButton.value = 0; tabs.index = 2 }).value_(1),
            percButton = Button().states_([["percussion", Color.white, Color.grey], ["percussion", Color.black]]).
                action_(
                {tabButtonReset.value; percButton.value = 0; tabs.index = 3 }).value_(1),
            ensembleButton = Button().states_([["interrupt highlights", Color.white, Color.grey], ["interrupt
                highlights", Color.black]]).action_(
                {tabButtonReset.value; ensembleButton.value = 0; tabs.index = 4 }).value_(1),
            accompButton = Button().states_([["fields / beats / flicker", Color.white, Color.grey],
                ["fields / beats / flicker", Color.black]]).action_(
                {tabButtonReset.value; accompButton.value = 0; tabs.index = 5 }).value_(1)),
        tabs = StackLayout(masterView.value, faderViews[0], faderViews[1], faderViews[2], faderViews[3], faderViews[4])
            );
};
)
```

ostinato_and_interrupt_lilypond_score_template.ly

```
\version "2.18.2"
\paper {
  top-system-spacing =
  #'((basic-distance . 15 )
  (minimum-distance . 15 )
  (padding . 0 )
  (stretchability . 0))

  #(set-paper-size "a4" 'portrait)
  system-system-spacing =
  #'((basic-distance . 24)
  (minimum-distance . 24)
  (padding . 0)
  (stretchability . 0))

  min-systems-per-page = 8
  max-systems-per-page = 8

  print-page-number = ##t
  oddHeaderMarkup = \markup \fill-line { " " }
  evenHeaderMarkup = \markup \fill-line { " " }
  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 {
  \on-the-fly #not-first-page
    \fill-line {
  \concat {
  "-"
      \fontsize #1.5
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
    "-"
      }
    }
  }
}
\header {
  title = \markup { \normal-text \italic {ostinato and interrupt}}
  piece = "part"
  opus = \markup { \concat {"version generated: "} #(strftime "%Y.%m.%d" (localtime (current-time)))}
  composer = "michael winter (mexico city, mx; 2017)"
  tagline = ""
}
#(set-global-staff-size 16)
\layout {
  indent = 0.0\cm
  ragged-right = ##t
  \context {
    \Staff
      \override StaffSymbol.line-count = #6
      \override StaffSymbol.staff-space = #1.8
      \override Beam.positions = #'(-5 . -5)
      \override Stem.direction = #DOWN
      \override Stem.stemlet-length = #1
      \override Beam.breakable = ##t
      \remove "Clef_engraver"
      \remove "Time_signature_engraver"
    }
}

#(define-public ((color-staff-lines . rest) grob)

   (define (index-cell cell dir)
     (if (equal? dir RIGHT)
         (cdr cell)
         (car cell)))

   (define (index-set-cell! x dir val)
     (case dir
       ((-1) (set-car! x val))
       ((1) (set-cdr! x val))))

   (let* ((common (ly:grob-system grob))
          (span-points '(0 . 0))
          (thickness (* (ly:grob-property grob 'thickness 1.0)
                        (ly:output-def-lookup (ly:grob-layout grob) 'line-thickness)))
          (width (ly:grob-property grob 'width))
          (line-positions (ly:grob-property grob 'line-positions))
          (staff-space (ly:grob-property grob 'staff-space 1))
          (line-stencil #f)
          (total-lines empty-stencil)
          ;; use a local copy of colors list, since
          ;; stencil creation mutates list
          (colors rest))

     (for-each
      (lambda (dir)
        (if (and (= dir RIGHT)
                 (number? width))
            (set-cdr! span-points width)
            (let* ((bound (ly:spanner-bound grob dir))
                   (bound-ext (ly:grob-extent bound bound X)))

              (index-set-cell! span-points dir
                               (ly:grob-relative-coordinate bound common X))
              (if (and (not (ly:item-break-dir bound))
                       (not (interval-empty? bound-ext)))
                  (index-set-cell! span-points dir
                                   (+ (index-cell span-points dir)
                                      (index-cell bound-ext dir))))))
        (index-set-cell! span-points dir (- (index-cell span-points dir)
                                            (* dir thickness 0.5))))
      (list LEFT RIGHT))

      (set! span-points
            (coord-translate span-points
                             (- (ly:grob-relative-coordinate grob common X))))
      (set! line-stencil
            (make-line-stencil thickness (car span-points) 0 (cdr span-points) 0))

      (if (pair? line-positions)
          (for-each (lambda (position)
                      (let ((color (if (pair? colors)
                                       (car colors)
                                       #f)))
                        (set! total-lines
                              (ly:stencil-add
                               total-lines
                               (ly:stencil-translate-axis
                                (if (color? color)
                                    (ly:stencil-in-color line-stencil
                                                         (first color)
                                                         (second color)
                                                         (third color))
                                    line-stencil)
                                (* position staff-space 0.5) Y)))
                        (and (pair? colors)
                             (set! colors (cdr colors)))))
                    line-positions)
          (let* ((line-count (ly:grob-property grob ’line-count 5))
                 (height (* (1- line-count) (/ staff-space 2))))
            (do ((i 0 (1+ i)))
                ((= i line-count))
              (let ((color (if (and (pair? colors)
                                    (> (length colors) i))
                               (list-ref colors i)
                               #f)))
                (set! total-lines (ly:stencil-add
                                   total-lines
                                   (ly:stencil-translate-axis
                                    (if (color? color)
                                        (ly:stencil-in-color line-stencil
                                                             (first color)
                                                             (second color)
                                                             (third color))
                                        line-stencil)
                                    (- height (* i staff-space)) Y)))))))
      total-lines))

#(define-public (bracket-stencils grob)
  (let ((lp (grob-interpret-markup grob (markup #:fontsize 3.5 #:translate (cons -0.3 -0.5) "[")))
        (rp (grob-interpret-markup grob (markup #:fontsize 3.5 #:translate (cons -0.3 -0.5) "]"))))
    (list lp rp)))

bracketify = #(define-music-function (parser loc arg) (ly:music?)
   (_i "Tag @var{arg} to be parenthesized.")
#{
  \once \override ParenthesesItem.stencils = #bracket-stencils
  \parenthesize $arg
#})

{
  \new Score
  \with {
    \remove "Bar_number_engraver"
    proportionalNotationDuration = #(ly:make-moment 1 16)
  }
  <<
  \new Staff
  \with {
    \remove "Stem_engraver"
  }
  %<<music>>
  >>
}
```