# bitKlavier

## *Manual & Documentation*

…

Noah Fishman & Dan Trueman, 2018–

v3.0

# Table of Contents

*If viewing in Google Drive, click “View > Open Document Outline” for a scrolling sidebar outline*

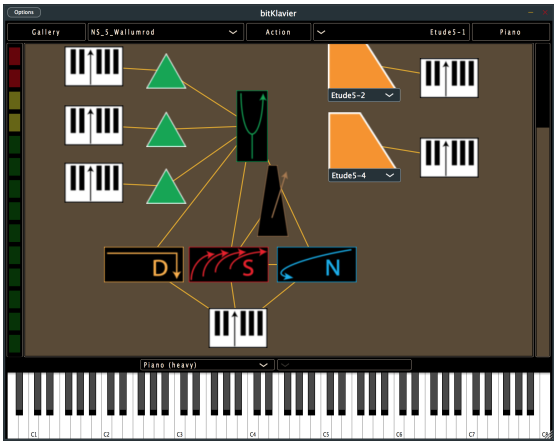
# Introduction

At the heart of bitKlavier is an analogy to the acoustic piano. Press any key on a piano and it triggers a hammer, which strikes a string. From the resulting vibration, we hear a pitch. Preparing a piano is the process by which we intervene with that straightforward linear system by physically altering the motion of the piano strings. We traditionally prepare a piano by placing screws, forks, coins, or really any object in the piano strings in order to change their sound: We insert a step between the hammer strike and the acoustic sound that emerges. Through preparation, we can turn a familiar and homogenous instrument into

something idiosyncratic, percussive or metallic, muted or dark. Yet under the hands, the instrument is still a piano.

If we understand the algorithm of the digital piano (key, MIDI signal, sample playback) in the same way that we understand the inner workings of the acoustic piano (key, hammer, vibrating string), we can think about altering the playback algorithm as a “digital preparation.” We’re inserting an additional step between the “hammer strike” (the MIDI signal) and the vibrating of the “string” (the playback of the sample).

The prepared acoustic piano has many limitations: A string can never vibrate without the impulse of the hammer, one string will never produce two pitches, and one pitch can never trigger another. bitKlavier provides an environment where virtual “screws” alter the behavior of the virtual hammer and string in ways that are impossible with the acoustic piano. bitKlavier is not only a sampler, effects unit, or sequencer: It is an instrument, responsive to the touch of the player, infinitely tweakable and refinable. It is an environment where performers and composers can build an instrument that takes the concept of the prepared piano into a new realm.




# New Features

## New in v2.5

- Blendrónic, a new preparation type

- Event targeting for Synchronic, Nostalgic, and Blendrónic, via Keymap
- MIDI Input device selection, via Keymap
- MIDI Out, for Synchronic
- NoteOn/Off invert option in Keymap
- Pattern rotate button for Synchronic / Blendronic multi-sliders
- Numerous bug fixes, and new bugs!

## New in v2.5.2

- Active preparation highlighting: preparations that are currently active will blink, making it easier to track what is going on in a complex Piano
- Active keys in selected Keymap will be displayed on main (bottom) keyboard, to make it easier to track prepared keys
- ToolTips can be turned on/off (they were always on before)
- Transposition "uses Tuning" option in Direct, Nostalgic, Synchronic; prior to this, transposition values were always interpreted literally, relative to main note; now they can be filtered by whatever Tuning is connected, so that a Tuning system can prevail in an entire preparation without micromanaging transposition values.
- Multislider (in Synchronic and Blendrónic) completely rewritten, should behave much better now, and will also save "gaps" in the slider values now
- bug fixes that were causing crashes with menus in plugin formats, some SoundFont loading bugs
- bug fixes with Nostalgic; envelopes for overlapping reverse and undertow notes were not behaving properly, and now do

## New in v2.6

- SoundFonts can be assigned to individual preparations
- Piano and Gallery menus are now sorted alphabetically
- New “music” folder that includes *Mikroetudes*, *Nostalgic Synchronic*, and the new *Machines for Listening* (by Dan) PDFs now part of the installer
- Numerous bug fixes, especially for plugin formats that were crashing

## New in v2.7

- Undo
- Harmonizer mappings in Keymap
- “Ignore sustain” option in Keymap
- “Use as sustain pedal” option in Keymap

## New in v2.8

- Modification smoothing
- Modification incrementing

- Modification alternating
- Velocity filtering in Direct
- Blendrónic input gain sliders for Direct, Synchronic, and Nostalgic
- All gain sliders changed to logarithmic dBFS scaling

### New in v2.9

- Velocity curves (in Keymap), and updates to back-end velocity handling
- Direct-from-disk sample streaming
- Flexible resource paths for sample folders, and ability to open any folder with consistently named audio files for playback

### New in v3.0

- Resonance, a new preparation type that flexibly models the sympathetic resonance of the piano
- Equalizer unit on the mixed output, to tailor the spectral balance depending on speakers, performance venue, and so on
- Reworking of how resonance-release samples are handled, dependent on both attack velocity and time since attack.
- The bitKlavier Grand: a sample library of a Steinway D, created for bitKlavier
- The Dunleavy Double Seconds: a sample library of steel pans, created for bitKlavier

# Implementation

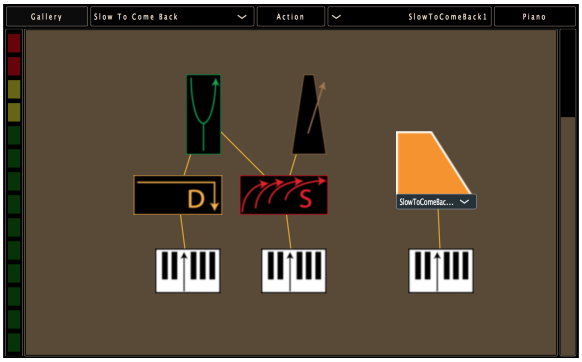
As a compositional playground and performance laboratory, bitKlavier can be used to explore tuning, rhythm, tonality, polyphony, counterpoint, and instrument design itself. The instrument is designed so that preparations can be fluidly connected and tweaked, and ideas can quickly be brought to life. Here are a few ideas which could be built within minutes using bitKlavier's preparations:

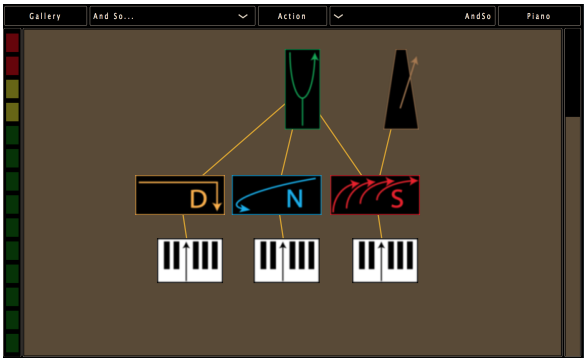
- A Piano where each hand is playing in different tunings. For example, everything above middle C is tuned in Pythagorean tuning, while everything below middle C is tuned to Equal Temperament.
- A Piano where the entire pitch structure is inverted: As you ascend the keyboard, the pitch becomes lower, and vice versa for descending the keyboard.

- A Piano where the white keys play dyads of major 3rds, and the black keys all play dyads of perfect 4ths.
- A Piano where every time a note is released, a sympathetic “Nostalgic” triad pitched slightly higher plays in reverse.
- A “Synchronic” sequence where, when a cluster of notes is played, a metronomic pulse is activated based on the tempo of your playing in real time.

bitKlavier can also be used in educational settings, and has been used to teach students about the design of the piano keyboard as a functional interface. When combined with bitSuite, a collection of educational apps with a similar interface to bitKlavier, it can help teach about tuning, temperaments, and rhythm. Much of this musicological material is explored in Dan Trueman’s course Reinventing The Piano, available online through the Kadenze platform.

bitKlavier has also been used by musicians for whom making music on a traditional piano has insurmountable physical obstacles. Galleries in bitKlavier can be designed so triads can be played with one finger, long arpeggios can be played with a two-note gesture, and every note is doubled an octave lower. As a customizable instrument that uses the traditional piano interface, bitKlavier can largely fit the needs of the composer/performer while remaining distinctly itself.



# Installation and Setup

bitKlavier is available for OSX (VST/AU and standalone), Windows (VST and standalone), and iOS devices. Individual installation instructions are below. For detailed desktop, laptop, or iOS installation instructions, visit the bitKlavier download page.

## Mac (v2.4 and higher)

Download the installer and run it!

You may get a warning from OSX saying this is from an “unidentified developer” or some such; typically, if you control-click on the installer, you can select “open” and it will open fine, or you may need to adjust your security settings in the OSX System Preferences.

The installer will install the following:

- a folder called “bitKlavier” in your /Applications directory, with the following in it:
    - the bitKlavier application
    - a “samples” folder with the core piano samples for bitKlavier
    - a “galleries” folder; galleries saved here will automatically show up in bitKlavier
    - a “pianos” folder; individual pianos saved here will automatically show up
    - a “preparations” folder; individual preparations are saved in subfolders here and will show up automatically
    - a “soundfonts” folder; soundfonts placed here will automatically show up in bitKlavier
    - the manual
- VST, VST3, and AU plugins in the /Library/Audio/Plug-Ins folder

Don’t move or rename the /Applications/bitKlavier folder or any of the folders inside of it; bitKlavier needs to have it there to find samples and galleries.

As of v2.9, you can move the “samples” and “soundfonts” directories as needed, but bitKlavier will need to know where you put them; set that in the Preferences.

## Windows (v2.4 and higher)

Download the installer and run it! Allow the installer to install in your Documents directory.

The installer will install the following:

- a folder called “bitKlavier” in your Documents directory, with the following in it:
    - the bitKlavier application (.exe)
    - a “samples” folder with the core piano samples for bitKlavier
    - a “soundfonts” folder; soundfonts placed here will automatically show up in bitKlavier
    - a “galleries” folder; galleries saved here will automatically show up in bitKlavier
    - a “pianos” folder; individual pianos saved here will automatically show up
    - a “preparations” folder; individual preparations are saved in subfolders here and will show up automatically
    - the manual
- VST2 (.dll) in C:\Program Files\Common Files\VST2
- VST3 (.vst3) in C:\Program Files\Common Files\VST3

Don’t move or rename the Applications\bitKlavier folder or any of the folders inside of it, or the Documents\galleries folder; bitKlavier needs to have them in those places to find samples and galleries.

As of v2.9, you can move the “samples” and “soundfonts” directories as needed, but bitKlavier will need to know where you put them; set that in the Preferences.

## iOS

bitKlavier is also available on iOS (iPhone and iPad) and can be downloaded onto your device from the app store. For full iOS functionality, you’ll need:

- An Apple iPad or iPhone.
- A “camera” adaptor from USB to lightning (iRig 3 octave keyboard already comes with lightning cable). Lightning is the input that all newer models have for charging. These adaptors can usually be found at Apple stores.
- Headphones or computer speakers. *NOTE: In the newer iPhones without headphone jacks, we haven’t yet found a solution for sending audio to headphones or speakers, although you will be able to listen through the device itself. New iPads still have headphone jacks.*

Note that because of the small screen size, some of the more complex preparation windows (Tuning, for instance) may be difficult to negotiate on an iPhone. We have tried to make it as accessible as possible, and will continue to work on improving the UI for small devices!

## VST/AU Support

bitKlavier can be used as a VST/AU in your DAW of choice. The implementation is not fully fleshed out – parameter automation is in the works – but bitKlavier can be used as a plugin

instrument and mixed like any other plugin instrument. If you encounter any crashes or bugs while using bitKlavier with your DAW, please contact us or open a new issue on our GitHub page.

## Loading Samples

bitKlavier binaries 2.2 and newer include provisional support for SoundFonts (.sfz or .sf2), a file format used for batch-loading samples in a digital instrument.

In the desktop versions of bitKlavier (Mac or Windows, v2.2 and later), SoundFonts (.sfz or .sf2) that you place in /Applications/bitKlavier/soundfonts will be accessible in the sample loading menu beneath the default piano sample sets.

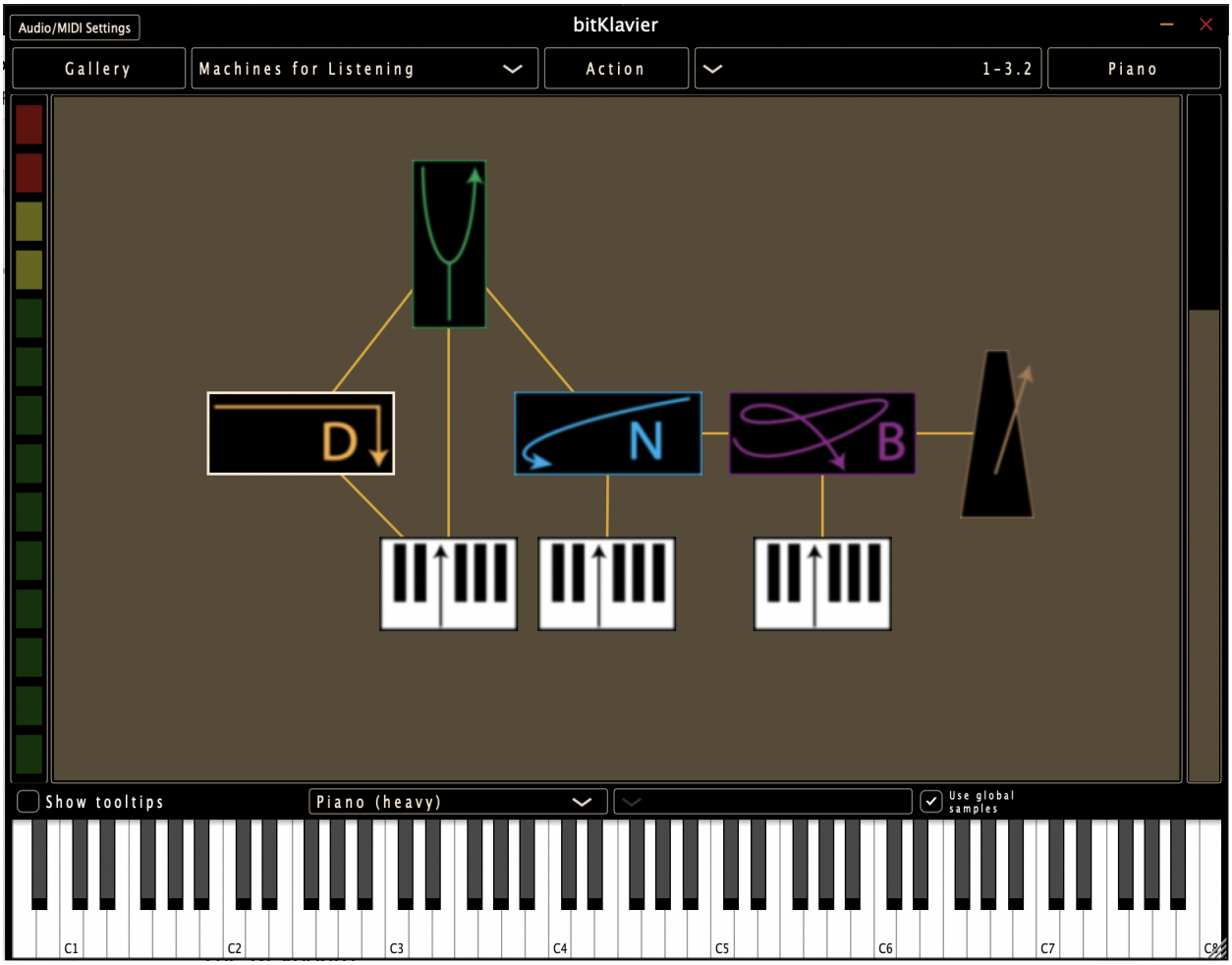
The iOS version of bitKlavier (v2.3 and later) is packed with a few default SoundFonts. If you would like to load more SoundFonts, you can add them through iOS File Sharing, accessible through iTunes when your mobile device is connected to your computer via USB.

Note: In the case of .sfz SoundFonts (in both the desktop and iOS versions), be sure to include the .sfz file in its parent directory along with any sample subdirectories.

As of v2.9, bitKlavier can also load any folder of properly named samples; see the Preferences section for details.

# bitKlavier Home Screen

bitKlavier uses a graphic interface similar to Max/MSP, Pure Data, and some digital synths such as Reaktor. Free-standing modules (the “preparations”) are added to an environment and connected to one another by drawing lines indicating signal flow through a given Piano.

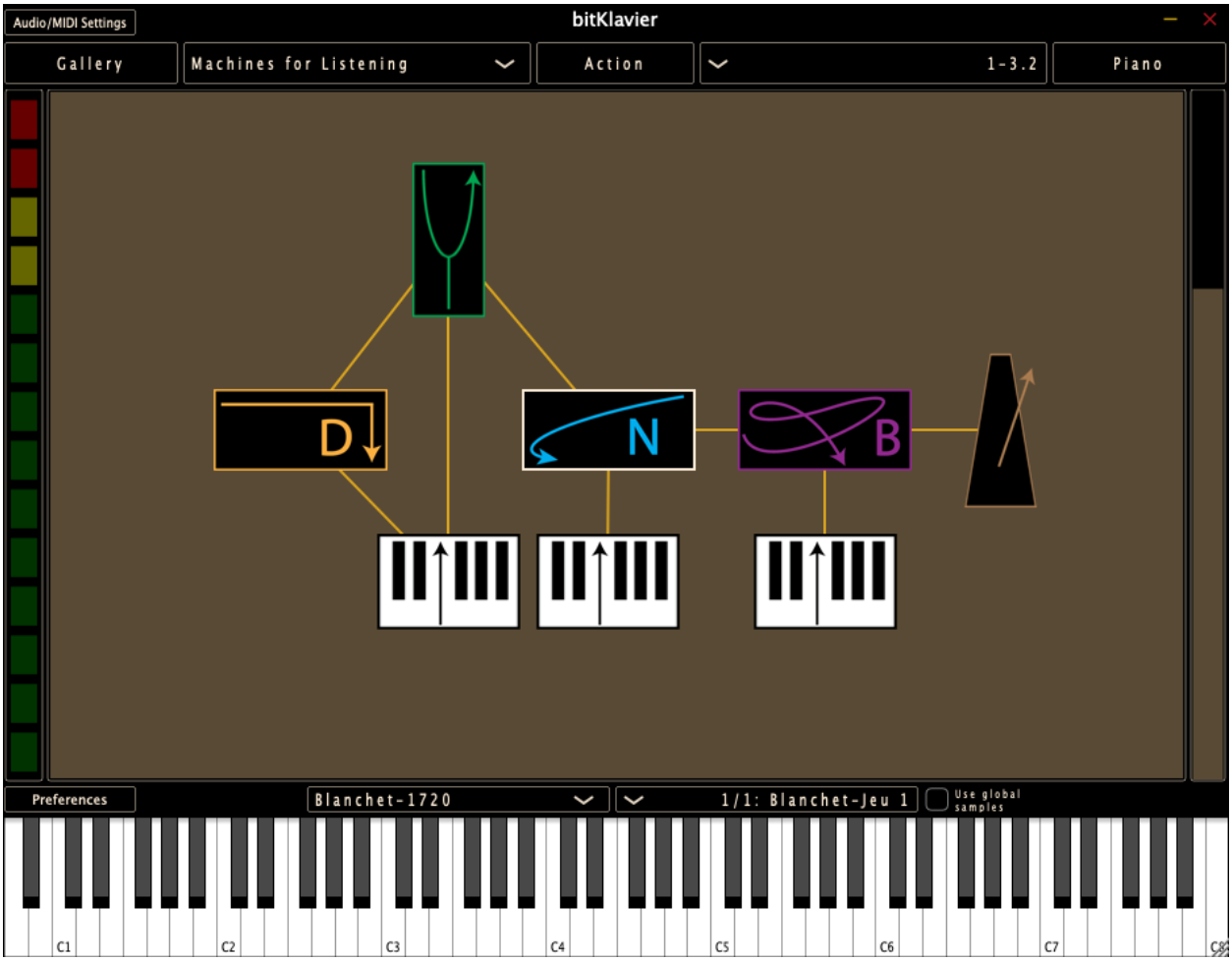



## bitKlavier Construction Site

The brown field in the center of the bitKlavier home screen is called the Construction Site. Preparations can be added to the Construction Site by using key commands or the Action tab. Double-click any preparation to open its corresponding settings window. Preparations can be linked (command-click and drag) and moved (click and drag). The location of a preparation has no influence on its behavior; it simply serves as an organizational way to understand the logic behind your bitKlavier Piano.

## Samples and SoundFonts

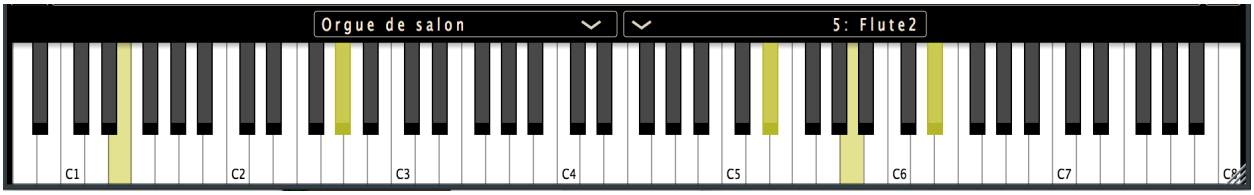
By default, bitKlavier loads a rich sample set of a grand piano, “Piano (heavy).” If memory is an issue, load the “meidum,” “light,” or “lightest” set. When a particular preparation is selected (as in Direct in the image above), the “use global samples” toggle appears, which indicates that this preparation should use the default sample set (“Piano (heavy”) in the above image). However, as of v2.6, you can also assign different sample sets to individual preparations:




Here, “use global samples” is unchecked, so the highlighted Nostalgic preparation will use the “Blanchet-1720” sample set, while Direct is still using the “Piano (heavy)” set (clicking on Direct will reveal that “Use global samples” is toggled on. These settings are saved with the gallery. Note that some SoundFonts have multiple embedded libraries, which can be chosen by the menu to the right (set to “1/1: Blanchet-Jeu 1” in the above image).

## The Keyboard

Along the bottom of the bitKlavier window is an 88-key keyboard. These keys can be used as a rudimentary MIDI keyboard. When a key is clicked, bitKlavier will behave as if it is receiving an external MIDI signal. This can be useful for exploring bitKlavier without a MIDI keyboard. When a MIDI keyboard is plugged in to your computer, the keyboard along the bottom will show which keys are being played.




## Gain

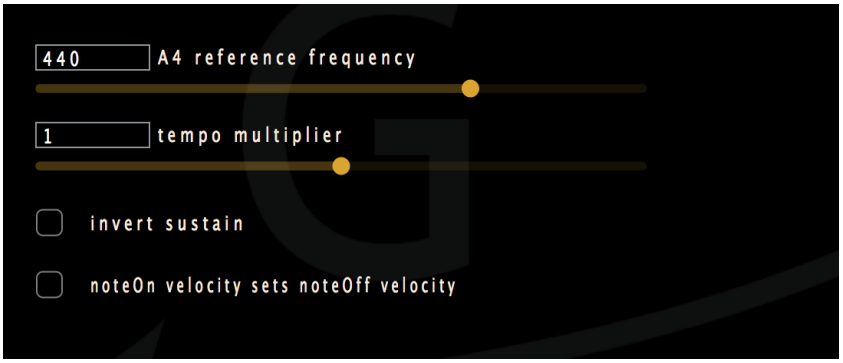
A colored VU meter on the left-hand side of bitKlavier indicates the audio output level. Along the right-hand side, a discrete vertical slider controls overall gain from +12.0 to -90.0 dB.

## Gallery Tab

The Gallery tab in the upper left provides global control for bitKlavier, plus controls for creating, renaming, and deleting Galleries (collections of preset Pianos).

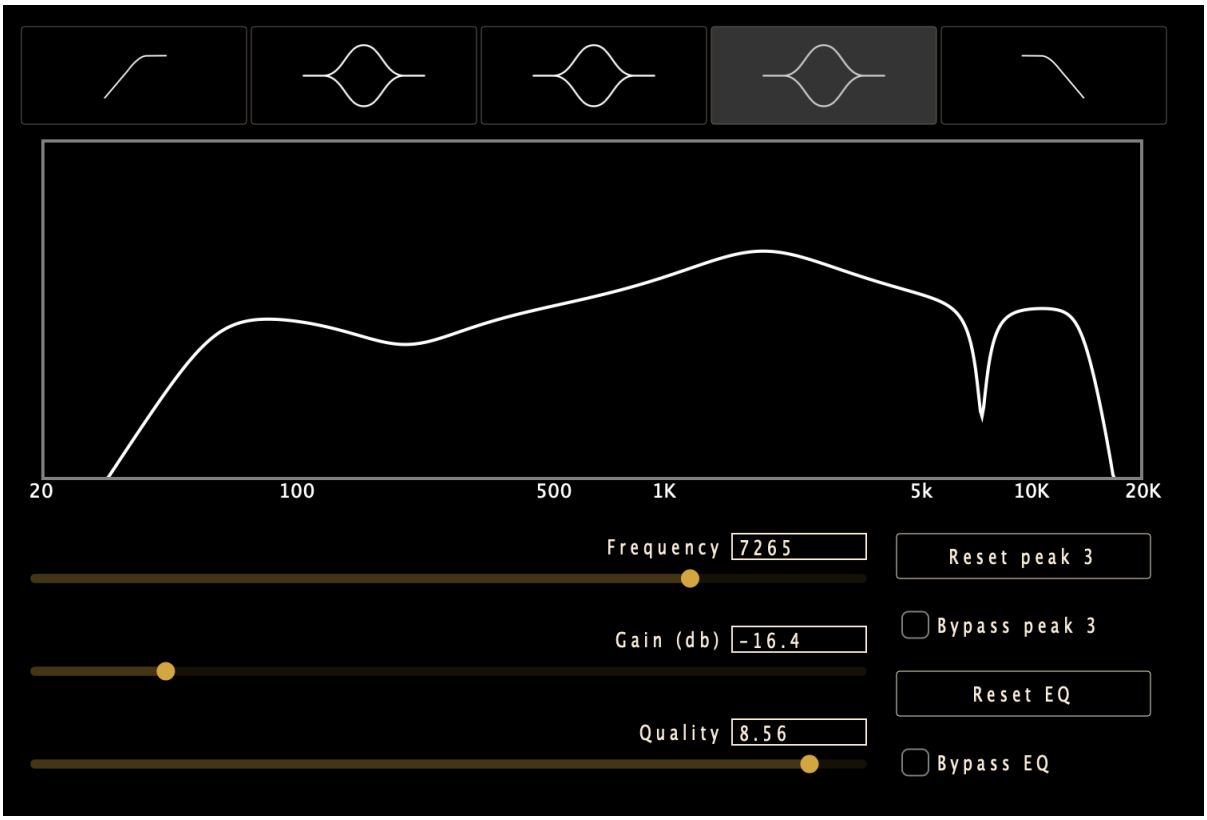

**New** - create a new empty Gallery; a dialogue allows you to name your new Gallery
**Save As** - save a copy of the current Gallery, move it to a new location on your HD
**Open** - load Gallery settings from a location on your HD
**Open (Legacy)** - load .json file from the original bitKlavier (made in Max/ChucK) and create a new Gallery from it.
**Clean** - remove preparation presets not current used by any Pianos, to avoid clutter of accumulated presets when working in a Gallery. Every time you add a preparation, bitKlavier creates a UI object and a corresponding preparation under the bitKlavier hood. When you create a preparation then delete it from the Construction Site, the preparation's settings can still be found in the dropdown preset menu of any corresponding UI preparation. "Clean" will remove any preparation that is not currently in use by a visible preparation.
**Share** - send bitKlavier settings via Email, Messenger, or Facebook, with an attached Gallery file in .xml format (this will only work with custom Galleries).
**Settings** -

- A4 reference frequency (by default 440hz)
- Tempo multiplier (by default 1.0) which adjusts the overall tempo for a particular performance or rehearsal by adjusting all tempi for the selected Gallery's Tempo preparations
- Invert Sustain - switches "pedal on" and "pedal off" signals; some pedals operate differently than others
- noteOn velocity sets noteOff velocity - many MIDI keyboards don't send discrete noteOff signals, which bitKlavier uses to set volumes for hammer and release samples, and several other settings throughout the preparations. This mode can be used to have bitKlavier set the MIDI noteOff velocity to the noteOn velocity.
- Equalizer - select either of the arrows on the sides of the current pane to display the Gallery equalizer. The equalizer and its functions are explained in further detail in the next section.



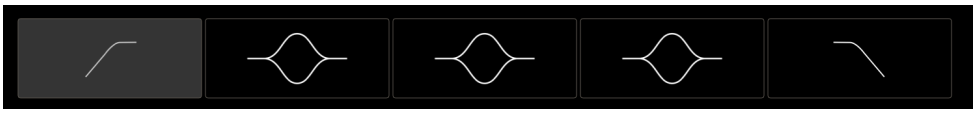

## Equalizer

New in v3.0, a parametric equalizer allows users to fine tune the sound spectrum of their custom Galleries by emphasizing or de-emphasizing specific bands of frequencies. Users might choose to alter the strength of certain bands of frequencies in order to better suit their room, speaker system, or audio equipment, or to re-envision entirely the sound of a custom Gallery.

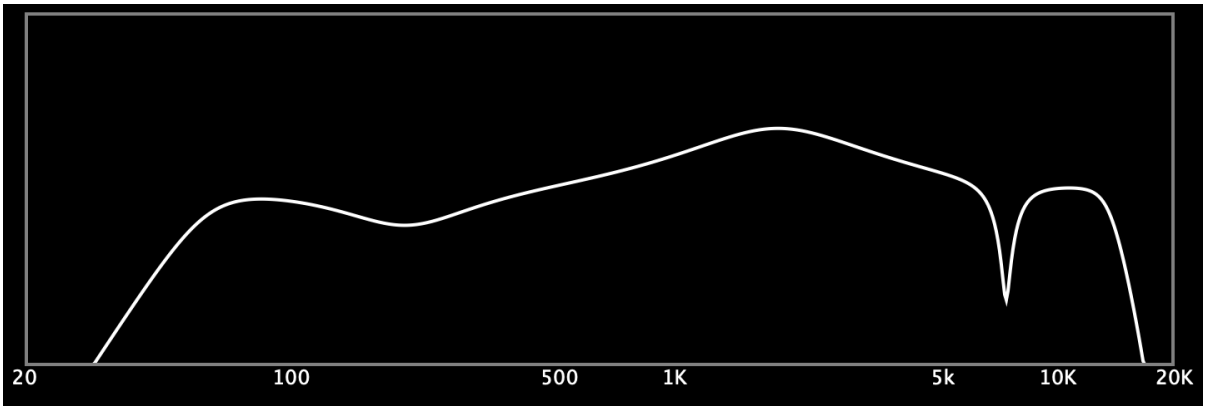



The equalizer can be shown by clicking the equalizer button in the lower right-hand corner, just above the main keyboard (or by navigating to Gallery settings under the Gallery tab and then scrolling one tab left or right using the arrows on either side of the pane). The equalizer affects the sound of the entire Gallery and parameter selections will be saved when a user chooses to “save” their custom Gallery.

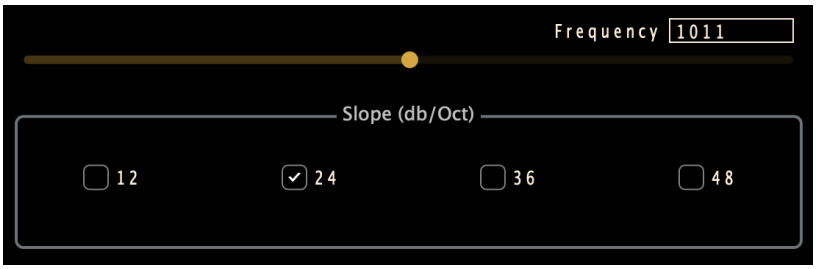
**Filter Selection Buttons** - choose between a low cut (high pass) filter, one of three peak filters, or a high cut (low pass) filter, respectively. The grey button indicates the currently selected filter.



**Graphic Display** - a visualization of how the equalizer’s parameters affect the frequency spectrum, from 20Hz to 20,000Hz. When the equalizer is bypassed, the display will appear greyed out.



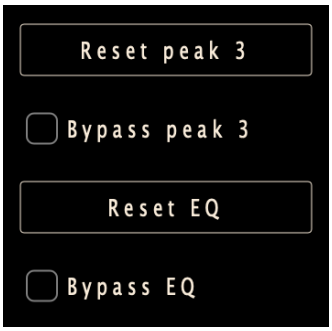

**Cut Filter Parameters** - both of the cut filters use the same parameters: frequency and slope. To alter these parameters, select one of the cut filters using the leftmost or rightmost filter selection button at the top of the equalizer. The frequency parameter controls the highest frequency (or in the case of the high cut filter, the lowest frequency) that the filter will not attenuate. The slope parameter affects how drastically the filter attenuates frequencies just below (or in the case of the high cut filter, just above) the cutoff frequency. A higher slope corresponds to more drastic attenuation just beyond the cutoff frequency. Try moving the frequency slider and selecting different slope options while glancing at the graphic display to get a feel for how these parameters affect the frequency spectrum.




**Peak Filter Parameters** - the three peak filters have their own parameters which behave differently than the cut filters. To alter these parameters, first select one of the peak filters using the three middle peak filter selection buttons. Each of the peak filters works by attenuating or boosting frequencies in the shape of a peak (or valley)—frequencies nearest the center are most strongly affected and frequencies far from the center are less strongly affected. The frequency parameter controls the frequency around which the peak centers. The gain controls how much the peak rises or sinks. 0db corresponds to no change, while a positive db indicates a boost (with higher numbers being a more significant boost) and a negative db indicates a cut (with lower numbers being a more significant attenuation). Quality relates to the bandwidth of the filter—the slope of the peak. A higher quality indicates a steeper slope, while a lower quality indicates a gentler slope. Adjust these sliders while

glancing at the graphic display to get a feel for how changing these parameters affects the frequency spectrum.


**Reset and Bypass** - the currently selected filter can be reset to its default values. It can also be temporarily bypassed to hear the effects of the filter compared with the unfiltered sound. Likewise, the entire equalizer can be reset to its default values. It can also be bypassed to temporarily audition the unaffected sound of the current Gallery. When this option is selected, the graphic display will appear greyed out.




## Gallery Menu

A Gallery is a group of preset Pianos. To the right of the Gallery tab there is a list of pre-programmed Galleries. If you create custom Galleries, they will be saved here. By default, bitKlavier comes with a number of Gallery presets:

**Basic Piano** - a simple starter Piano with a Keymap, Direct, and Tuning preparation which together serve as an equal-tempered 88-key Piano.

**Mike's Gallery** - a Piano with seven crazy presets designed by bitKlavier co-creator and programmer Mike Mulshine. Follow the prompts in the Piano titles for a musical adventure.

**Examples** - 27 preset example Galleries, most of which include detailed commentary about what the Piano does and how it works. These are categorized by preparation type, and are an excellent hands-on introduction to how bitKlavier works.

**Nostalgic Synchronic** - 8 Galleries for etudes written by bitKlavier creator, performer, and composer Dan Trueman for his work *Nostalgic Synchronic*. Visit the *Nostalgic Synchronic* website for more information. Sheet music included in the “music” folder.

**Mikroetudes** - inspired by Béla Bartók’s *Mikrokosmos*, the *Mikroetudes* are Galleries built to be used in conjunction with short pieces written specifically for bitKlavier, with the intent of introducing users to bitKlavier’s features and preparations. Visit the *Mikroetudes* website for more information. Sheet music included in the “music” folder.

**Machines for Listening** - These nine open-form sketches are intended as active listening guides for bitKlavier, a kind of digital musical machine configured in specific ways to process the operator's input and generate sound. Each "listening machine" has specific settings and interconnections that yield sometimes unexpected rhythms and textures, but are in fact completely deterministic — anything that seems like randomness is a product of the specific interactions between operator and machine. Sheet music included in the “music” folder.

## Action

The middle “Action” button in the bitKlavier menu bar allows users to add new preparations. Click “Action>Add…” and choose from one of the many preparations. Each preparation can also be created with a corresponding key command, listed in the Action menu for reference. The preparation will be generated at an arbitrary location and can be dragged, moved, and connected within in the Construction Site.

Also under the Action menu is the “All Off” button, which stops all sound from bitKlavier, like a MIDI panic function. This can be useful if a Synchronic or Nostalgic preparation unintentionally begins cascading indefinitely and the source can’t immediately be pinpointed.

When preparations are selected, however, this menu has different options, including copy/cut/paste, and options for connecting, disconnecting, and visually aligning the selected objects. Note that you can copy/paste into a new piano; this is different than creating a Linked Copy (see below), which creates a new Piano with shared (linked) preparations (changing the preparation in one Piano will also change it in the other), whereas copy/paste creates copies of the preparations that can be edited independently from the originals.

## Pianomap Dropdown

A number of different Pianos can be created within a given Gallery: These Pianos can be saved and located in a drop-down menu in the upper right of the bitKlavier Home Screen. For example, “Mike’s Gallery” is a single Gallery, but includes seven different Pianos with slightly different preparations and settings. The drop-down Pianomap menu allows you to navigate through different Pianos in a given Gallery. Using a Pianomap preparation, you can

use Keymap signals to move between Pianos on the fly. More information on how this works can be found in the Preparations section of the manual, or number 27 in the Examples Gallery.

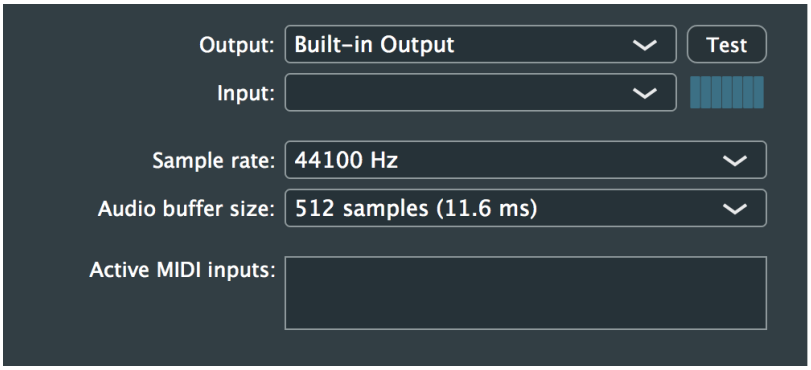
## Piano Menu

To the far right, a drop-down menu allows users to create, rename, or remove Pianos from a given Gallery. It's also possible to create a "linked copy" of the piano; this will create a new piano that can be modified, but the preparations between the copy and the original will be shared, so changing one will change the other. This is different than copy/paste from the Action menu above, which creates copies of all the preparations that can be edited independently. Finally, individual Pianos can be exported and imported here as well; these are saved in the "pianos" folder in the home bitKlavier folder.

## Audio/MIDI Settings

In the far upper-left corner, a small "Audio/MIDI Settings" button leads to audio settings as well as saving or loading a bitKlavier state. In the Audio/MIDI settings, users can choose and test their audio output source, select a sample rate, (44.1Khz by default), buffer size (512 samples by default), and select from available active MIDI inputs. If your MIDI keyboard is not working properly, this menu is the first place to check.

Regarding the buffer size, different sizes will impact the feel of bitKlavier. A smaller buffer will result in a quicker response, but will create more of a CPU burden which could impact performance. Experiment with different values and find a setting that works for your computer and playing style. (Loading a lower-resolution sample set will *not* lighten the CPU burden, but will allow bitKlavier to open and load more quickly.) *We find that setting the buffer size to **128 samples** works well.*

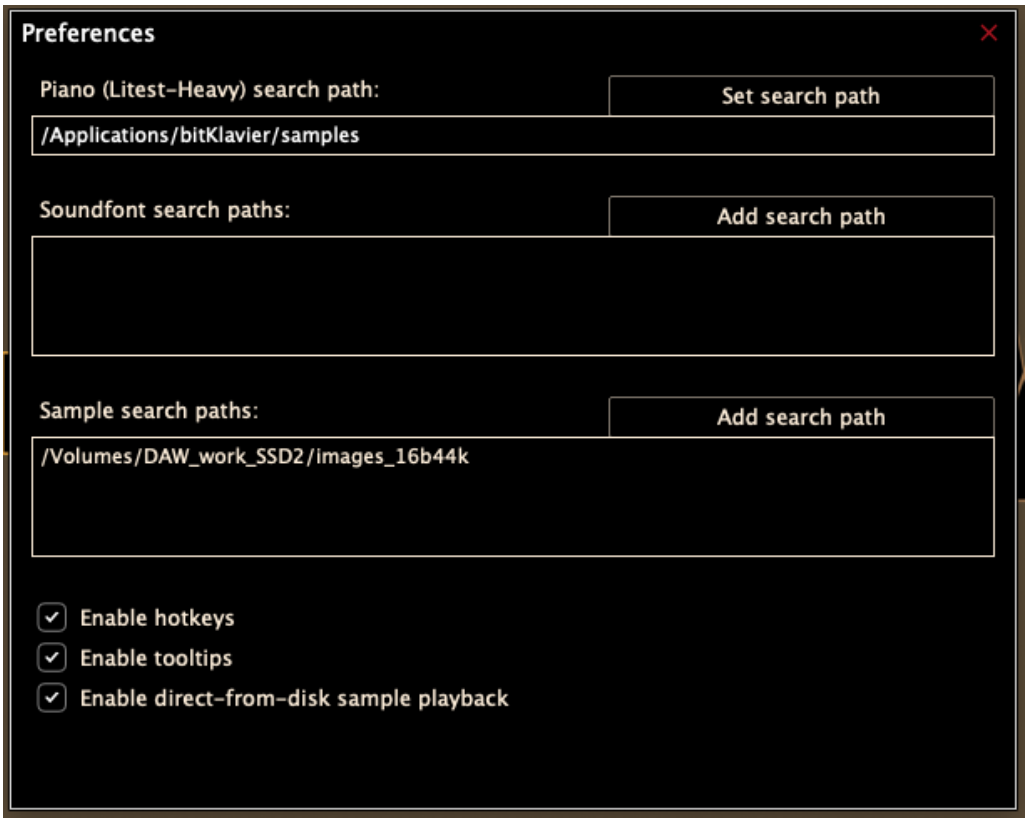



Under "options," users can save the current state of bitKlavier. This could potentially be useful when using bitKlavier as a plugin in a DAW. "Save state" saves a file of bitKlavier's current audio/MIDI settings. "Reset to default state" tells bitKlavier to behave as if you just

opened it for the first time, loading the Basic Piano and initial sample set (your preset Galleries and Pianos will not be deleted).

## Preferences

As of v2.9, at the bottom-left corner, the “Preferences” button enables selection of a number of preferences that are saved with the application (and apply to ALL Galleries):




**Piano (Litest-Heavy) search path -** If you prefer to keep the built-in “samples” folder somewhere other than the Applications/bitKlavier folder, you can put it wherever you like and then set the search path for it here so bitKlavier knows where to find it. This may be because of storage space issues, or because you want to use Direct-from-Disk playback and need to put these samples on a solid-state-drive (SSD).

**Soundfont search paths (for custom samples) -** Similarly, if you prefer to keep the built-in “soundfonts” folder somewhere other than the Applications/bitKlavier folder, you can put it wherever you like and then set the search path for it here so bitKlavier knows where to find it. This may be because of storage space issues, or because you want to use Direct-from-Disk playback and need to put these samples on a solid-state-drive (SSD).

**Sample search paths -** as of v2.9, bitKlavier can now import a set of samples directly, as long as they follow the file naming scheme indicated below. Set the search path(s) here so bitKlavier can find your samples. The naming scheme is as follows:

- Pitches should be named ***[Note][Octave]v[Velocity].wav***, e.g. “F#6v5.wav”
- Hammer release sounds should be named ***rel[key number].wav***, e.g. “rel76.wav”
- Resonance release samples should be named ***harm[Note][Octave]v[Velocity].wav***, e.g. “harmD#sv3.wav”

Note that bitKlavier *should* be able to handle any reasonable collection, and doesn’t need to have hammer or resonance samples, or any particular pitches (even one pitch will do!)

**Enable hotkeys -** disable this if you don’t want to use hotkeys (like ‘s’ to create a Synchronic object) in the construction site

**Enable tooltips -** disable this if you don’t want popup help text to show as you hover over particular sliders, menus, and so on

**Enable direct-from-disk sample playback -** when checked, bitKlavier will attempt to play the built-in “Litest-Heavy” and your custom samples (as found in the “sample search paths”) directly from disk, without loading them into RAM. This has obvious advantages for large sample libraries, but requires a very fast disk and connection to work well; internal and Thunderbolt SSD drives seem to work well, and USB3 SSD drives might also work well, but old fashioned HDD spindle drives will almost certainly not work well; if you plan to do a lot of this, you’ll want space on your internal SSD drive or invest in an external Thunderbolt SSD. *Note that direct-from-disk is NOT implemented for Soundfonts at this time, so those will always play loaded into RAM, regardless of how this preference is set*.

## Quitting bitKlavier (OSX)

On OSX, there are two ways to quit bitKlavier and they aren’t quite identical. Clicking the red X at the top right corner of the app will close bitKlavier AND save the state of it, so that when you reopen it, the same sample set and gallery should open that you had when you closed. Command-q will also quit the application but will NOT save the state of it. This may change in the future.

# bitKlavier for iOS

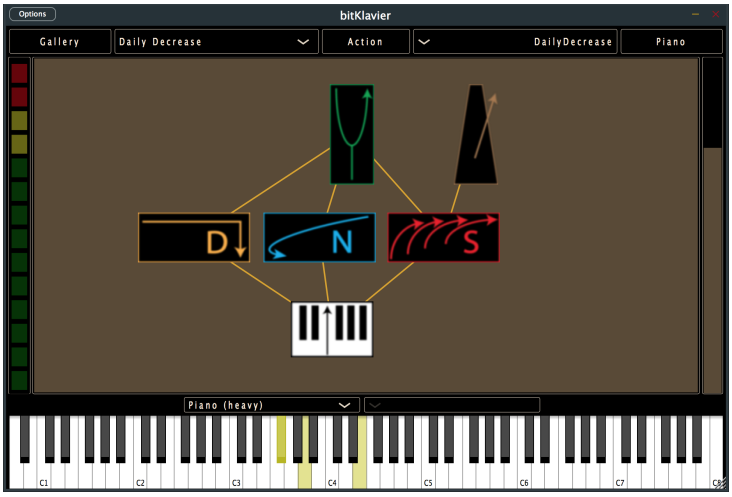
bitKlavier is available for iPhone and iPad and can be downloaded onto your device from the app store. bitKlavier for iOS uses the same windows, interfaces, and layout. One missing visual element is the mini keyboard, which has been hidden for economy of screen space. Tap the colored audio level meter on the left-hand side of the screen, and a mini keyboard will appear.

For full iOS functionality, you'll need:

- A "camera" adaptor from USB to lightning (iRig 3 octave keyboard already comes with a lightning cable). All newer iOS models use the lightning port for charging. These adaptors can be found through Apple or Amazon.
- Headphones or computer speakers. *NOTE: In newer iPhones without headphone jacks, using a Bluetooth MIDI keyboard or speakers is a possible workaround for the lightning port issue on newer iPhones. New iPads still have headphone jacks.*

# bitKlavier Workflow

The concept behind bitKlavier as a "prepared digital piano" is that we begin by making alterations to a Basic Piano. When using bitKlavier, we select keys and route them to different preparations so that samples will be processed in different ways. A bitKlavier Piano will have at least one Keymap preparation which selects a group of incoming key signals to be used by a linked preparation. That preparation may be: **Direct** (with minimal parameters such as gain, envelope, transposition), **Tuning** (traditional temperaments and adaptive, responsive systems), **Synchronic** (a customizable, adaptable sequenced pulse), or **Nostalgic** (reversed, sympathetic notes and clusters). Additional preparations control internal tempo parameters, such as **Tempo**, **Modification**, and **Reset**. If working in bitKlavier is entirely unfamiliar, check out our installation video and Basic Piano tutorial video, which provide step-by-step guides for getting familiar with bitKlavier.

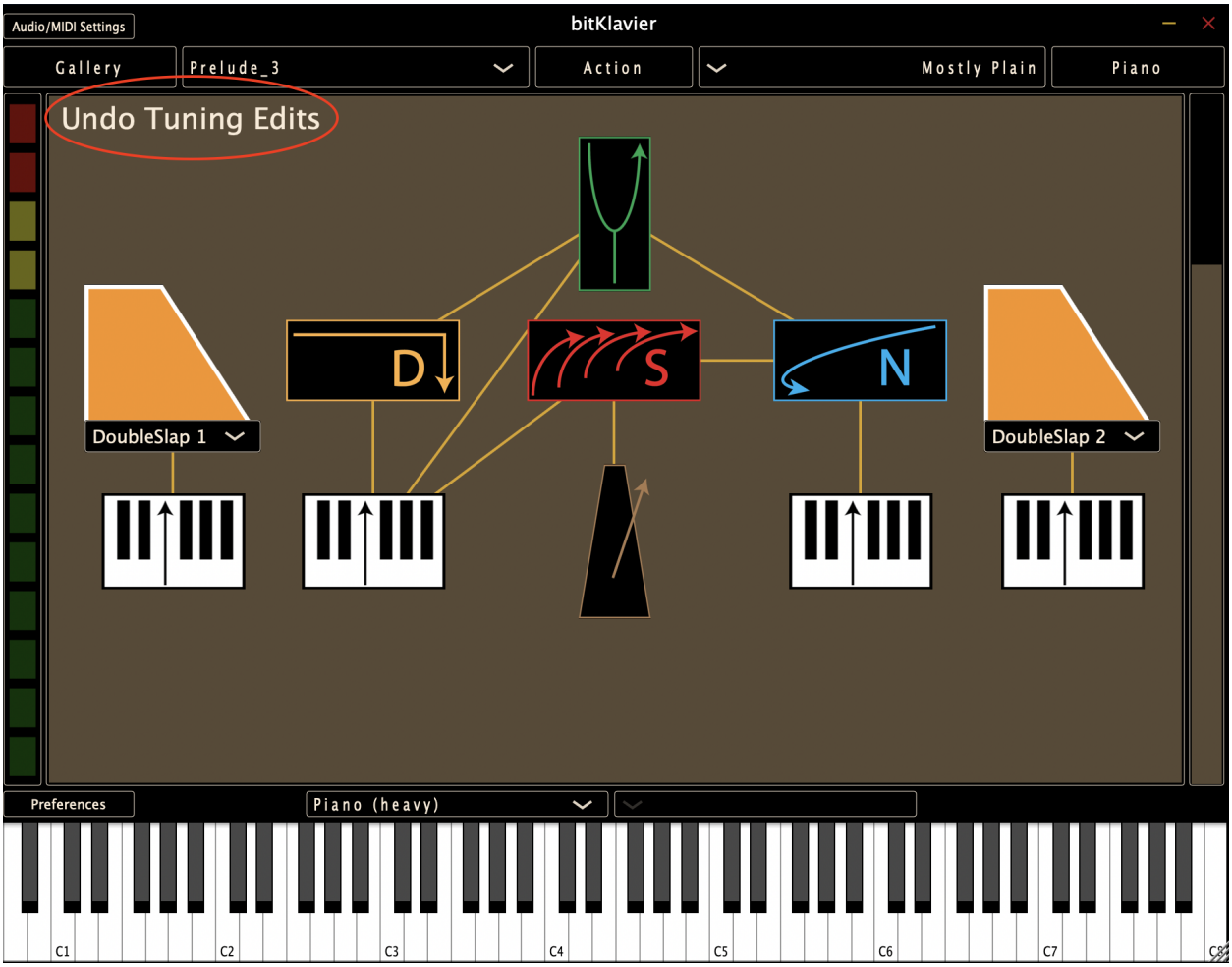



To begin experimenting, add a Keymap (k) plus any other preparation, and control-click-drag to connect them. Add another preparation, a third, a fourth. Double-click a preparation to tweak its settings. You can copy or delete preparations with standard copy/paste key commands (command-c and -v), or by clicking “Action” while a preparation is selected.

To create a Gallery–a group of preset Pianos–select “new” under the Gallery tab on the right-hand side. Any number of new Pianos can be saved and stored in this new Gallery by selecting “Piano>New” in the upper right-hand corner.

Every time a preparation is added, bitKlavier creates a UI object and a corresponding preparation “under the hood.” You may, for example, experiment with several Keymap preparations and save both of them with different names. Even if you only end up using one preset, the others will remain in that Gallery within a given preparation’s settings window. Double-clicking any preparation will allow you to load settings from any previously saved preparation in that Gallery. These settings can build up fast: Using the “clean” function (“Gallery>Clean”), any saved preparation *not currently in use by any Piano in the current Gallery* will be deleted. So before you “clean” a bitKlavier Gallery, be sure that any presets you’ve created in that Gallery are currently being used by a Piano somewhere in that Gallery.

As of v2.7, it is possible to undo most recent actions, with command-Z, or command-shift-Z for redo, or by selecting from the Action menu. This is available in the construction site, but not within individual preparation windows; if you want to undo a recent change to a preparation, close out the preparation window and then undo—your recent change in the preparation should be undone, and the temporary message will, as usual, tell you what was actually undone (in this case, the temporary message indicates that the most recent changes made in the Tuning preparation were undone:




# The Preparations

Envision the internal chain reaction of a traditional piano: Key, hammer, vibrating string, sound. The digital equivalent has the same conceptual components: keypress, MIDI signal, triggered sample, vibrating speaker. In bitKlavier, the preparations play back and manipulate pre-recorded samples, encompassing tuning, pitch, envelope, time, duration, and complex generative processes that trigger notes and rhythmic pulses.

There are four groups of preparations:

1) The **Core Preparations** (Direct, Nostalgic, Synchronic) can be directly connected to a Keymap preparation.

2) The **Control Preparations** (Keymap, Tuning, Tempo) can be used to control and automate the core preparations.

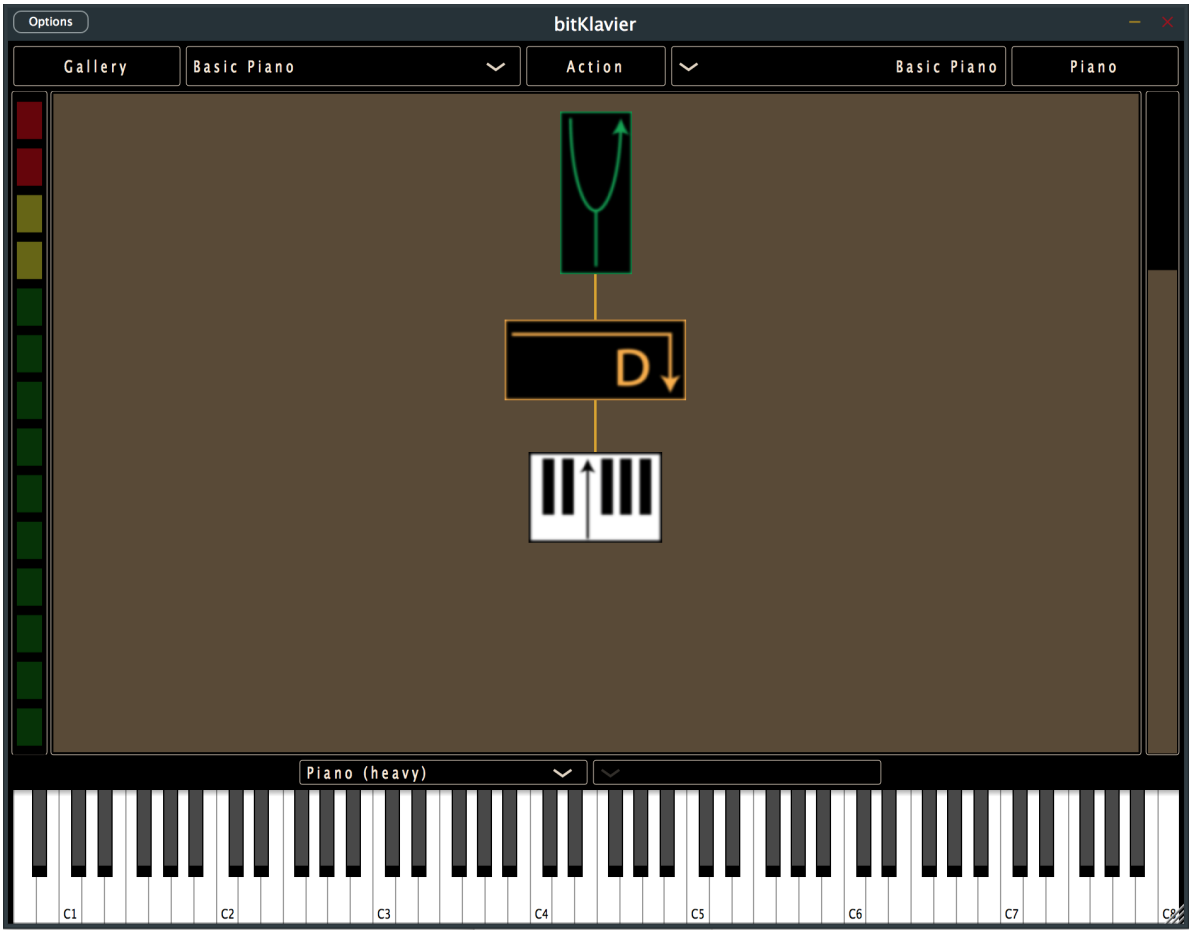
3) **Modification** and **Reset** allow you to change and preparation values on the fly, then return them to what they were, using Keymap input.

4) The **Piano Map** provides a convenient way to change between Pianos in the same Gallery.

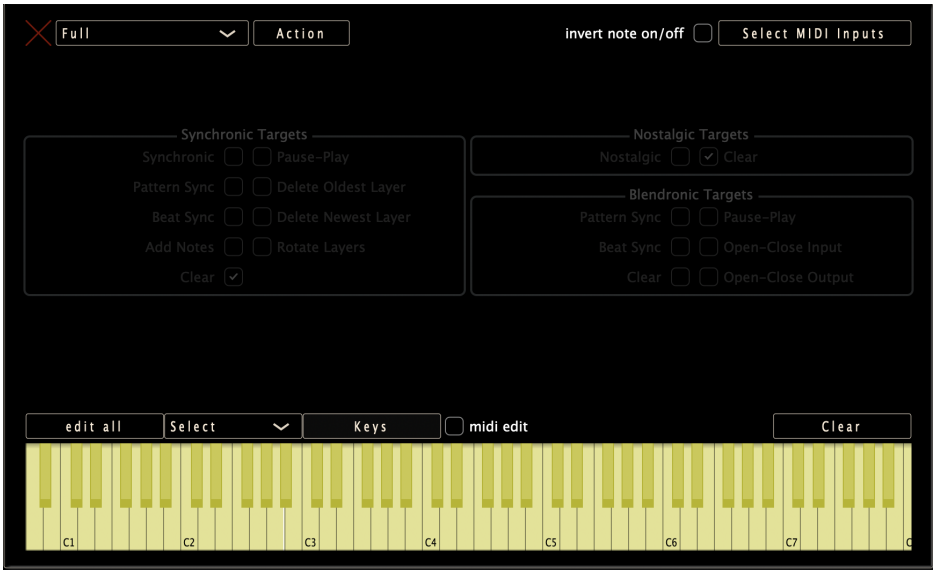
## Keymap

Keymap is one of bitKlavier's essential control preparations. It is the starting point for understanding how the bitKlavier environment works, and the basis for every bitKlavier Piano. At the basic level, using a Keymap allows us to select and route signals from our MIDI keyboard pass through to our various bitKlavier preparations and modifications.

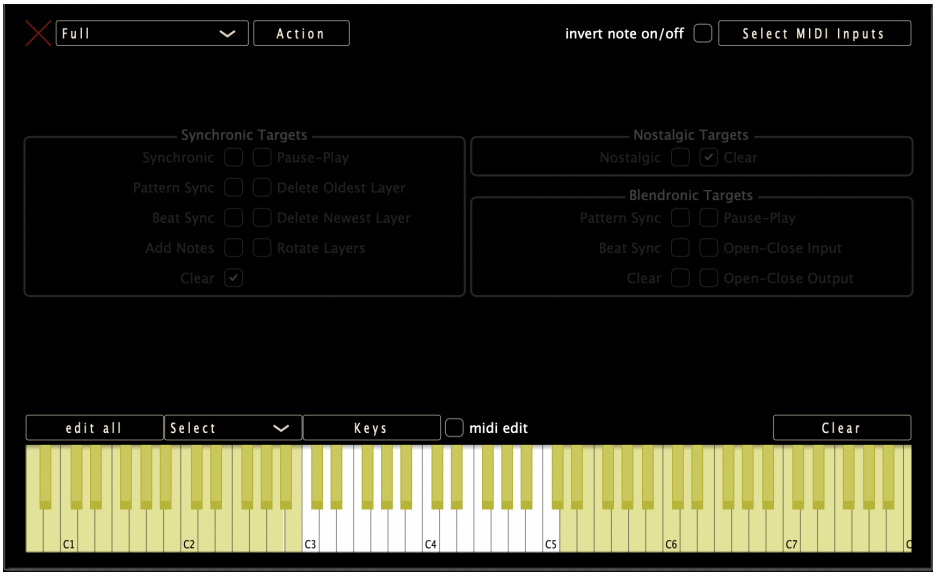
To begin experimenting with how Keymap functions, open the "Basic Piano" Gallery. Take a moment to note the signal flow of the Gallery: From MIDI input, the signal arrives at the Direct preparation via the Keymap, and the Tuning preparation provides temperament parameters.



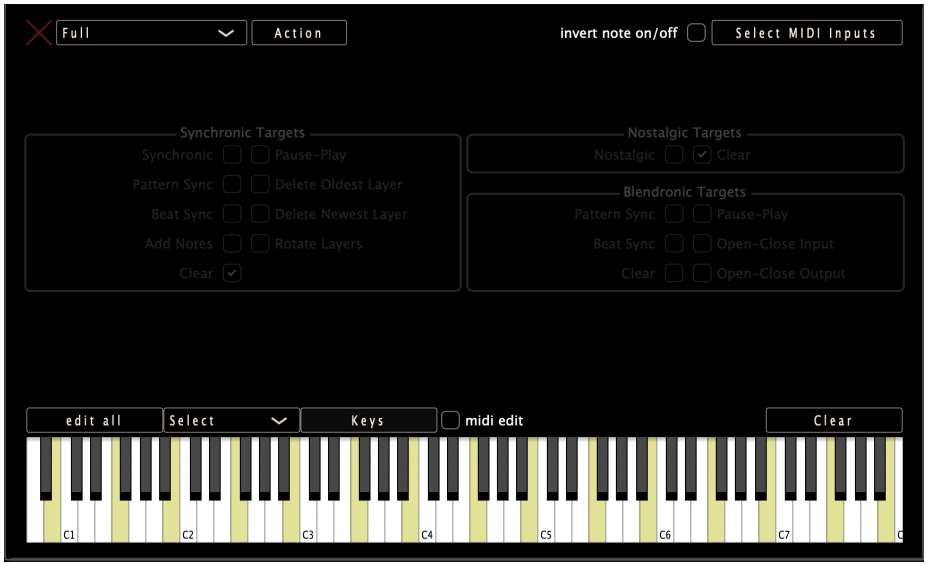

Double click the Keymap preparation: You will see a popup window with a keyboard along the bottom. All 88 keys are highlighted yellow: This means that all 88 notes are being passed through to the next preparation, which in this case is the Direct preparation.



To test how the Keymap works, click C4 (middle C). It will turn *white*, indicating that it has been *deselected*. Play C4 on your MIDI keyboard, and it will not sound, while all other notes remain active. Now, click C3, and drag across the white keys on the keyboard until C5. You have just deselected two octaves of white keys: Play your MIDI keyboard, and note that for those two octaves, *only* the black keys produce any sound.



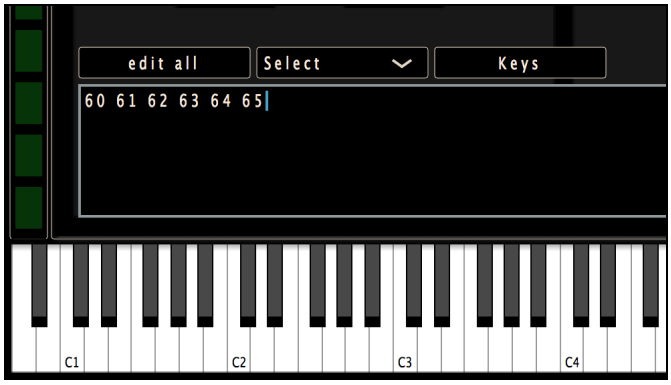

Keymap has several ways of selecting keys, since individually selecting and deselecting each desired key could quickly become tedious. Above the Keymap keyboard, hit “Clear” on the upper right to deselect all keys. Then, to the right of the “Select” button, choose “All… > B.” Every B on the keyboard will now be selected. Try it again with F.



There are many creative applications here, especially when using multiple Keymaps to process different parts of the keyboard. You could pair this F/B keyboard with another keyboard where every note is selected except for F and B: This means that we could prepare the F/B keyboard to make it behave in some unusual way. Imagine composing for a Piano where every F and B is actually pitched an octave higher, or those notes only play in reverse. Using Keymaps, we can easily route different MIDI notes to different destinations.

Deselecting keys is simple too. Choose “Select>All,” then change the “Select” dropdown to “Deselect” and choose “Octatonic 1.” The Octatonic 1 preset is a diminished scale beginning on A, so with those notes *deselected*, we end up with the notes outside of that scale, which are the notes of a Bº7 chord. Sweep your hands along the keyboard, or bring your hands down on random clusters: The only notes that sound are from Bº7 (B, D, F, Ab).

Another way to select and deselect keys is by using the “Edit All” field. This area uses MIDI values for each activated note. If you press “Clear,” then type 60 into the “Edit All” field, you will see C4 selected on the keyboard. MIDI values can be typed in or pasted here from other applications, as is the case with many parameters in bitKlavier.

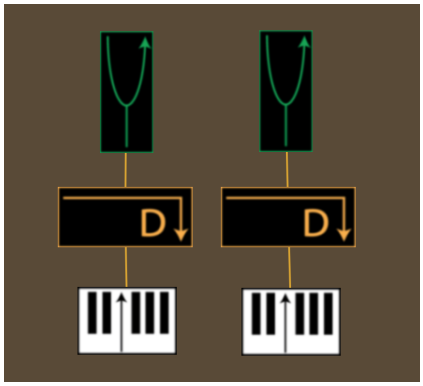



It is easy to save and recall presets in all bitKlavier preparations. In Keymap, select “new” and a Keymap will be auto-named and saved in the menu to the left. This Keymap can be modified, renamed, and deleted. These saved Keymaps are saved along with the Gallery, and can be used with other Pianos in that Gallery. Go to “Gallery > Clean” to delete all unused presets.


Individual Keymap settings can also be exported if you want to then later import them into other Galleries, or share with other bitKlavier users. These are saved as xml files in the “bitKlavier/preparations/Keymap” folder.

## Example: Using Two Keymaps

Let’s look at a straightforward example of how using Keymaps can open creative possibilities. Begin by creating a duplicate of the Basic Piano layout in the bitKlavier environment. To do this quickly, hit (k) for Keymap, (d) for Direct preparation, and (t) for Tuning preparation. Arrange the preparations by dragging them, then link them by click-dragging while holding the command key.

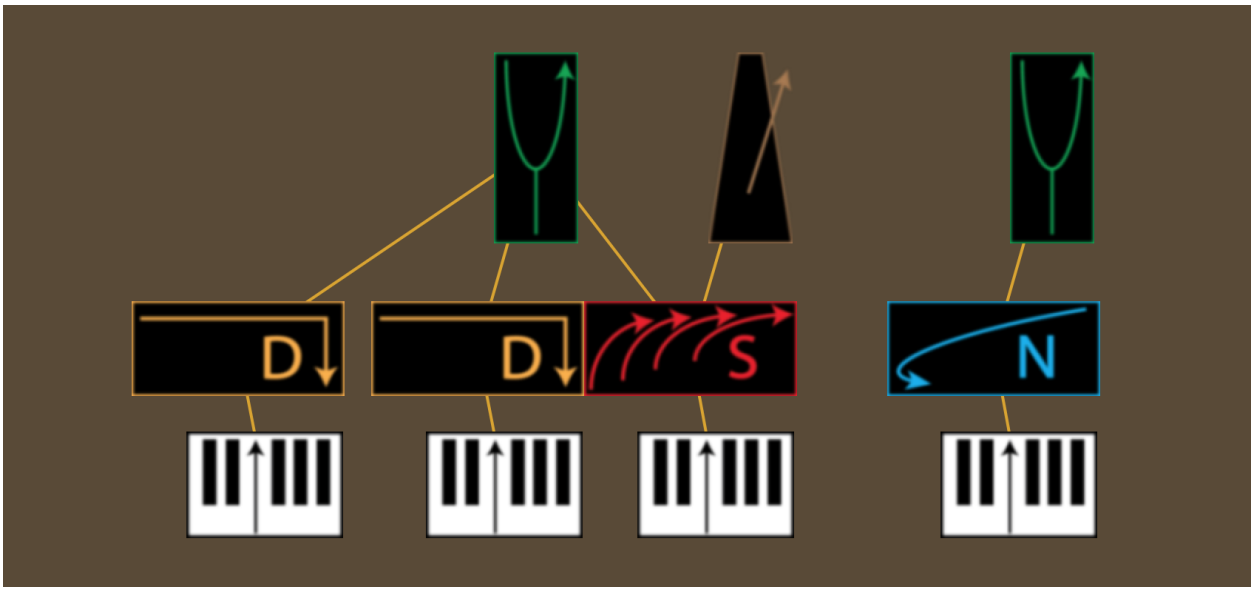



Double-click the Keymap preparation on the left-hand side of the screen, and select all notes *below* C4. Then double-click the Keymap preparation on the right-hand side of the screen, and select all notes *above* C4. We’ve now divided the keyboard in half. One way to test this is by opening one of the Direct preparations and dragging the “gain” slider to a different position. This should make one half of the keyboard louder or softer, depending on which side you chose.

Try experimenting with your split keyboard. Here are some ideas:

- Open the right-hand side Tuning preparation and change the tuning from Equal Tempered to Just. Play some hands-together scales and chords and listen to the way the two temperaments resonate with each other.

- Open the left-hand Direct preparation and adjust the Transposition slider to -0.5 (you can slide the bar or double-click and type the new value). The lower half of the keyboard will now be pitched one quarter-tone lower than the upper half.
- Add a Nostalgic preparation (n) and connect it to the Keymap on the left-hand side. The notes below C4 now replay in reverse with every MIDI-off signal. (Read more about the Nostalgic preparation later in the manual.)
- Add a Synchronic preparation (s) and connect it to the Keymap on the right-hand side. The notes above C4 now trigger a metronomic pulse. (Read more about the Synchronic preparation later in the manual.)

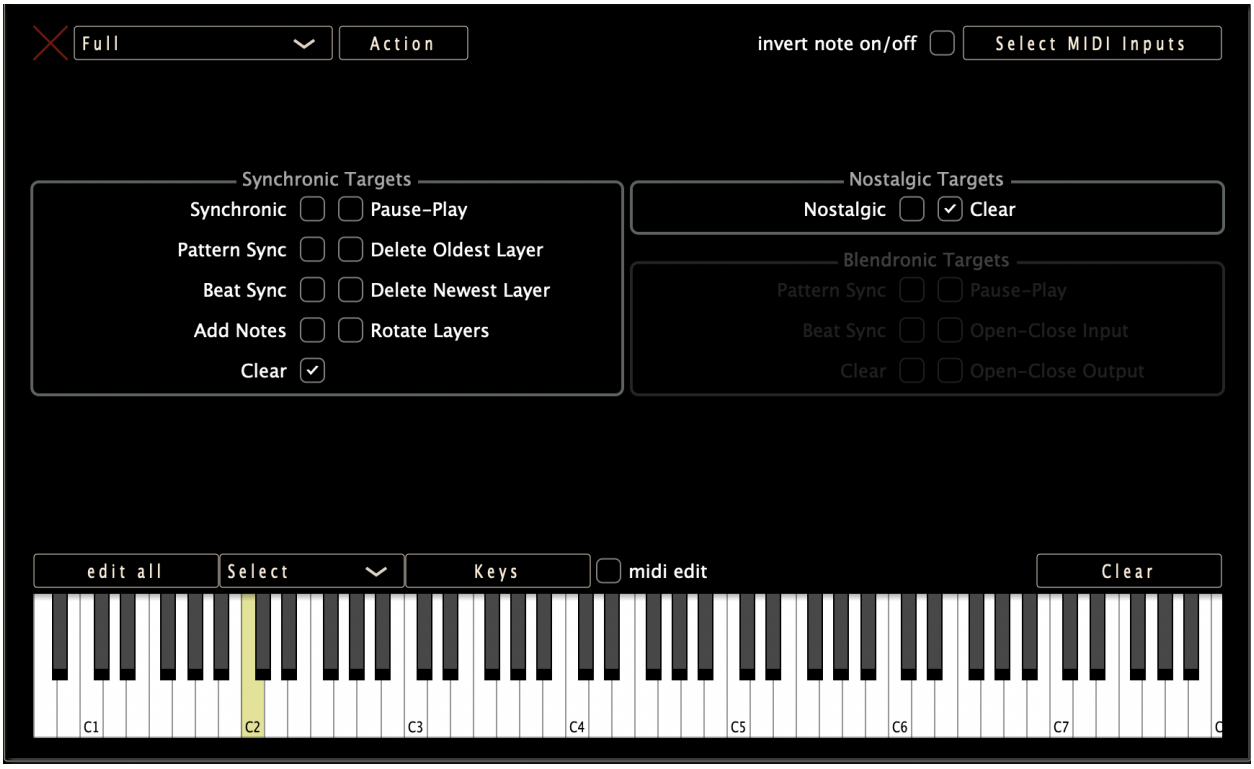



There are many preset Galleries that include multiple Keymaps, including “And So…” and “Cygnet” in the *Mikroetudes*, “Wallumrod” in Nostalgic Synchronic, and Example 16 “Tuning 1.”

## Event Targeting

New in v2.5, this feature enables a Keymap to target particular events in Synchronic, Blendrónic, and Nostalgic; for example, you can use a Keymap with one key selected to pause/restart playback in Synchronic, clear all sounding notes in Nostalgic, or restart the beat cycle in Blendrónic.

In the example below, C2 is prepared to clear all Synchronic and Nostalgic preparations to which this Keymap is connected; the Blendrónic targets are greyed out because no Blendrónic preparation is attached to this Keymap:

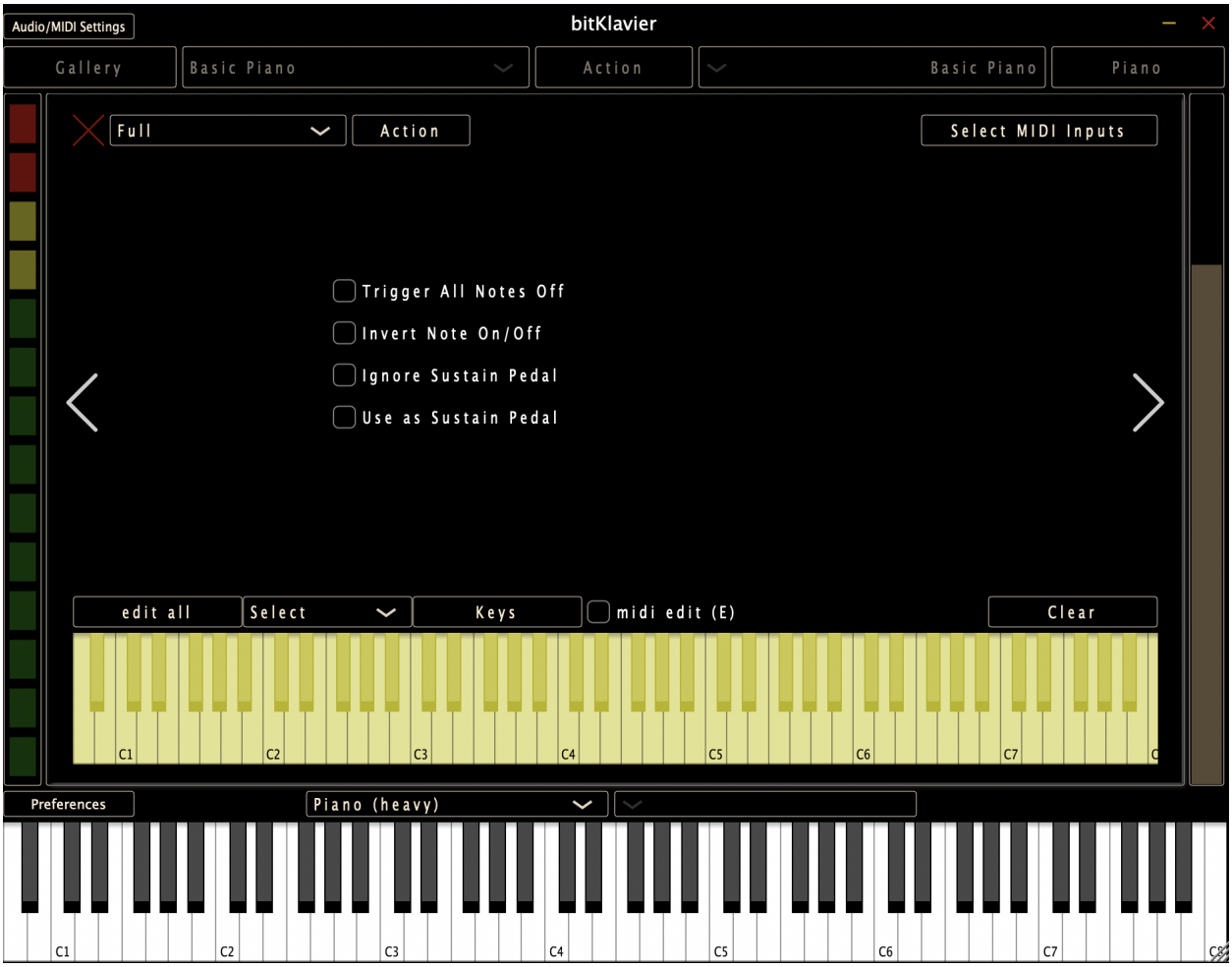



The specifics of each kind of target are detailed below in the sections for each preparation type. By default, when you connect a Synchronic or Nostalgic preparation to a Keymap, the “Synchronic” and “Nostalgic” targets will be activated, which will reproduce the default, pre-v2.5 behavior of those preparations. Switch them off if you want this Keymap to target subset behaviors like, for instance, Beat Sync in Synchronic, which will reset the pulse without starting the patterns (accents, transpositions, etc…) over.

## MIDI Devices

At the top right you can choose which of the connected MIDI devices you want to be activating this Keymap.

The following four options are available in their own tab within Keymap:




## Invert Note On/Off

Toggling on this option will reverse the expected behavior of the keyboard; when you press a key down, connected preparations will receive a noteOff message, and when you release a key, connected preparations will receive a noteOn message. Diabolical. Combine this with a negative value for Semitone Width in the Tuning preparation to turn your world completely upside down and inside out.

## Ignore Sustain

This option instructs the Keymap and any attached preparations to ignore the sustain pedal. One possible application: some keys ignoring the sustain pedal while others don't, using two different Keymaps.

## Use As Sustain Pedal

This can be useful in situations where you want particular keys to replicate the behavior of the sustain pedal; try it out but setting one key in a Keymap with this toggled on and using

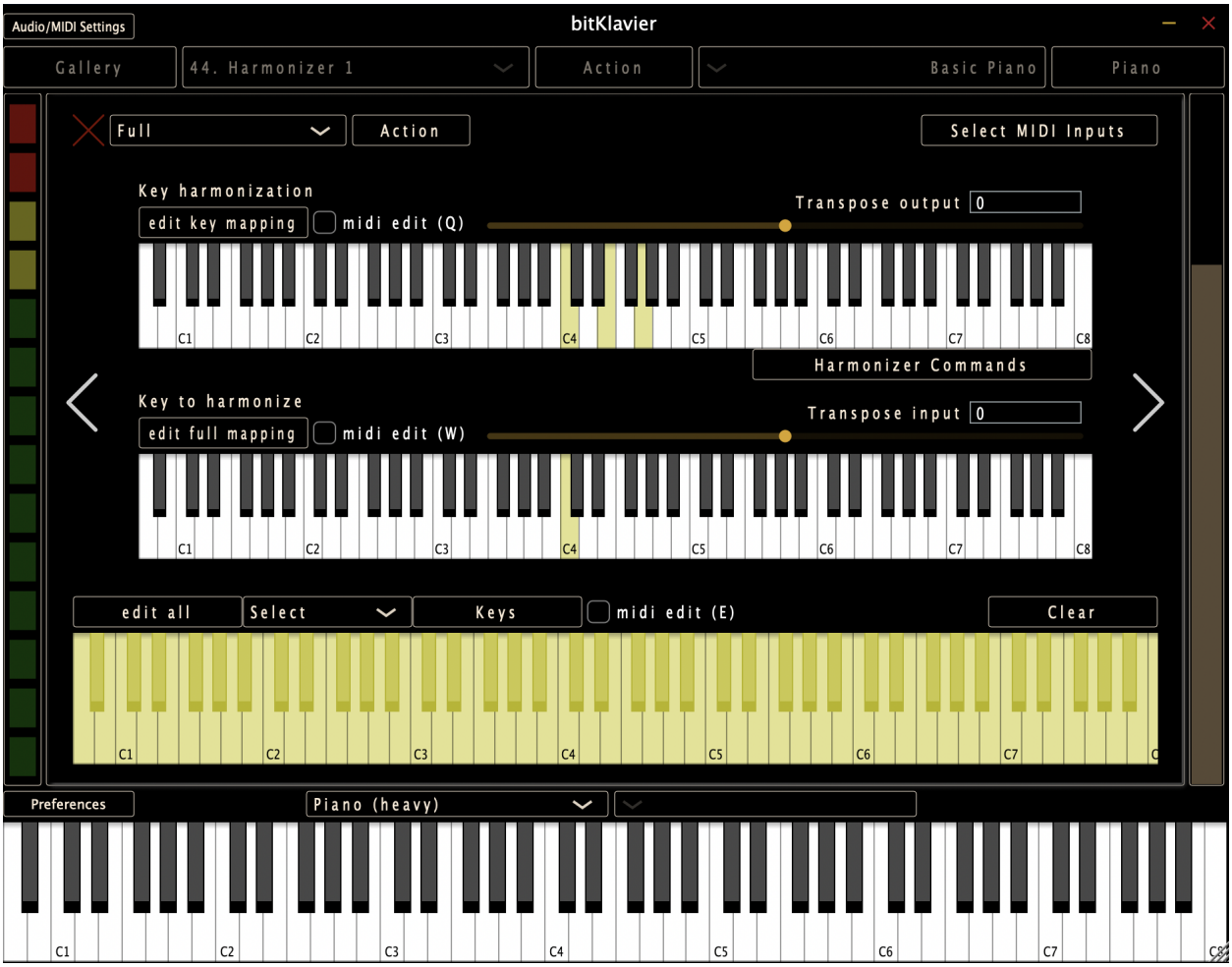
that key as a sustain “pedal.” Note that in this instance the Keymap does not need to be attached to anything, and its behavior will apply to bitKlavier as a whole (as an actual sustain pedal would).

## Trigger All Notes Off

Like “Use as Sustain Pedal,” this is a global behavior that applies to all of bitKlavier; when toggled on, and active keys in this Keymap will call the “all notes off” function, which should silence bitKlavier.

## Harmonizer

New in v2.7, this is a significant addition to Keymap’s functionality, allowing the creation of a range of “harmonizer”-like behaviors:




There are a lot of keyboards there!! The very bottom one is the “played” keyboard, which reflects whatever the attached keyboard is doing (and can be played directly on-screen as well, with the mouse). The second-from-bottom keyboard (in all yellow) is the standard

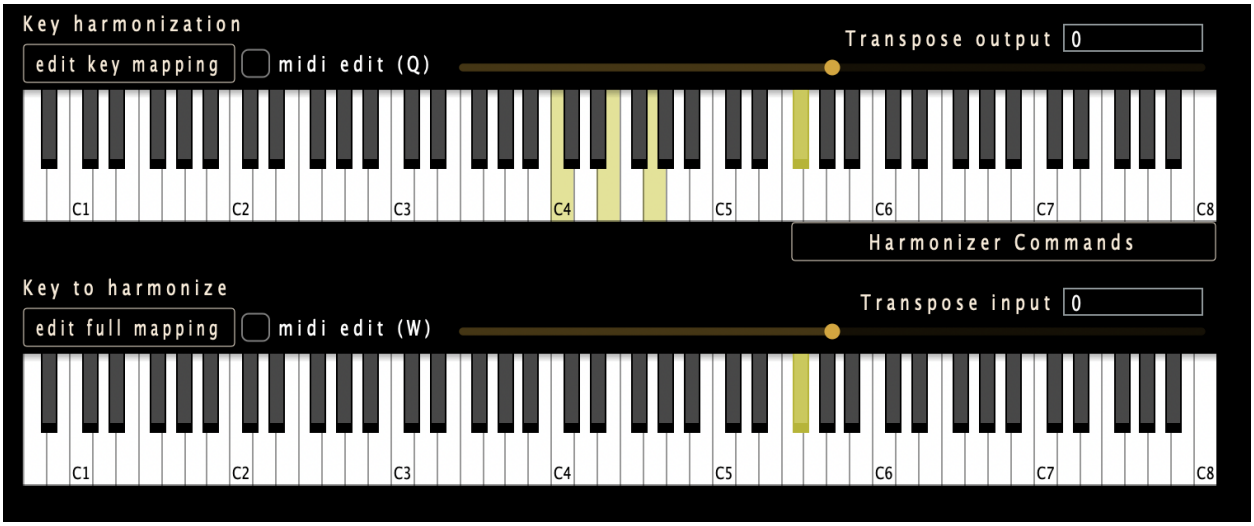
Keymap keyboard, which allows assignment of individual keys to be active or not (all of them are active in this example). The second-from-top keyboard allows you to select a particular key that you want to have harmonized in some way, and the top keyboard allows you to set what notes are played by that key. So, in this case we can see that playing C4 will result in a major triad. We can select other keys to see how they are harmonized, or press “edit full mapping” to see (and edit) all of the harmonizations as text

Using this functionality can be quite involved, requiring the assignment of individual harmonies to each and every key. In many cases, that sort of work is just the way it is, required as part of the composition process. However, there are a handful of macros that could be of use:

**“Clear”**
This will clear your harmonizer mapping *completely*. Note that this means your Keymap will actually not send anything through at all, as if none of the keys are selected.

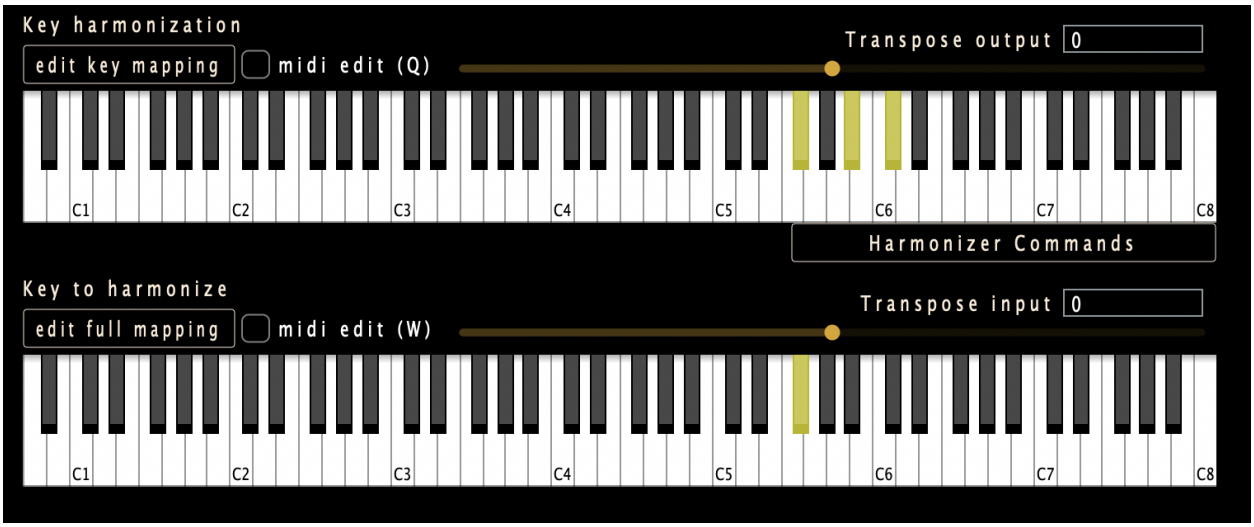
**“Default”**
This will *add* the default one-to-one harmonizer mapping to the top keyboard, so every key will play as expected. Note that this is added to what is already in your harmonizer; it won’t be cleared first unless you explicitly call “clear.”

**“Copy Mapping to All”**
This will take the mapping for the currently selected key in the “Key to harmonize” keyboard and copy it to all the other keys, *without transposing it*. So, in the example above, if you “copy mapping to all” for the C triad on C4, ALL notes will transpose to a C triad on C4; it’s a kind of “pitch trapping” mapping that can be quite interesting. So if I select, say, F#5 on the “Key to harmonize” keyboard, I’ll see this:



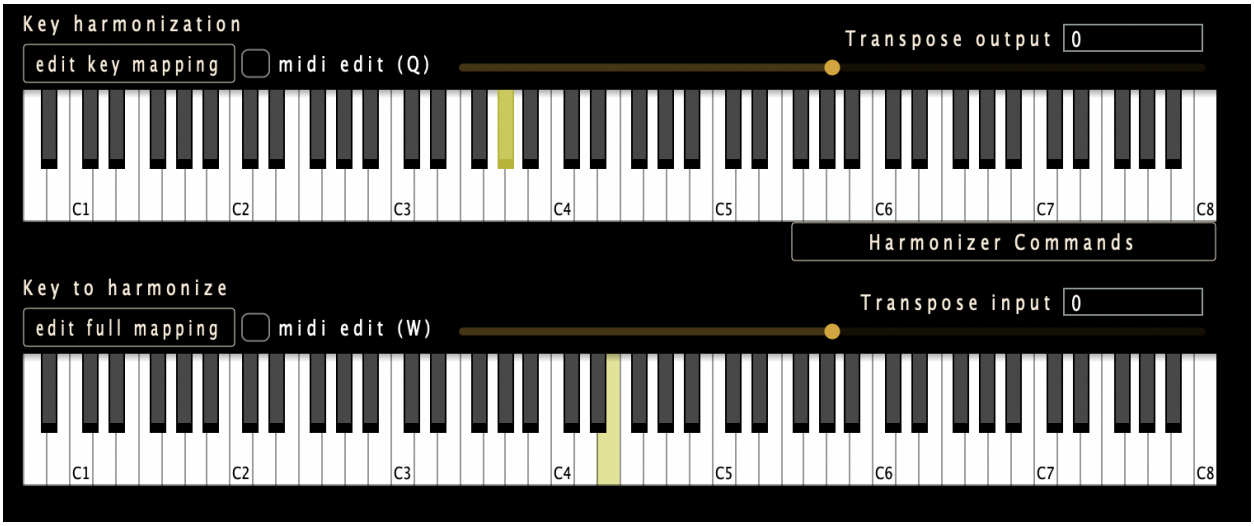
And, when I play and F#, I'll hear that F3 AND the C triad.

**"Copy Pattern to All"**
This will take the mapping for the currently selected key in the "Key to harmonize" keyboard and copy it to all the other keys, *transposing it*. So, if you want a major triad on all keys, you can call that from our original example, and every key will result in a major triad. Again, selecting F#5 will then show this:

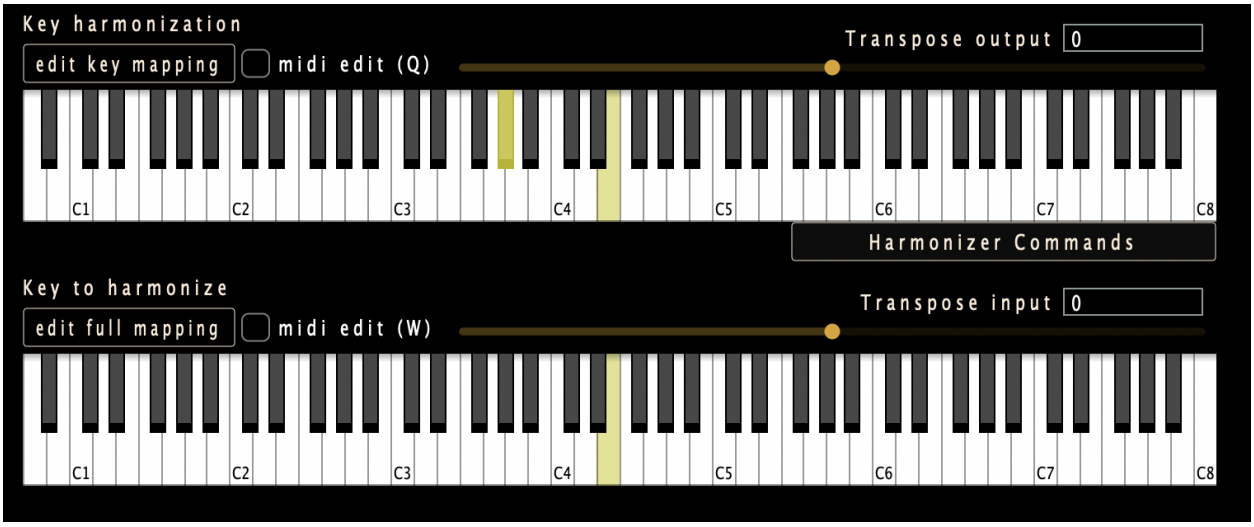



**"Copy Mapping to All" and "Copy Pattern to All"**
These are just like the above, except the mapping/pattern is copied only to all the octaves of the currently selected key.

**"Mirror"**
First, note that this macro will obliterate whatever mapping you have! Then it will map every note symmetrically located around whatever note is selected when you call it. So, for instance, if C4 is selected in "Key to harmonize" and you call "Mirror," C4 will map to C4, D4 will map to Bb3, E4 will map to Ab3, etc. Selecting E4 will then look like this;




Playing a scale up will result in a scale going down, around C4. If you want to actually hear the note you are playing as well, call “Default” and that will be added:

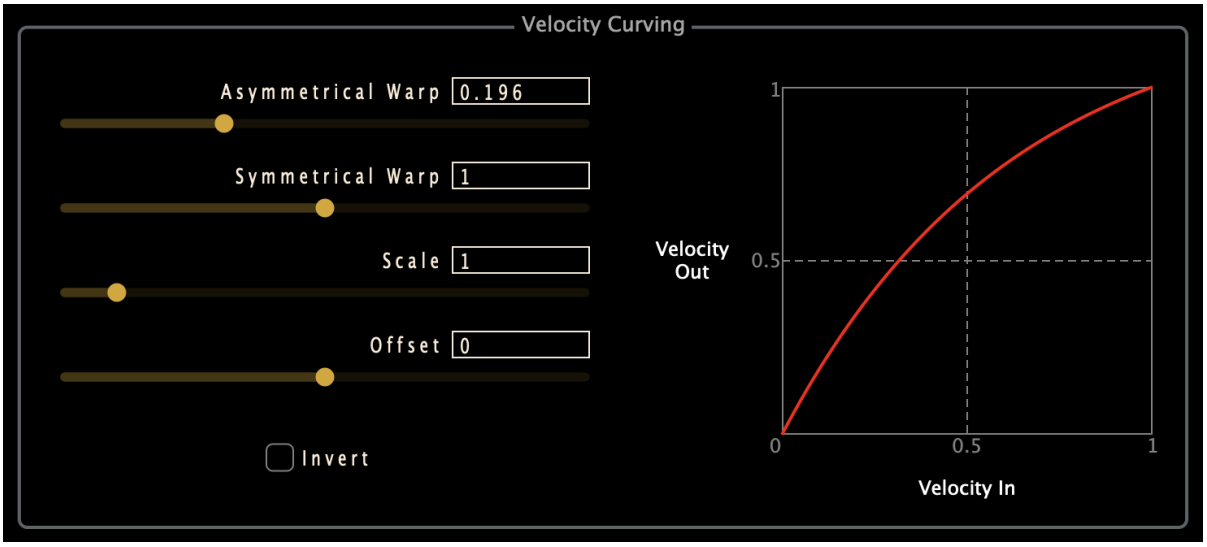



The “transpose” sliders allow you to move the behaviors that you have created up and down, either on the “input” side, the “output” side, or both. You really just have to play with these to see how they work, but this allows for the creation of harmonizations that can then be easily transposed in various ways.

Note that the “midi edit” toggles, which have associated keystrokes, allow use of the MIDI keyboard to interact with the two keyboards, making creation of key-by-key harmonizer maps a bit easier.

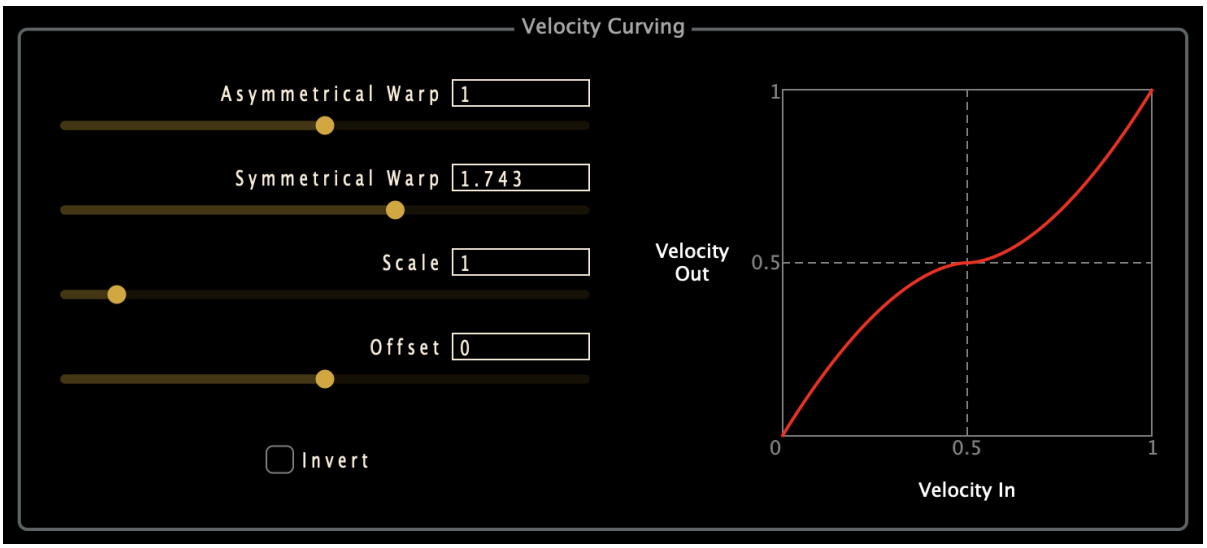
## Velocity Curves

As of v2.9, velocity curves can be created and saved within Keymaps to adjust the feel of your keyboard:

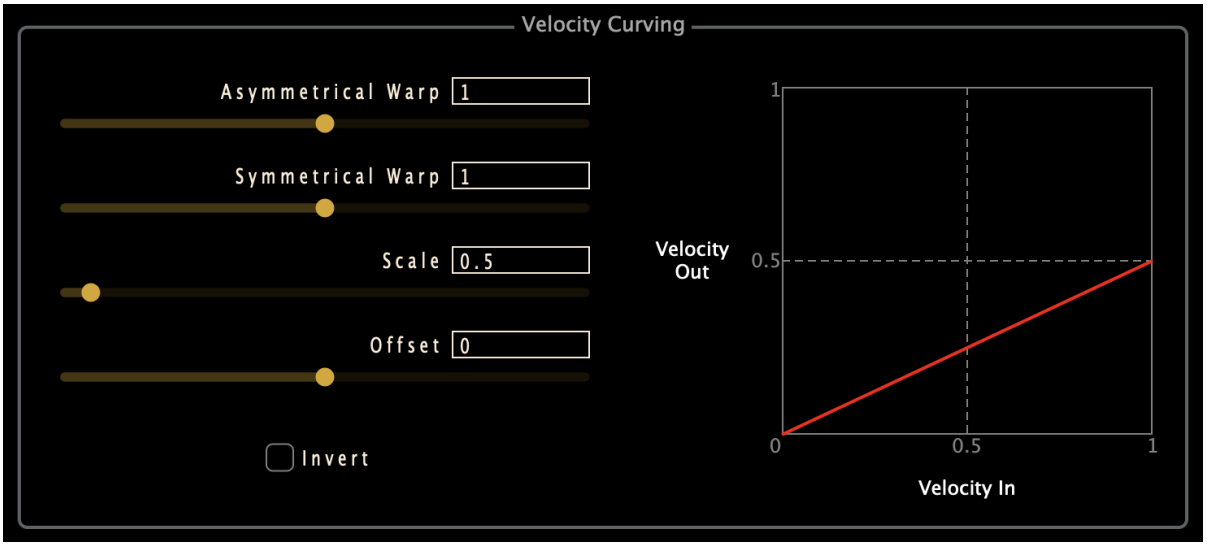



**Asymmetrical warp -** will focus the velocities on either extreme, giving more resolution to higher or lower velocities as desired; this is the most common and familiar type of curve. A default value of 1 leaves the velocities unchanged.

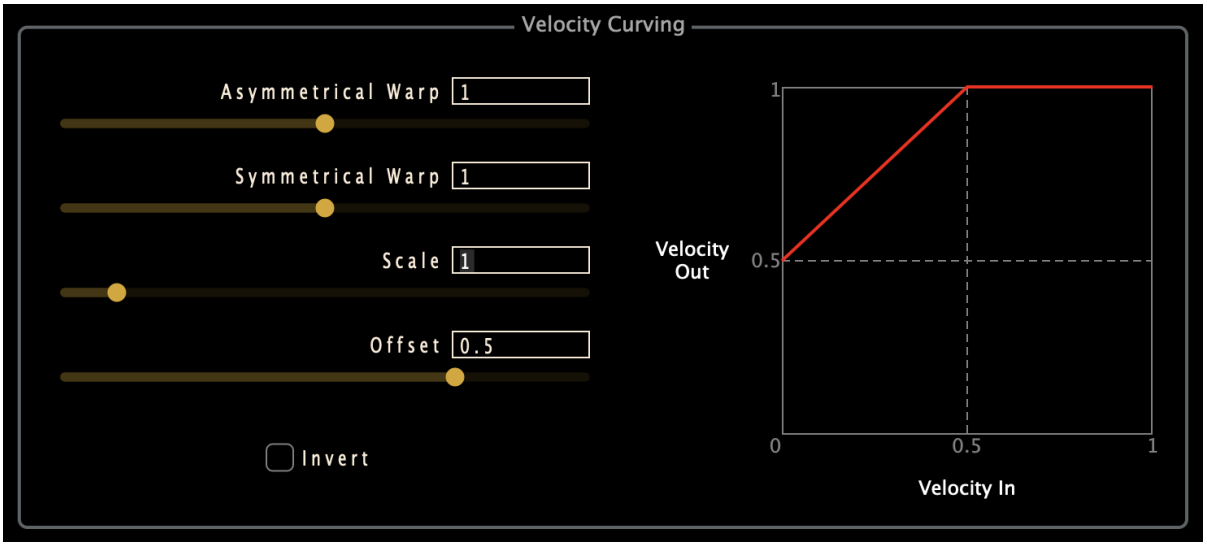
**Symmetrical warp -** will focus the velocities symmetrically around the midpoint velocity, either emphasizing the middle velocities or the extreme velocities. A default value of 1 leaves the velocities unchanged:



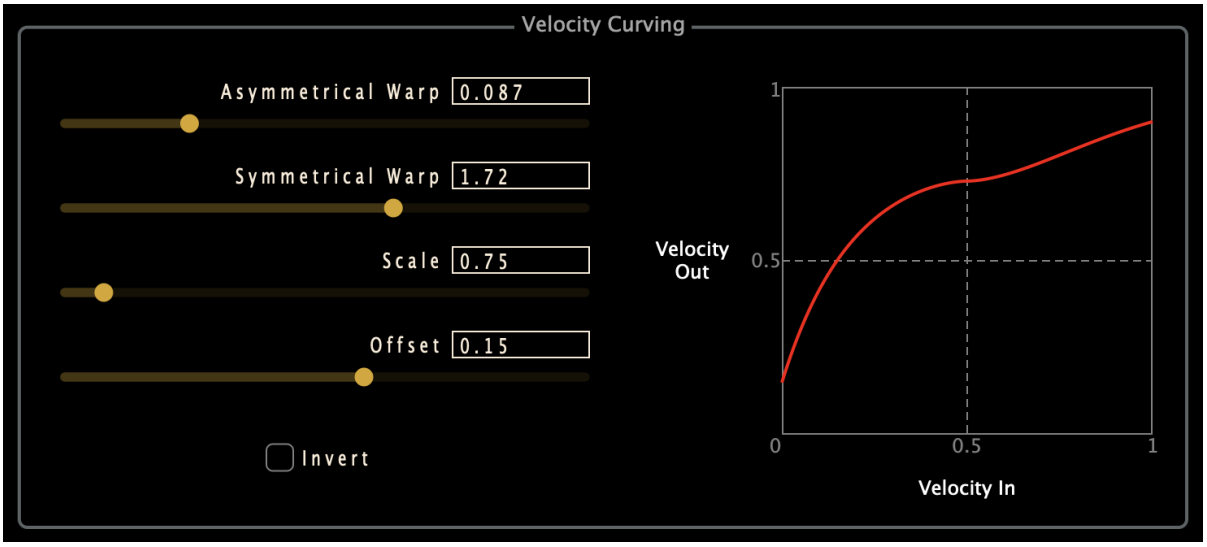
**Scale -** will multiply the incoming velocities by this value. A default value of 1 leaves the velocities unchanged:



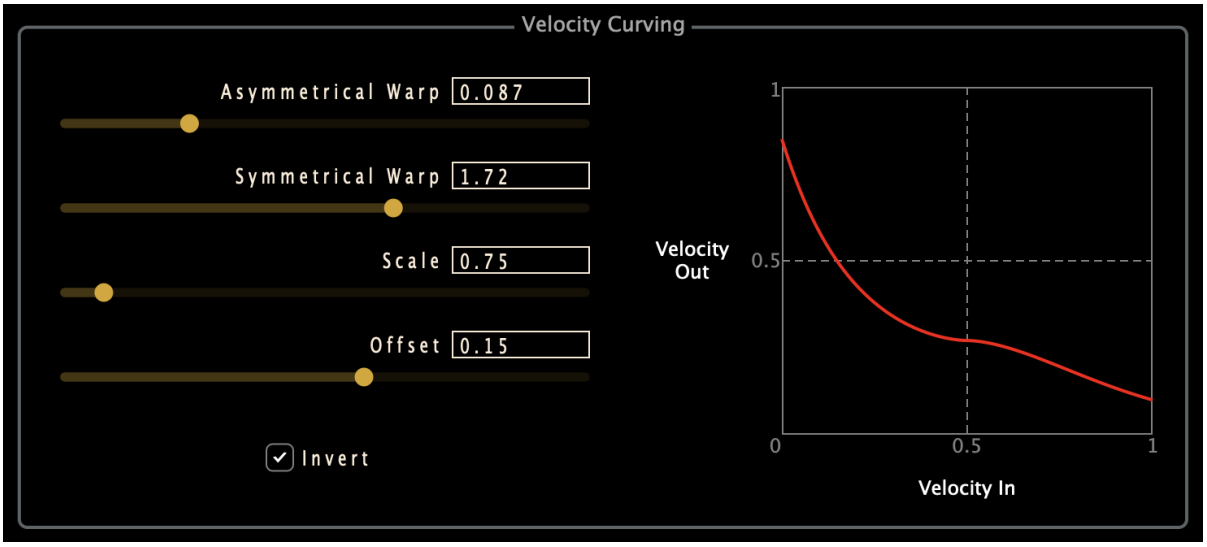

**Offset -** will add this value to the incoming velocities. A default value of 0 leaves the velocities unchanged:




And of course these can all be combined to create various kinds of feels:




And finally, you can **invert** the velocities, so pressing hard will result in a soft note; the curves will be similarly inverted:




## Direct

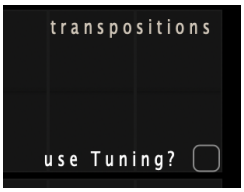
The purpose of the Direct preparation is to provide a minimal sampler interface, and to adjust the overall timbre and shape of the sampled sound while making no major rhythmic or timbral adjustments. It is one of bitKlavier's core preparations alongside Nostalgic and Synchronic, and can be connected directly to a Keymap.

Double-clicking the Direct preparation will open the interface, which includes a number of sliders controlling various parameters:

- Resonance Gain* - volume of pre-recorded keyboard resonance sample, a cross between sympathetic reverb and sustain, plays upon release of key
- Hammer Gain* - volume of pre-recorded hammer signal, plays upon release of key
- Overall Gain - volume of the entire keyboard

**Hammer and Resonance gains are controlled by MIDI noteOff velocity. If your MIDI keyboard does not send noteOff signals, you can set your noteOn velocity as your noteOff velocity under "Gallery>Settings."*


The Direct preparation allows us to tune the bitKlavier keyboard by dragging the large Transposition slider (shift-drag to move by integers, if you don't want microtonal transpositions), or double-clicking the transposition window and typing a value. The Direct preparation can be used as a basic harmonizer as well (though see the Keymap Harmonizer for a more flexible approach to building a harmonizer). Control-click to add additional voices to the slider window, or double-click the window to type multiple values, separated by a space. For an example, check out Example Gallery 21, "Direct 1." Note: The unit for the transposition control is semitones, not cents. As of v2.5.2, transpositions can be filtered by the attached Tuning, by toggling on the "use Tuning" toggle:




Use only integer values if you want the tuning to be applied as expected!

“Velocity min/max” enables a kind of velocity filtering, where only values within range will actually activate the preparation. This can be set with min > max, in which case only incoming velocities *outside* of the specified range will activate the preparation.

The final parameters in the Direct preparation are for envelope control, found in the second Direct panel (use the left/right arrows to the left and right of the preparation window to navigate to adjacent panels).

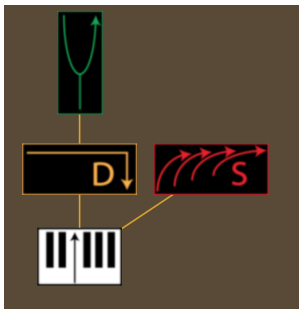

Units for attack, decay, and release are in milliseconds. Sustain by default goes from 0 to 1, with 1.0 being the current Direct gain. You can set this value to be greater than 1, in which case the attack will ramp up to 1, and decay will ramp up again to the given value.

The “blendronic send volume” slider sets how much of this preparation’s signal to send to any attached Blendrónic objects.

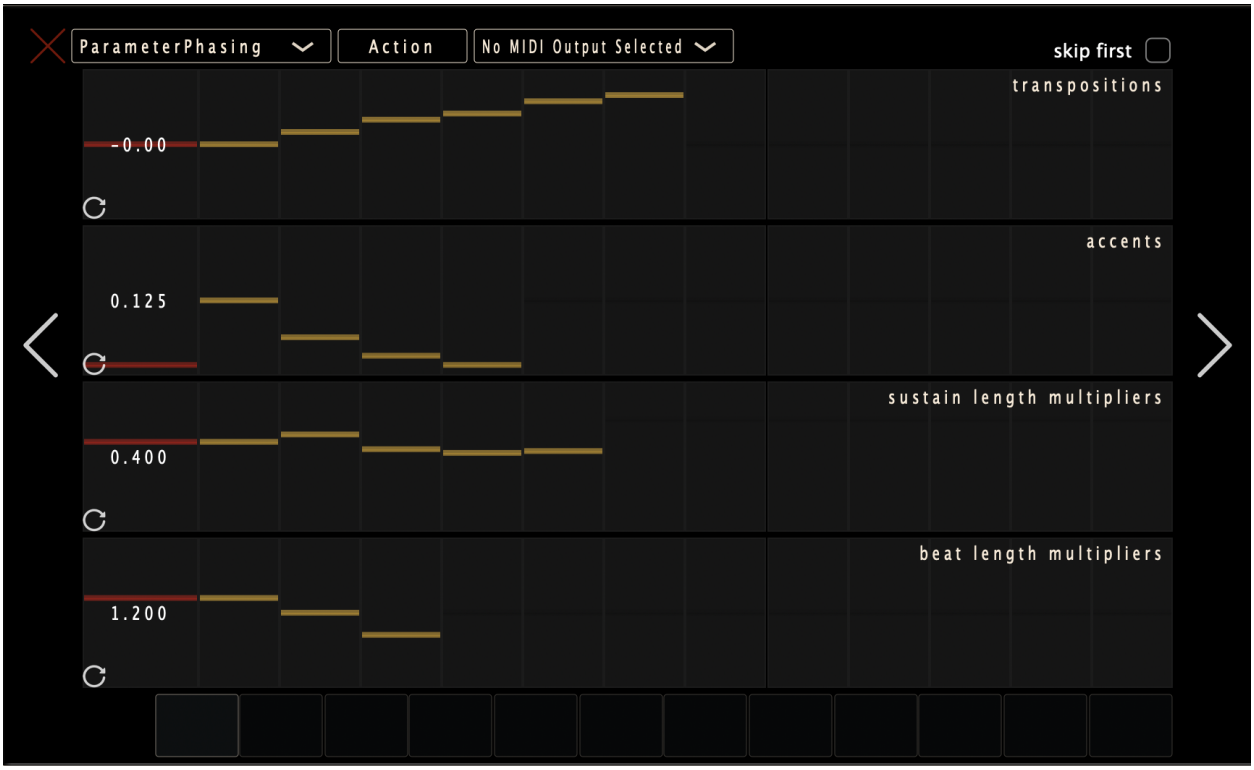
As with all bitKlavier preparations, saving presets is easy and convenient. Simply click “Action” to save, rename, duplicate, delete, export/import, or clear any presets, and use the drop-down menu to the left to navigate the Gallery of presets. Your preparations are saved within the current Gallery, and can be used with other Pianos in the same Gallery, and can be exported if you want to open them in other Galleries. Go to “Gallery > Clean” to delete all unused presets.

## Synchronic

Synchronic is one of the most tweakable and versatile core elements of bitKlavier. In essence, Synchronic uses MIDI input to trigger repeated pulses of notes. It’s like a cross between a delay module and step sequencer with far greater playability and interactivity. Every element is customizable, from the phrasing and pitch of the Synchronic sequence to the cluster size and time threshold required to trigger a Synchronic reaction. In short, the Synchronic preparation can lead to deeply musical results in bitKlavier.

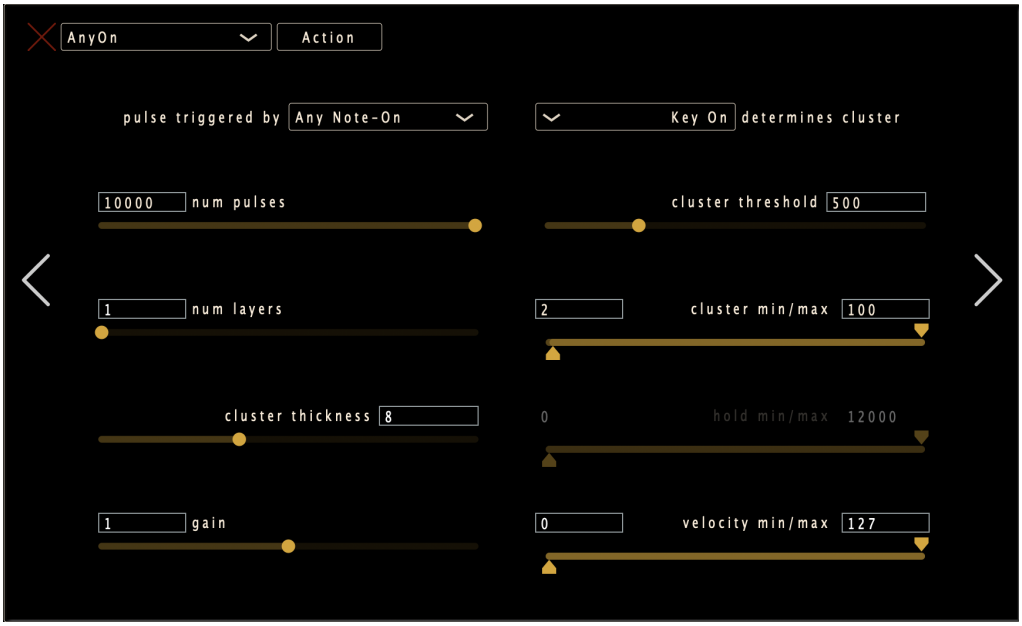



The easiest way to begin exploring Synchronic is to open the Basic Piano Gallery and add a Synchronic preparation (hit “s” or select it from the dropdown “Action > Add...” menu). Connect Synchronic to the Keymap (click-command-drag). Double-click the Synchronic preparation to open the two settings panels:




*Synchronic Window 1: sequenced parameters*

Click the arrows to the left and right to move to the other Synchronic parameters:



*Synchronic Window 2: control parameters*

A quick summary of Synchronic parameters: Note, the parameters in the first window are sequenced effects, each of which allows for up to 12 values (or more if manually added) which Synchronic will cycle through.

**Pulse Triggered By** - what kind of note event in a cluster of notes triggers Synchronic pulses
**Determines Cluster** - determines whether key presses or key releases trigger pulses
**Num Pulses** - number of Synchronic pulses triggered by MIDI input
**Num Layers** - number of simultaneous layers of pulses
**Gain** - volume of the Synchronic notes
**Cluster Threshold** - window of time (in milliseconds) within which Synchronic will group incoming MIDI notes as a cluster
**Cluster Min/Max** - quantity of notes required to *trigger* a Synchronic gesture
**Cluster Thickness** - maximum number of notes that will *sound* in an individual Synchronic pulse
**Hold Min/Max** - range of played durations required to trigger Synchronic
**Velocity Min/Max** - range of played velocities (loudness) required to trigger Synchronic

• • •

**Transpositions** - sequenced pitch offset of Synchronic notes or clusters from ET. Values can be manually typed beyond an octave range, and diads can be added by using bracketed notation (such as [5 -3]) or control-clicking
**Accents** - sequenced gain adjustment of the Synchronic notes or clusters from 0% to 200%, or more if input manually
**Sustain Length Multipliers** - adjusts the duration of a note by dictating how much of the beat is "filled in," so to speak. A value of -2.0 will both reverse and double the length of a single Synchronic note or chord; a value of 0.10 will produce a note of 1/10 the expected duration.
**Beat Length Multipliers** - adjusts the actual length of each beat and time between successive notes, controlling the speed at which the rest of the parameters step through their values. Technically, it divides the tempo, or multiplies the distance between consecutive downbeats. Example: If Beat Length is 0.5 the pulse will change from 120 to 240bpm; Sustain Length of 0.25 will result will be 8th notes at 240bpm.
**Envelopes** - using Synchronic, you can sequence through different ADSR envelope settings. The red square indicates the currently active envelope; the sequencer is synchronized with the other Synchronic parameters.

Using the default Synchronic settings, play any note and you'll hear it repeat on a fixed metronomic pulse. Reading through the settings gives us an overall picture of what is happening: When any cluster of **1-12** notes is played within the space of **500** milliseconds, it

triggers **20** Synchronic repeats, none of which are transposed, sequenced, or otherwise manipulated in any way. Start manipulating settings; notice how things change!

## Determines Cluster Menu

As of v2.4, key releases as well as key presses can be used to determine if and when a Synchronic sequence is triggered. This choice will then impact some of the other available parameters, as detailed below.

## Pulse Triggered By Menu

One remarkable thing about Synchronic—which is, fundamentally, a very simple concept: a resettable metronome—is how many different kinds of musical and interactive behaviors it can exhibit. This is especially evident when determining precisely how and when pulses are launched. Let's look at the different possibilities, beginning with when clusters are determined when keys are pressed, but they may not actually be *launched* until keys are released.

### Key-On Mode

When the determines cluster is "Key On," four drop-down options determine where the Synchronic pulse begins, based on MIDI note-on and note-off signals:

**"First Note-On"**
When you play a series of notes within the cluster threshold, the phase (new downbeat) of the Synchronic pulse will be set by the timing of the first note-on signal recorded within the cluster.

**"Any Note-On"**
The phase of the pulse will be set by the timing of the most recent note played within the cluster. If the cluster threshold is less than the period set by Tempo, then you won't hear a pulse until you stop playing notes. If the tempo is high (say, 250bpm, when the cluster threshold is 500ms), you will hear pulses while you are still adding to the cluster, but the phase will constantly be reset since the pulse is restarting based on the most recent note-on signal.

**"Last Note-Off"**
When you play a series of notes within the cluster threshold, the phase of the pulse will be set by the release of the last note played, regardless of whether it's within the cluster time. In this case, the cluster threshold is still determining which notes are included in the threshold, but you'll only hear them when you release the final held-down note. For example, if you play a single note, the Synchronic sequence won't begin until you release that note. Try playing an arpeggio, low to high: The pulse will only begin when the highest note is released.

**“Any Note-Off”**
When you play a series of notes within the cluster threshold, the phase of the pulse will be set by the release of the most recent note played within the cluster. To test this, play a chord and release notes one by one: You’ll hear the pulse readjust as each successive note-off signal is registered.

### Key-Off Mode

When the cluster mode is “Key Off,” the Note-On modes above are removed, and we have a new Note-Off mode, “First Note-Off.” In this case, clusters are BOTH determined and launched when keys are released (so when you actually pressed these keys down is irrelevant).

**"First Note-Off"**
When you *release* a series of notes within the cluster threshold, the phase (new downbeat) of the Synchronic pulse will be set by the timing of the first note-off signal recorded within the cluster.

**“Last Note-Off”** and **“Any Note-Off”** are as above, in Key-On mode.

## Num Pulses

The “num pulses” slider indicates the number of Synchronic notes triggered by MIDI input. Default value is 20. Change this value to a number under 5 (by dragging the slider or double-clicking and typing into the text field), play a cluster of notes, and count while listening as the Synchronic notes play one by one. A blue bar under the “how many” slider indicates process from the first to last Synchronic pulse. The lowest possible value is 1, which produces a one-time repeat of the cluster notes. The highest value is 100, so Synchronic can essentially repeat a cluster once, semi-indefinitely, or any number of times in between. You can, however, type in a much higher value, and the slider will rescale automatically.

## Num Layers

New as of v2.4, this allows multiple independent Synchronic layers to be activated. With a value of two, for instance, you can trigger one layer, wait, and then trigger a second and both will play simultaneously, with independent phases. If you trigger a third, the first will be replaced—older layers will be replaced first. This is different than simply having two different Synchronic objects with the same settings, attached to the same Keymap; in that case, both Synchronics will be triggered at the same time, and have the same phase.

When in Last Note-Off or Any Note-Off modes, currently sounding layers are impacted as follows when keys are depressed:

- If there is only one layer, it will be silenced until the notes are released; this reproduces the pre-v2.4 behavior
- If there are 2+ layers, then the currently depressed notes will be removed from existing layers. If that results in a current layer having fewer notes than the Cluster Min, then it will effectively be silenced. Otherwise, the layer will continue to sound, with fewer notes.

## Gain

The gain slider at the bottom left of the Synchronic preparation controls only the gain of the triggered Synchronic notes or chords. It is essentially a multiplier for the MIDI velocity. If you play a pianissimo note with a Gain value of 2.0, the Synchronic sequence will play back at twice the gain. If you play multiple notes within the cluster threshold time, each notes will retain its relative value, each independently scaled by the Gain multiplier.

## Cluster Threshold

The Cluster Threshold parameter helps differentiate Synchronic from a standard delay effect or MIDI sequencer. If consecutive notes are played (or released, when in Key-Off mode) within the time indicated by the Cluster Threshold, they will be grouped together as a Synchronic sequence. And as long as the time between notes remains under the Cluster Threshold, new notes will continue to be added to the cluster. For example, with the Cluster Threshold set to 2000ms, any cluster of notes played within 2 seconds becomes a Synchronic pulse. Synchronic essentially has a running timer that resets every time a note is played: If the timer runs out of time before you play a new note (ie. exceeds the cluster threshold), then a new cluster begins. Otherwise, the new note is added to the current cluster, and the timer is reset. You can observe how this is working in real time by looking at the blue progress bars under “how many” and “cluster threshold” in Synchronic.


Controlling what will and will not be included in a Synchronic cluster can be a powerful compositional tool. If you’re playing a fast phrase, they may easily be included in the same cluster. This can be done rather loosely: For example, setting the cluster to <50ms will mean that only very fast notes get clustered together, and >5000ms means most notes will be clustered together. This could also be done very deliberately: If the tempo is set to 120bpm, than any group of four eighth notes played in time will become a Synchronic cluster. Taking into account adaptive Tempo preparation, the possibilities are vast.

## Cluster Min/Max

In Synchronic, you can choose precisely how many notes played within a certain amount of time will trigger a “cluster.” Default is 1-12, meaning that any number of notes up to 12 will trigger a Synchronic reaction. However, this parameter can be used to specify one or several values: With the Cluster Threshold set to 1000 (1 second), set both the min and max cluster number to 7, and play around a bit. When you play a 7-note cluster, either a chord or melodic line played within 1 second, the Synchronic pulse will be triggered. Playing any more or fewer than 7 notes within 1 second will cause the threshold counter will reset.

The Min cluster value can be *greater than* the Max, creating a zone within which a certain number of notes will not trigger a Synchronic sequence. For example, if Min is 4 and Max is 2, then one, two, or four notes can start a pulse, but three will not. You could use this, for example, to have a constant Synchronic pulse where any triad will bring the sequence to a stop.


Note that if you continue playing scales with successive notes spaced by less than the Cluster Threshold, and you exceed Cluster Max, the pulses will stop! If that’s not what is desired, simply increase Cluster Max.

## Cluster Thickness

This caps the number of notes that can be in a cluster (before v2.4, this was internally set to 8, but is now available to the user to set). Let's say Cluster Max is 10, and Cluster Thickness is 8: If you play a chord or arpeggio of 9 notes, a cluster will begin, but that cluster will only have 8 notes. This can lead to some interesting musical effects. For instance, open Example Gallery "Synchronic 1" and play rapid (notes less than 500ms separated) scales of various lengths; a scale of 20 notes will begin playing after you stop, but only the last 8 notes you played will be included in the cluster. Now set it to First Note-On mode, and play the same ascending and descending scales; you'll hear the cluster change over time, shadowing the register of your scale.

## Hold Min/Max

This parameter is only available when in Key-Off mode, as it depends on how long individual notes are held. Notes that are held for durations within the range (in milliseconds) specified by this slider will be included when determining if and how a Synchronic cluster is triggered. Notes with durations outside these limits will be ignored; this acts as a kind of filter that will

allow the player to expressively decide which notes are included or not. Note that Min can be greater than Max, in which case notes in the middle will be ignored.

## Velocity Min/Max

Similar to Hold Min/Max, this parameter sets a range of velocities (how “hard” the keys are pressed) that limit the notes that might trigger a Synchronic cluster pulse. Again, Min can be greater than Max, essentially enabling notes at the extremes to trigger clusters, and notes in the mid ranges will be ignored.

## Transpositions

One of five sequenced effects, the Transpositions sequencer determines how the Synchronic pitch will shift over time. Each column from L-R represents absolute semitone offset from the played pitch. A sequence of 0 1 2 3 4 5 6 7 8 9 10 11 12 will play an ascending chromatic scale.

To begin exploring the Transposition setting, drag the single bar on the far left up until it reads 2.0 (you can also double-click the Transposition box and manually enter the value). Play A4, and note that after the initial MIDI input, every Synchronic note will be B4 (A4 + 2.0 semitones).

In the next column to the right, click and drag a bar with value -1.0 (this can also be done by double-clicking the Transposition box and entering a second value, separated by a single space). Play A4 again, and observe that for the duration of the Synchronic repeats (20 by default), the pitch will go between B4 and G#3 (A4 + 2.0 semitones, then A4 - 1.0 semitone).

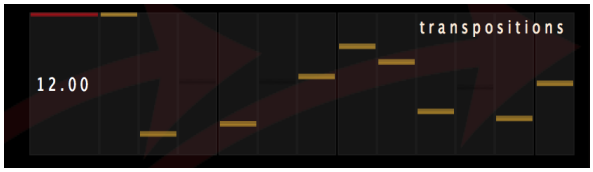
As you can probably guess, using the Transpositions sequencer can lead to some wild results. Double-click the Transposition box near the left-hand side and copy the following values:

0 12 19.02 24 27.86 31.02 33.69 36

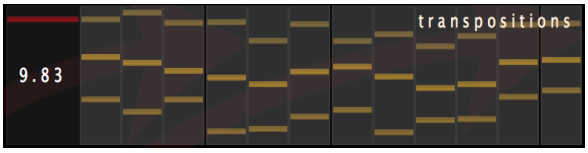
Play a single note, and the Synchronic preparation will cycle that note through the first values of the harmonic series. Try this with chords, arpeggios, clusters. Try adding a (-) to every value to hear the undertone series. Experiment with values that go in both directions to create an oscillation: Copy the following sequence of values, then play a sustained note or cluster and listen to how pitches go in and out of phase with one another:

0 0.2 0.4 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.4 -0.2

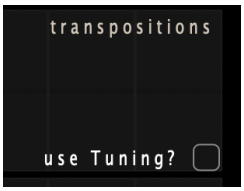
Aside from dragging individual bars, or inputting numerical values for each step, you can click and drag more gesturally to create a shape. Here’s the result of randomly dragging the mouse from one side of the Transpositions window to the other:




Multiple transpositions can be added for each sequenced step: Any values within a set of **brackets** (like [-5.5 3.0] or [2 4 7]) will be interpreted as a cluster. You can control-click to add a new transposition, and can add more than 12 values (the window will automatically re-scale).



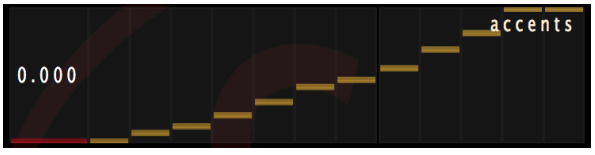

As of v2.5.2, transpositions can be filtered by the attached Tuning, by toggling on the “use Tuning” toggle:




Use only integer values if you want the tuning to be applied as expected!

## Accents

The Accents setting uses the same principle of the Transpositions setting in that we program a series of values through which bitKlavier’s Synchronic preparation can cycle. Let’s start with a blank slate: Go to the Action button near the upper-middle of the Synchronic settings window and select “clear” to return to Synchronic’s default settings. Now, draw a rough amplitude ramp by clicking in the lower-left and dragging to the upper-right:




By default the lowest value in the window will be 0.0, meaning no amplitude at all, and the highest value will be 2.0, or double the amplitude. Values for Accents are multiplied in relation to the initial MIDI velocity value and are not cumulative. If you have a value of, say,

1.0, 1.0, 1.0, you won't end up with a final note with velocity 3.0 - instead, you'll have three notes of the same initial velocity.

By double-clicking the Accents box, you can input values greater than 2.0, which will automatically change the visual scale of the other bars. This isn't necessarily recommended because high amplitude values could make the samples clip during playback, but for a piece that's largely pianissimo (MIDI velocity <15.0 or so), it could be very effective to have one periodic loud accent. You can also add more cells/steps by typing more values in the edit field.


## Sustain Length Multipliers

Using the same interface as Transpositions and Accents, the Sustain Length Multipliers sequences through values for "sustain" in the ADSR envelope, with negative values reversing the note. A value of -2.0 will reverse *and* double the length of a given Synchronic note or chord, and a value of 0.10 will produce a note of 1/10 the expected duration. Try the following example values to hear the effect of the Sustain Length Multipliers more acutely:

Example 1: Ramp from 2.0 Reverse to 2.0 Normal
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2

Example 2: Dash Dot Dot Dot, Reverse, Dash Dot Dot Dot
1 3 0.5 0.5 0.5 0.5 -3 0.25 0.25 0.25 0.25

## Beat Length Multipliers

Changing the Synchronic beat length is a drastic effect, allowing the user to deviate from the metronomic pulse of the preparation. By default the value is 1.0, which is 120bpm until we change it using a Tempo preparation (more on that soon). Any deviation from 1.0 is a multiplier of the quarter note value. Try setting the second column to 2.0, a half note, and listen to how the pulse shifts from **||:** quarter-quarter **:||** to **||:** quarter-half **:||**

Copy and paste this sequence to have a Synchronic sequence of a recognizable rhythm:

1 1 2 1 1 2 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 2

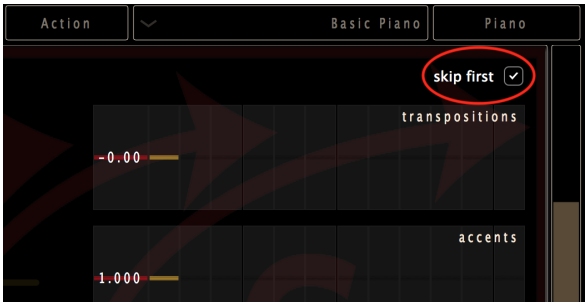
Here's a custom sequence of numbers used to simulate a sort of strumming pattern:

0.8 0.1 0.02 0.02 0.02 0.02 0.02

Note: Entering a negative value in the Synchronic beat length multiplier window will cause the Synchronic preparation to stop working. Synchronic cannot parse a beat length with value less than 0.

## Skip First

The “skip first” feature tells Synchronic to jump over the first value in all of the sequenced parameters on the first pass (Transposition, Accent, Sustain Length Multipliers, Beat Length Multipliers, Envelopes). This is useful for when you want the “downbeat” to be exactly what you play, rather than arriving one pulse after you play it. “On” is the more common setting and is used in most Synchronic Example Galleries. Compare Synchronic Examples 1 and 8 (Skip First *on*) with Examples 2 and 9 (Skip First *off*).




Note that in v2.4.1 there was a bug that caused skip-first to misbehave in some circumstances; this has been fixed in v2.5, but it’s possible that Synchronic preparations made in v2.4.1 will need to be modified to reproduce the desired behavior.

## Envelopes

Twelve boxes along the bottom each include their own envelope settings. Using Synchronic, you can program and sequence various envelope settings in conjunction with other sequenced settings. Click any box to open an ADSR settings panel; when you adjust any of the sliders, they will go from greyed-out to bright, and this envelope will be toggled on. Press “close” to leave the ADSR view. Shift-click on individual boxes to turn on/off individual envelopes.

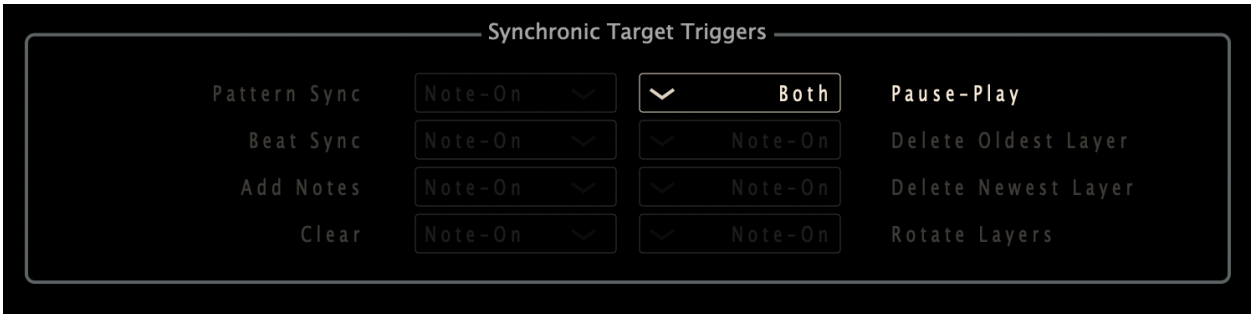
When a Synchronic reaction is triggered, bitKlavier will step through these envelope values just like any other setting. The currently active one will be in red.

## MIDI Out

As of v2.5, Synchronic can target external MIDI devices, sending noteOn and noteOff messages for each pulse. This is an experimental feature!

## Synchronic Targets

When a Keymap attached to Synchronic has one more more Synchronic targets activated, they will be bright in the Synchronic preparations Target Triggers window (reached by pressing the left/right arrows <> on the left/right sides of the preparation window); in the image below, Pause-Play is enabled in the Keymap object, so its pair is bright in the Synchronic Target Triggers window:




Synchronic can then be set to respond to the triggers it receives in particular ways: when it receives a noteOn, a noteOff, or both. In this example, Synchronic will pause on noteOn, and then begin playing again on noteOff, since it is in “Both” mode. These targets behave as follows:

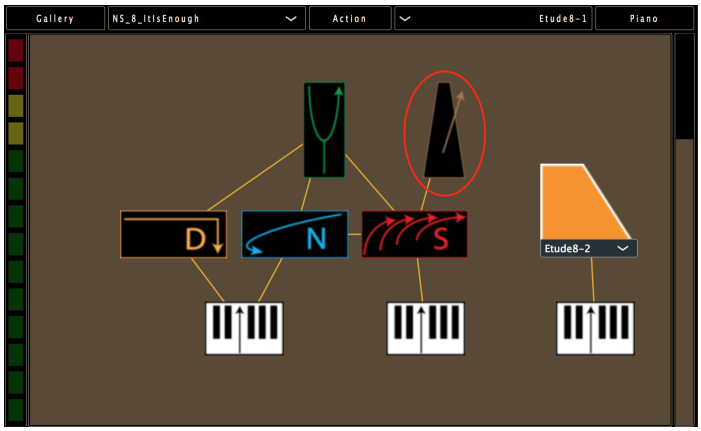
**Pattern Sync** - restarts the patterns (Accents, Transpositions, etc…) on the next beat; the pulse itself is NOT reset.
**Beat Sync** - the pulse is reset, but the patterns are not. The normal Synchronic behavior is for both of these to be reset, so this allows them to be decoupled.
**Add Notes** - played notes are added to the cluster without resetting the patterns or pulse.
**Clear** - clear all ongoing clusters/pulses.
**Pause-Play** - pause all clusters if they are playing, or restart playing them if they are paused.
**Delete Oldest Layer** - if using multiple layers, this will remove the oldest one.
**Delete Newest Layer** - if using multiple layers, this will remove the most recent one.
**Rotate Layers** - this will rotate the layers so the phase and state of an older layer can be changed.

• • •

Synchronic is a powerful tool and is used throughout Dan Trueman's *Nostalgic Synchronic* and the *Mikroetudes* written specifically for bitKlavier. Synchronic is also featured in many of the lessons and examples from Dan Trueman's "Reinventing The Piano" online Kadenze course. For more information and inspiration on using Synchronic in bitKlavier, check out the above links, as well as the Synchronic Examples in the bitKlavier Galleries.

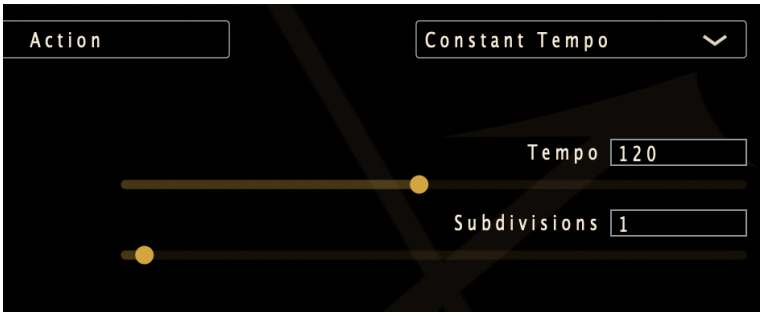
# Tempo

The Tempo preparation provides control to any connected Synchronic preparation in bitKlavier, just like the Tuning preparation alters the Direct and Keymap preparations. There are two Tempo settings, **Constant Tempo** and **Adaptive Tempo**. To begin using the Tempo preparation, build the Basic Piano, add a Synchronic preparation, then create a Tempo preparation – key command (m) – and connect it to Synchronic. The Tempo preparation now controls that Synchronic preparation: If using multiple Synchronic preparations, a single Tempo preparation can control all of them, or each Synchronic preparation could use its own individual Tempo preparation. Tempo only controls the preparations to which it is connected.




## Constant Tempo

A single slider controls metronome values from 40.0 to 208.0, or any higher/lower positive value can be typed into the window. Play a note to trigger a Synchronic pulse, then adjust the slider to hear how Tempo changes the Synchronic speed.




The “subdivisions” slider (new as of v2.4) sets how many pulses will be sent per “beat” as set by the “tempo” slider; for instance, when subdivisions = 4, and the tempo = 120, this will be interpreted as pulses at 480bpm; this should allow for more intuitive tempo settings, especially when working within a DAW.

## Adaptive Tempo

Adaptive Tempo determines the pace of a Synchronic sequence by taking a running average of the time between user-played notes. Adaptive Tempo will only work if it is attached to a Synchronic preparation *and* a Keymap preparation, because the input from the Keymap provides information that adapts the tempo to your playing.

Adaptive Tempo Settings:

- **History** - number of MIDI strokes bitKlavier is retroactively counting to find the average time
- **Subdivisions** - tempo multiplier by interpreting the history. If you play 4 quarter notes and subdivision is set to 2, it doubles the tempo by basing the current tempo value on four eighth notes. Values less than 1 are also possible, to create slower tempi.
- **Min and Max** - tells Tempo to ignore notes above or below a certain length and spacing

For more information on how to use the Adaptive Tempo feature, load the corresponding example Pianos from the Examples Gallery (Adaptive Tempo 1-4, Examples 23-26) and explore how they function.

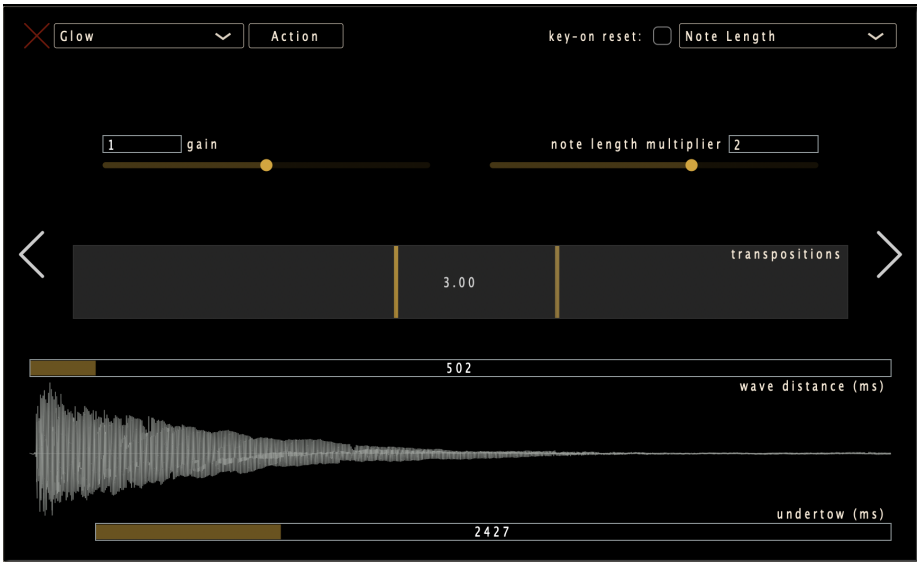

### Host Tempo

When using bitKlavier as a plugin, choosing “Host Tempo” will enable bitKlavier to follow tempo changes of the host directly.

## Nostalgic

The Nostalgic preparation is another way that the digital prepared piano looks beyond the on/off binary that a traditional acoustic piano key represents. One of bitKlavier’s core preparations, Nostalgic allows bitKlavier users to program notes that play in reverse when a key is lifted, or when a key is pressed with timing set by an attached Synchronic preparation. Alongside the Synchronic preparation, bitKlavier’s Nostalgic preparation provides a feature-rich set of parameters that control the volume, swell, and precise sound of the instrument *after* notes have already been played.

The Nostalgic preparation has three windows (as of v2.4), accessed via left/right arrows, as follows:




*Nostalgic Window 1: the main parameters*


*Nostalgic Window 2: the performance control parameters*


*Nostalgic Window 3: the envelope parameters*

To dive in, add a Nostalgic preparation to the Basic Piano – key command (n) – and connect it to the Keymap preparation. Double-click on the Nostalgic preparation to open the settings panel. A brief overview of the parameters, moving through the windows:

**Key-On Reset Toggle** - resets currently sounding Nostalgic notes when they are played again.

**Note Length Multiplier** - takes value of note duration and multiplies it. A whole note at 60bpm will have a value of 4 seconds: With multiplier at 2.0, Nostalgic response will be 8 seconds.

**Gain** - controls the volume of the Nostalgic response. A MIDI note or chord of velocity 50 with Gain set at 1.5 will result in a Nostalgic response of 75.

**Wave Distance** - controls the start and end point of the Nostalgic playback, in milliseconds from the beginning of the sample. This value can be dragged from left to right or double-click to add a numerical value. Change this value to 500 and observe that the Nostalgic response, while still equal to the duration of the initial note, starts and ends .5 seconds before the head of the sample.

**Undertow** - provides a fixed value for a second Nostalgic response after the initial reversed sample. Units are also in milliseconds, and the undertow begins wherever the Wave distance is set. Change the undertow value to 500 and observe that after the Nostalgic response, there will be a .5 second bounceback of the sample.

**Hold Time** - sets the range of performed note lengths that will launch Nostalgic notes. *Currently only implemented when in Note-Length mode.*

**Velocity Min/Max** - sets the range of performed velocities that will launch Nostalgic notes.

**Cluster Min** - how much time there is between note *releases* (cluster threshold) determines whether Nostalgic notes are launched. *Currently only implemented when in Note-Length mode.*

**Cluster Threshold** - sets the amount of time between successive note releases for notes to be included in a cluster.

**Reverse ADSR / Undertow ADSR** - provides ADSR control of both the Nostalgic response and the undertow.

**Transpositions** - controls the pitch of the Nostalgic response and undertow. Note, multiple values can be added to create clusters of Nostalgic notes, shift-drag to limit to integer transposition values.

## Nostalgic: General Notes

The Nostalgic preparation is highly interactive and customizable. A blue tracker line in the Nostalgic waveform window provides real-time indication of how Nostalgic is moving through the sample. Other settings also provide a visual indication of how Nostalgic is altering the sound.

For every MIDI key off, a new Nostalgic note is created, and every note in the attached Keymap preparation will have a Nostalgic response. In the default mode (Note Length), a timer begins as soon as a key is pressed: When that key is lifted, that same duration is applied to the Nostalgic note. This can easily be observed by playing several notes, holding them, and lifting them one at a time. The same is true of Undertow notes too (more on that below). For more clarity while learning how Nostalgic works, adjust the Transposition to an obvious interval (such as +2.0 semitones) and listen back to how and when the Nostalgic notes occur.

In Synchronic Sync modes, as opposed to Note Length modes, the length of held notes does not matter, since the Nostalgic duration is set be the relationship between noteOn signals and the Synchronic pulse. More on that in the Synchronic Key Up/Down section below.

## Key-On Reset

When this is toggled on, currently sounding Nostalgic notes will be interrupted when their notes are played again. Of course, new ones will be launched as well! But this is a way of thinning out Nostalgic textures.

## Note Length Multiplier

You can choose how long you want your Nostalgic notes to be as a function of the duration of the input note (defined as the time between MIDI key down and MIDI key up). If you strike a key and release it 1 second later, the Nostalgic response will have that same duration of 1 second. The Note Length Multiplier multiplies the note duration when applying it to a

Nostalgic wave. A whole note at 60bpm will have a value of 4 seconds: With a multiplier of 2.0, the Nostalgic response will be 8 seconds. This can be used to create very short Nostalgic responses too.

Note Length Multiplier is the only parameter that controls the actual length of Nostalgic notes. Wave Distance and Undertow control the start and end points of the *sample*, but the duration is entirely controlled by the Note Length Multiplier when in Note Length mode.

## Gain

Gain controls the volume of the Nostalgic response. A MIDI note or chord of velocity 50 with Gain set at 1.5 will result in a Nostalgic velocity of 75. This can be used to create Nostalgic notes and clusters that are faint, thunderous, and everything in between.

## Wave Distance

The slider above the Nostalgic waveform window controls the start and end point of the Nostalgic playback in *milliseconds from the beginning of the sample*. This value can be dragged from left to right. As usual, double-click to add a numerical value. Changing this value alters the location of the blue tracker line, which indicates the end point of the Nostalgic wave.

Change the Wave Distance value to 500 and observe that the Nostalgic response, while still equal to the duration of the initial note, starts and ends .5 seconds before the head of the sample. If you play a note of around one second in length, the blue line will travel for one second until it reaches the end point indicated by the Wave Distance parameter.

## Undertow

Using the Nostalgic preparation, we can have an “undertow” response as well, which provides a second, forward-motion playback of the sample after the initial Nostalgic notes. Undertow *begins* precisely where Nostalgic *ends*, as indicated by the Wave Distance parameter.

The slider for Undertow beneath the Nostalgic waveform determines the fixed length of the Undertow in milliseconds. When Undertow is set above 0, every Nostalgic note will have a corresponding Undertow. The Undertow will last only the length you indicate: It does not change with Note Length or Synchronic Sync settings. An Undertow of 1000ms will last 1 second after the Nostalgic cluster has ended, regardless of the length of the Nostalgic notes.

Set the Undertow slider to a value above 250ms, play a note, and watch the blue tracker line: At the end of the Nostalgic wave, the line will bounce back to the right as it plays through the sample again for one quarter of a second. Play a triad, release the notes one at a time, and observe how each Undertow is generated independently.

## Hold Time

Keys that are held down for durations set by these sliders will launch Nostalgic notes; so, for instance, you may want to only have longer durations launch Nostalgic notes. Min can be greater than Max, which means that durations between Min and Max will be ignore. *Currently only implemented when in Note-Length mode.*

## Velocity Min/Max

Similar to Hold Min/Max, this parameter sets a range of velocities (how “hard” the keys are pressed) that limit the notes that might trigger a Nostalgic notes. Again, Min can be greater than Max, essentially enabling notes at the extremes to trigger notes, and notes in the mid ranges will be ignored.

## Cluster Min and Thresh

This sets the minimum number of notes that have to be *released* within a small amount of time (the *cluster threshold*) to launch Nostalgic notes. So, if this is set to, say, 3, and you release two notes together, they won’t launch Nostalgic notes. Note the the threshold is between successive notes, so the timer resets every time you release a note; if the threshold is 150ms, and you release 10 notes in row, all 100ms apart, they will all be included in the cluster measurement. *Currently only implemented when in Note-Length mode.*

## Cluster Threshold

The maximum amount of time between successive note releases for notes to be included in a cluster. Only relevant when Cluster Min > 1.

## Reverse Envelope / Undertow Envelope

You can control ADSR values for Nostalgic and Undertow by using the bitKlavier envelope interface. For a clear example, try some fairly intense settings. With the following values, you won’t be able to discern the end of a Nostalgic wave from the beginning of the Undertow:

**Reverse Envelope**: Attack (700), Decay (600), Sustain (.6), Release (400)
**Undertow Envelope**: Attack (500), Decay (700), Sustain (.8), Release (500)

It’s important to note that envelopes always run forward in time, even when being applied to the Nostalgic note, which plays in reverse. In other words, Attack is applied as the reversed note begins, *not* at the “attack” of the original sample. To control the crossfade between Reverse and Undertow notes, you’d adjust the Reverse Release and the Undertow Attack times.

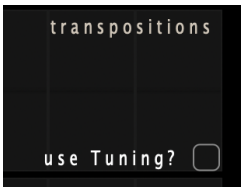
## Transpositions

Nostalgic and Undertow don't have to be the same pitches as the notes that triggered them. Simply drag the slider to adjust the pitch of the Nostalgic note. This could be a subtle effect, such as having the Nostalgic response be pitched ever-so-slightly lower than the initial input, or a drastic effect, creating crashing chords and dense clusters.

As with many bitKlavier parameters, double-clicking the Transposition window in the Nostalgic preparation allows for additional values. Type or paste (-12 7.0 12) to have a Nostalgic cluster response of an octave lower, a fifth higher, and an octave higher from the initial pitch. Copy the following values to have Nostalgic generate the first eight overtones of the harmonic series:

0 12 19.02 24 27.86 31.02 33.69 36

As of v2.5.2, transpositions can be filtered by the attached Tuning, by toggling on the "use Tuning" toggle:




Use only integer values if you want the tuning to be applied as expected!

## Synchronic Key Up/Down

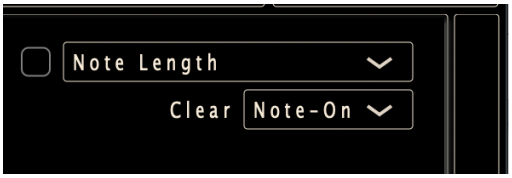
The Synchronic Sync KeyDown/KeyUp modes control the length of the Nostalgic swell based on a the pulse of a linked Synchronic preparation. *Nostalgic* **must** *be connected to a Synchronic preparation in the same Gallery for Sync mode to work: If Nostalgic is not connected to Synchronic, only the Undertow will play.*

In KeyUp mode, the Nostalgic note or cluster begins when the key is *released*, as it does in Note Length mode, and peaks in tandem with the next Synchronic beat (or later, depending on "beats to skip" slider). In KeyDown mode, the Nostalgic note or cluster begins *as soon as the key is pressed*, and will peak at the next Synchronic beat (or later, depending on "beats to skip;" this slider replaces "note length multiplier" slider when in a KeyDown mode).


Check out the Example Galleries Nostalgic 4-6, or “Hurra” from *Mikroetudes,* for examples of how to use the Sync mode between Nostalgic and Synchronic.

### Nostalgic Targets

Nostalgic has only one target that can be triggered by a Keymap: “clear,” as seen below.



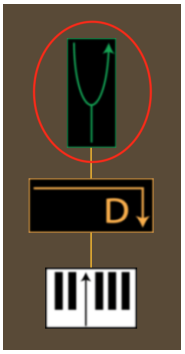

This can be set to be triggered by noteOn or noteOff messages, or both, and it will be greyed out if there isn’t an attached Keymap set to target “clear.”

• • •

For more information on using Nostalgic, check out the corresponding sessions in the “Reinventing The Piano” Kadenze Course, as well as the *Nostalgic Synchronic* etudes.

## Tuning

The Tuning preparation is another of the control features which sets bitKlavier apart from both the acoustic piano and the more conventional sampler. Tuning provides a hugely flexible array of musical and compositional opportunities, from creating and manipulating a custom temperament to manually editing the precise pitch of every key on the keyboard.




To begin exploring the Tuning preparation, open the “Basic Piano” and double-click the Tuning preparation, which looks like a green tuning fork. There are a series of dropdown menus and buttons on the left-hand side that should

be familiar from the other preparations. “Action” allows us to manage presets and other global actions, including saving and naming presets, clearing the settings, and duplicating the preparation.

Summary of Tuning settings:

**Action menu –** menu in the upper-left allows for saving and naming Tuning presets

**Adaptive tuning menu –** currently active adaptive tuning system (if any)

**Temperament menu –** menu in the upper-middle of screen includes many preset temperaments which will be loaded into the temperament keyboard. See Appendix for Tuning details.

**Root menu –** determines pitch to serve as the root from which the temperament is generated

**Temperament keyboard** - graphic interface where each note of the one-octave temperament can be manually adjusted or typed into the “edit all” menu

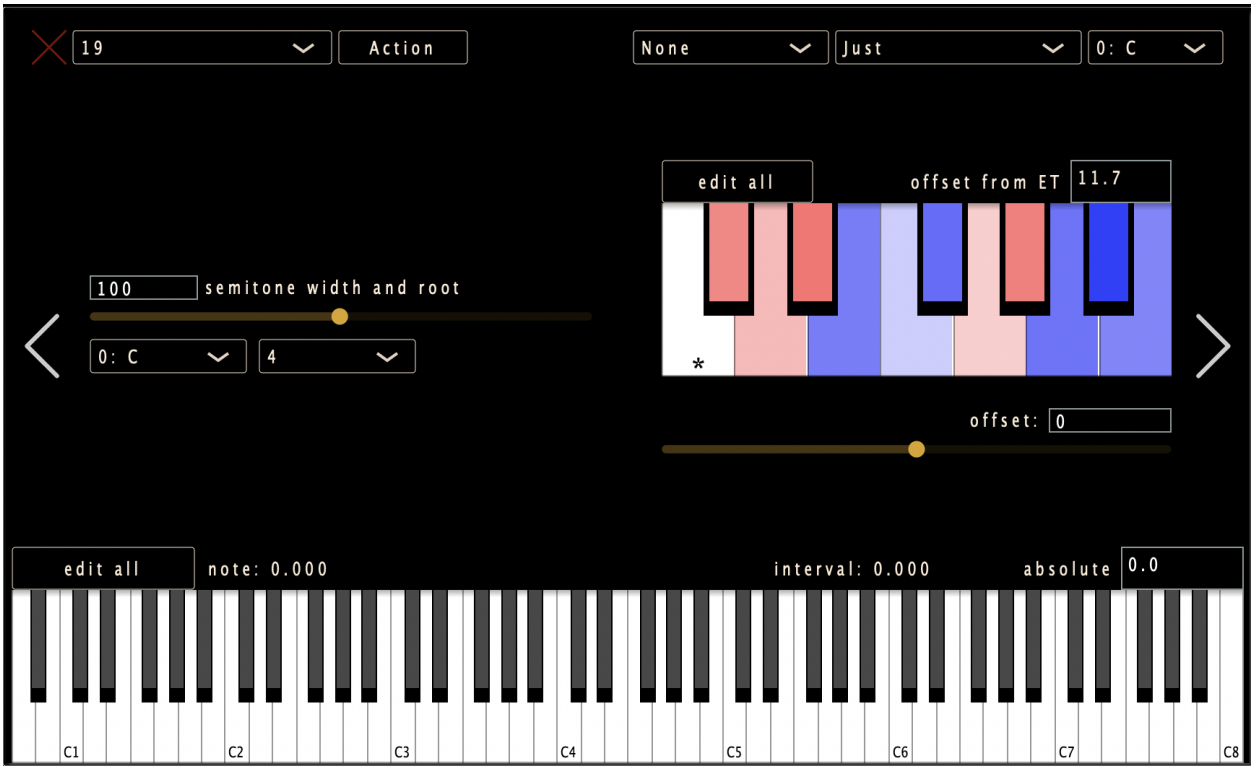
**Offset –** transposition from system tuning (default 440hz, can be changed in Gallery > Settings)

**Semitone width and root –** defines “semitone,” which by default is 100 cents. 50 will be a quarter-tone keyboard. Root tells which note to begin on - if root is A4, then A4 will be tuned to system settings (440hz by default), regardless of semitone width

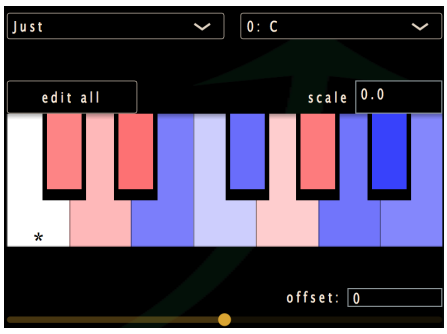
**Tuning Keyboard –** a full 88-key keyboard along the bottom allows you to tune each note of the keyboard manually, by click-dragging or entering the values manually. The format is “MIDI note: cent value” (no quotations), so 60: -10.5 will lower C4 by 10.5 cents

## Temperament

Let’s start by looking at the Temperament section on the right-hand side of the Tuning preparation window. To review, a temperament is essentially a one-octave tuning scheme that adjusts certain intervals in order to prioritize consonance between them. A temperament is repeated *every* octave of the keyboard. The root of the temperament can be determined by the user — more on that soon.




The drop-down menu to the second-from-right contains presets for different tunings. Full charts for these tunings are located in the Appendix at the end of the manual. The default temperament is Equal Temperament: Click the drop-down menu and select “Just” temperament. Note how the temperament keyboard will visually change to reflect how each interval has been tuned from Equal Temperament. **Blue** means flat, **Red** means sharp, and the intensity of the color reflects the degree to which a pitch has been raised or lowered from ET.

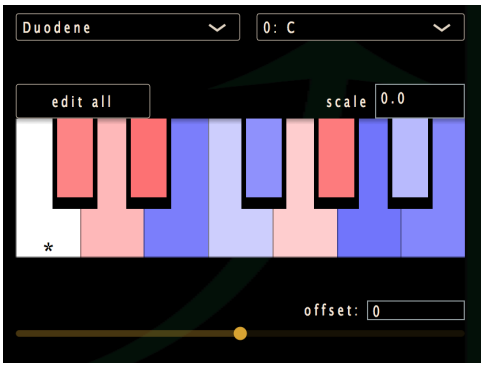


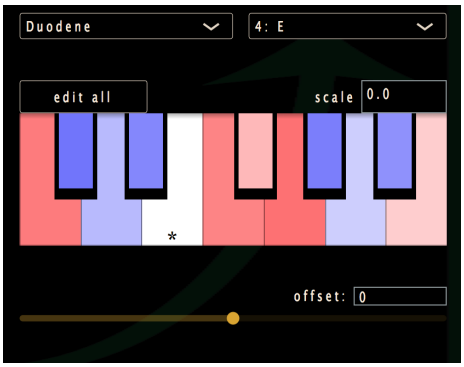

Each key can be manually adjusted: Go back to Equal Temperament, click D, and begin to drag *up* while holding the mouse down. Notice that several things change: As the D becomes more red to indicate a rise in pitch, the temperament changes from Equal Temperament to Custom. The pitch of each key can also be changed by clicking at various locations on the surface of each key.

The “cents from ET” window at the upper end of the temperament keyboard, which shows the specific amount the pitch has been altered, changes in real time as well. Play along the keyboard, and notice that every D is now pitched quite a bit higher.

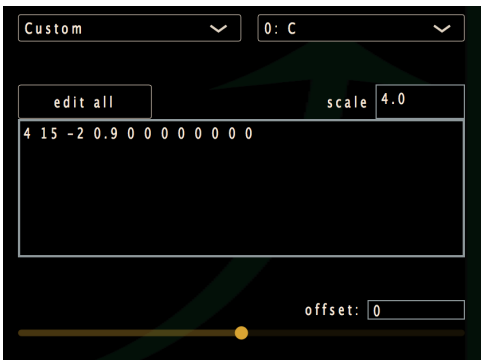
Let’s say we want to create a temperament where every G is lowered by 8 cents. Simply select G on the temperament keyboard and type -8.0 into the “scale” window. Now play C5 and G5 on your MIDI keyboard, and note that 5th doesn’t sound quite as “perfect” as it does in ET. Go along the keyboard and you will find that every G is now slightly lower.

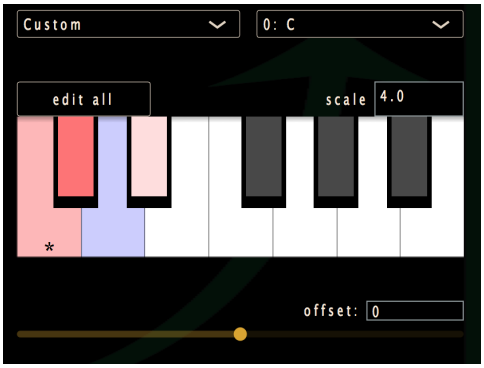
You’ll notice an asterisk on C. This indicates that C is the root, meaning that the temperament is generated using a fixed scheme of offsets beginning on C. Try selecting Duodene from the drop-down menu. Notice that the first few notes of the temperament are slightly raised, while the last few notes of the temperament are slightly lowered. Now, change the root of the temperament to E, and notice how this pattern is translated 4 semitones higher.




For greater depth and control, all 12 notes in a temperament can be quickly edited via a text field. With Equal Temperament selected, click “Edit all” to reveal a field of 0s. Each of these corresponds to the current pitch value of each ascending note in the temperament.



Change the first few numbers and click “Edit all” again (or press the return key) to view the temperament keyboard; note the visual changes based on the values you typed.

A slider below the temperament keyboard moves the entire pitch of the keyboard up or down. To quickly raise the pitch of the keyboard, move the slider to the right, or change the “offset” pitch to a higher number (the unit is cents). Type 100 or -100 into the “offset” box to immediately raise or lower the temperament keyboard a semitone.

## Semitone Width

Changing the semitone width in bitKlavier can lead to some wild results. Semitone width simply translates to “number of cents in every semitone,” or half step. By default, this value is 100. Changing this value to 200 means that every semitone on the keyboard is now a whole step. By extreme contrast, changing this value to -100 inverts the keyboard. As you ascend the physical keyboard, the pitch becomes lower.

Of course, the “semitone width” feature can be used in more subtle ways as well. Change the value to 102, which adds 2 cents to every semitone. Over two octaves, the notes are spaced +48 cents apart, nearly a quarter-tone. This could be a subtle effect or a powerful compositional device depending on your application and intention. Note that the semitone width applies to the entire range of the keys selected in the Keymap preparation. To have a segment of the keyboard with a different semitone width, you will need to create two Keymaps with different ranges or pitches selected, and route them to different Tuning preparations.

Adjusting the semitone width while *also* using a non-Equal Temperament system has some wild results. For example, if you create a quarter-tone Piano using a semitone value of 50c, try setting the tuning to Just intonation. When you play C4 to Ab4, ordinarily a minor sixth, you’ll end up with a Just-tuned major third. Check out the Tuning and Temperament session from “Reinventing the Piano” for more information.

## Semitone Root

Under “semitone width” there is also a drop-down menu for “semitone root.” This sets a specific note to Equal Temperament as a starting point for a non-100-cent semitone scheme. By default, C4 will sound at ET as expected, with subsequent half-steps generated from there. But any pitch can serve as that initial reference, and can actually change the range of available notes on the keyboard.

For example, Set the semitone width to 50, creating a quarter-tone keyboard, and play some notes around middle C. Then change the root to D4. In the first case, C will actually sound a C, and in the second D will actually sound a D.

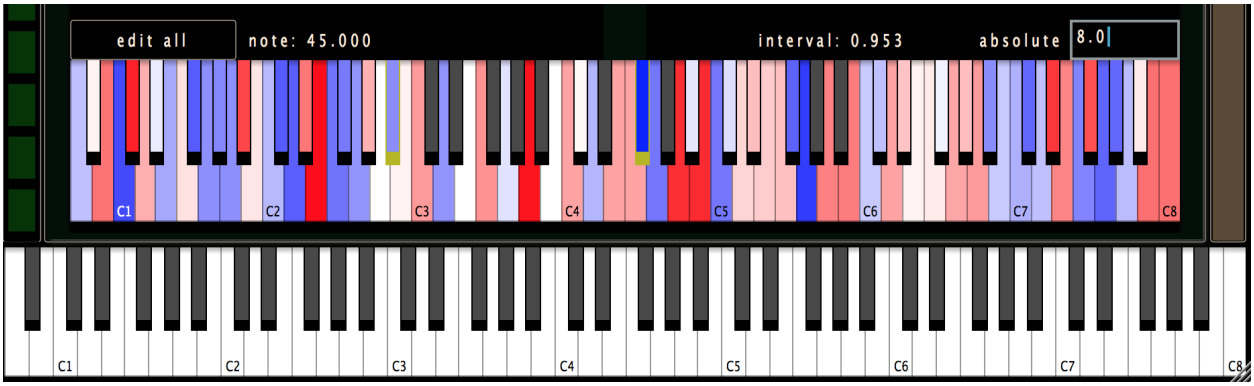
Another example: Let's say you want to perform with a violinist who tunes to A440, but you want to use a quarter-tone Piano. If you set the semitone width to 50, simply set the root to A4, so that A4 can be used to "tune" bitKlavier. Starting on A4, the violin and bitKlavier will both be at 440hz, then the bK keyboard will be in quarter-tones above and below the new root of A4.

Note that the root is only relevant if the semitone width is significantly larger or smaller than 100. At 101 or 99, you won't hear much of a change within the few surrounding octaves.

## Full Keyboard Tuning

Along the bottom of the Tuning preparation there is a full 88-key keyboard. As in the Keymap preparation, each key corresponds to exactly one pitch, so unlike editing on the Temperament keyboard, any edits made on the lower keyboard will apply to a single pitch. Think of this as absolute tuning versus circular tuning: Tuning by temperament is circular (repeated every octave), while tuning individual notes is absolute tuning and is not translated or repeated anywhere else on the keyboard.

Any key along the bottom keyboard can be individually tuned by click-dragging, by clicking at a certain location on the surface of the key, or by adjusting the pitch in the "absolute" box to the upper right of the keyboard. Here's the result of some random clicking:




Note that when you mouse over the lower keyboard, the note's relative offset will appear in the "absolute" box.

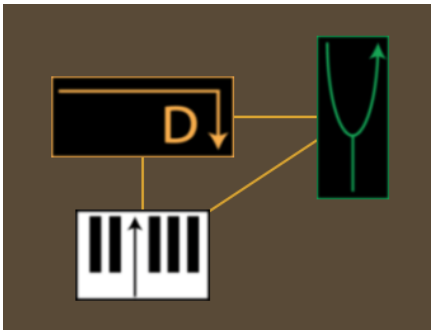
IMPORTANT: any changes made to the absolute tuning of the keyboard happen *in addition* to the existing temperament! Example: If every E is raised 14 cents in the temperament, and E4 is lowered by 8 cents on the absolute keyboard, then E4 will still be 6 cents sharp, while every other E will still be 14 cents sharp.

For more control, each key can be individually edited in a text field. Open the "Edit All" field and type **60:-13.7** in the empty space. Press return or hit "Edit All" again to return to the

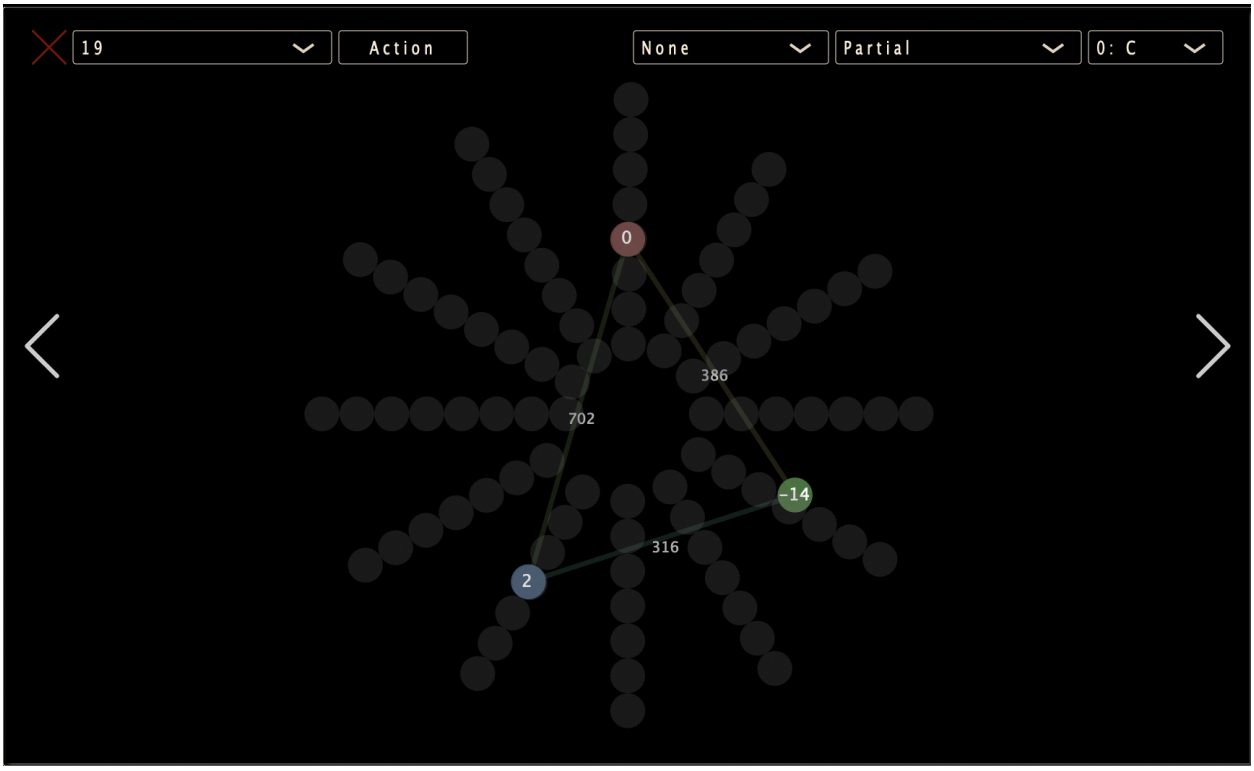
keyboard view; you will see that middle C (MIDI value 60) has been lowered -13.7 cents *relative to the temperament tuning.* This makes it easy to edit note values. These values can be copied into bitKlavier from other programs.

## The Spiral

The spiral—accessed via the left/right arrows on each side of the screen—provides a way to visual the tuning relationships of currently active notes. For this to work, in your Piano, Keymap must be connected to Tuning *as well as* Direct, so that Tuning knows what notes to display in real time.




The spiral begins inside with the lowest note on the piano and gradually circles outwards to the highest note. Active notes appear as circles near the equal-tempered references, with numbers indicating how far they diverge from ET, and lines between notes show the interval between them, in cents.




Loosely inspired by Erv Wilson's logarithmic spiral, the motivations for having the spiral will become evident later when we look at *spring tuning*, but even just as a way to quickly see how notes are tuned it is quite useful.

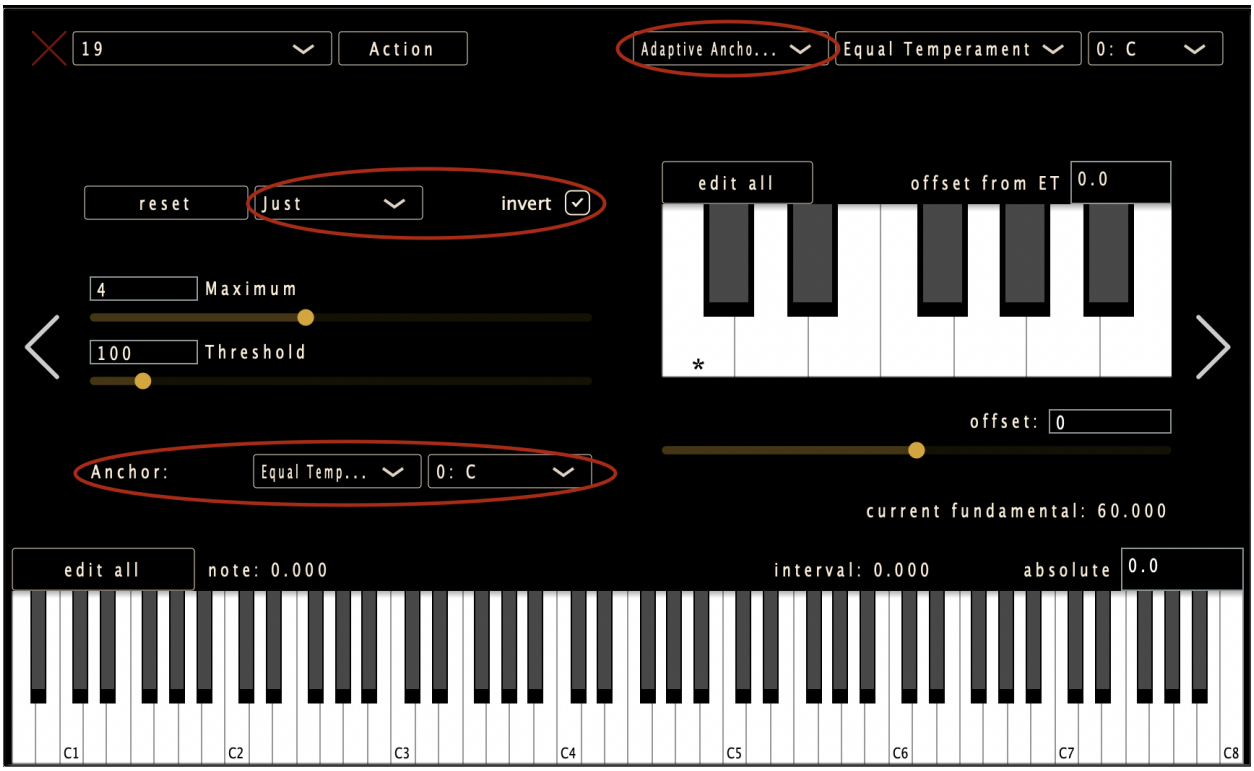
## Adaptive Tuning

Temperaments are fixed systems. We can pre-plan changes on the fly using Piano maps and Modifications, but conceptually, temperaments are inherently static. By contrast, *adaptive tunings* change in direct response to what you play. The tuning and temperament continuously regenerate based on your playing via a limited set of constraints. While these processes can be used to create "optimal" tunings–systems that minimize idiosyncratic intervals–based on your playing, they are also intended for exploration and play, and as such can yield unexpected results. bitKlavier currently has three basic flavors of adaptive tuning.

With both **Adaptive Tuning 1** and **Adaptive Anchored Tuning 1**, the fundamental (root) of the established adaptive scale (set with the menu to the left of the Tuning window) will shift automatically, either after a certain number of notes have been played (the Maximum slider parameter) or after a certain amount of time has passed (the Threshold slider parameter). Each note is immediately re-tuned based on a moving fundamental, and various parameters determine *when* the fundamental should move and *where* it should move. In your Piano, Keymap must be connected to Tuning *as well as* Direct, so that Tuning knows what notes to adjust in real time.

## Adaptive Anchored Tuning 1

In the Adaptive Anchored system, when either of the adaptive parameters are exceeded (Maximum or Threshold), the fundamental will change to the tuning determined by the “Anchor” scale, located in the bottom left of the Tuning window:




To begin, set Maximum to 1, meaning that the fundamental will reset with every note played. Set “Threshold” to 1000 (for reasons to be clear shortly), “Adaptive scale” to Just, and “Anchor scale” to Equal Tempered. Now, play an *ascending* scale. You should hear an Equal Tempered scale where the “current fundamental” changes to a new integer value with each note you play. You should also see that the “note” (to the bottom left about the big keyboard) is always an integer (Equal Tempered) value.

Now set Maximum to 2 and try playing a scale again. As long as each note is within 1000ms of the previous note, you should see the “current fundamental” change with every *other* note played, and that the “note” values of the in-between notes will have some fractional value; this is where the system starts to kick in. With these same settings, try this:

1. Play C4 then Bb5 (an ascending minor-7th) and listen to the tuning of the Bb
2. Play D4 then Bb5 (an ascending minor-6th); hear how the Bb is tuned differently!

In (1) C4 is the fundamental, so the Bb is tuned as a 7/4 minor-7th above C, which is about 31 cents flatter than an Equal Tempered minor-7th.
In (2) D4 is the fundamental, so the Bb is tuned as a 8/5 minor-6th above D, which is about 14 cents *sharp* to Equal Temperament, so it sounds nearly a quarter-tone sharp to the Bb in (1).

Next, set Maximum to 4 and play this pattern over and over again: C4-Bb5-D4-Bb5; in this case the Bb should always sound the same because the fundamental (60.0) does not change to D4 mid-pattern. Play with this and other settings while monitoring the "current fundamental" and "note" values to learn more about how it is working.

### Adaptive Tuning 1

Select Adaptive Tuning 1, and you'll notice the Anchor scale menus disappear. The next adapted note will now become the new "anchor," which will continuously change while playing. Set Maximum back to 1 and play an ascending scale; you should see that the "current fundamental" gets reset with every note played (as before) and that its value is always the same as the "note" tuning.

This is where things can get strange and fun! Play this pattern over and over again:

||: C4-D4-E4 :||

Why does the pitch drift up? Every note is being tuned with the *prior* note as the fundamental. We begin with two whole steps, each tuned as the *first* whole step in a Just scale (interval = 2.039, slightly larger than two ET half steps), followed by a descending major-third, also tuned as it would be in a Just scale (interval = -3.863, so slightly *smaller* than four Equal Tempered half-steps). The two just-tuned whole-steps yield a major third that is *larger* than the just tuned major-third, so the whole system begins to drift upwards.

Try this pattern, and hear how the pitch drifts down:

||: E4-D4-C4 :||

If the "invert" toggle is checked, the downward drift should move at the same pace as the upward drift from the prior example, because Tuning will consider both ascending and descending intervals as the same (a whole-step is a whole step, whether going up or down). But with "invert" unchecked, the second example will drift down more quickly than the first. This is because, in short, the downward Just-tuned whole step is treated like the minor-7th of the Just scale, which is a much bigger whole step than the upward whole step. For a temperament with some slight differences, try the Duodene scale. Feel free to refer to the Appendix at the end of this manual for how each pre-programmed bitKlavier interval is tuned.

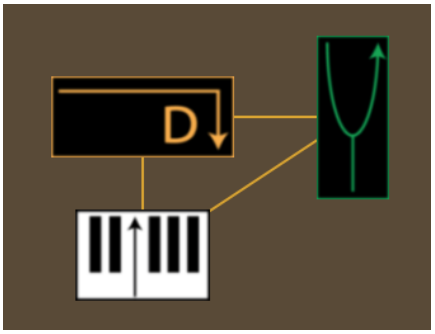
To further explore the Adaptive Tuning feature, load Example Gallery 17, “Tuning 2.” Instructions for how to begin using the Adaptive Tuning settings are written directly into the Gallery. Adjust the maximum number of pitches, time threshold, and adaptive temperament, each of which helps to determine how and when the Tuning preparation will shift based on Keymap input. Also, check out the *Mikroetude* called *Didymus,* which uses Resets to manually anchor the Adaptive Tuning.

Adaptive tuning is a wide-open field of study, and these two approaches (Adaptive & Adaptive Anchored) are meant to be more exploratory than “solutions” that always yield some kind of optimal or neutral tuning. While Adaptive Tuning systems may be used to create consistent optimal tunings, systems with their own idiosyncrasies create valuable compositional opportunities, and these Adaptive Tunings can be used to satisfy either goal. We are currently exploring a range of richer systems, including the spring tuning system described next.

### Spring Tuning

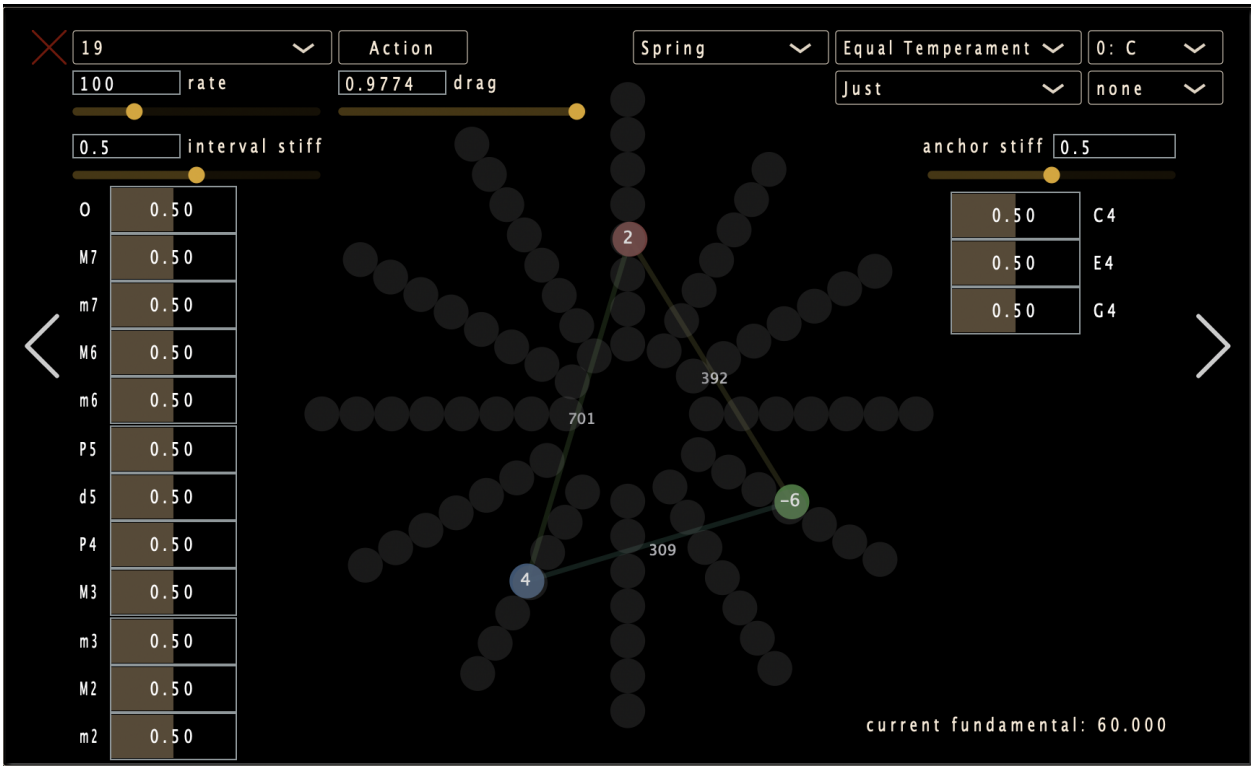
Some years ago, John de Laubenfels proposed an intriguing system for optimizing the tuning of sonorities. This system is detailed in William Sethares’ book *Tuning Timbre Spectrum Scale*, but was never to our knowledge fully implemented (though recently Stange, et al. created a similar system). Our implementation is largely inspired by de Laubenfels, though is not identical, and is also different in key ways from Stange’s system, discussion of which is beyond the scope of this manual.

As with the other adaptive systems, in your Piano, Keymap must be connected to Tuning *as well as* Direct, so that Tuning knows what notes to activate in the spring system.

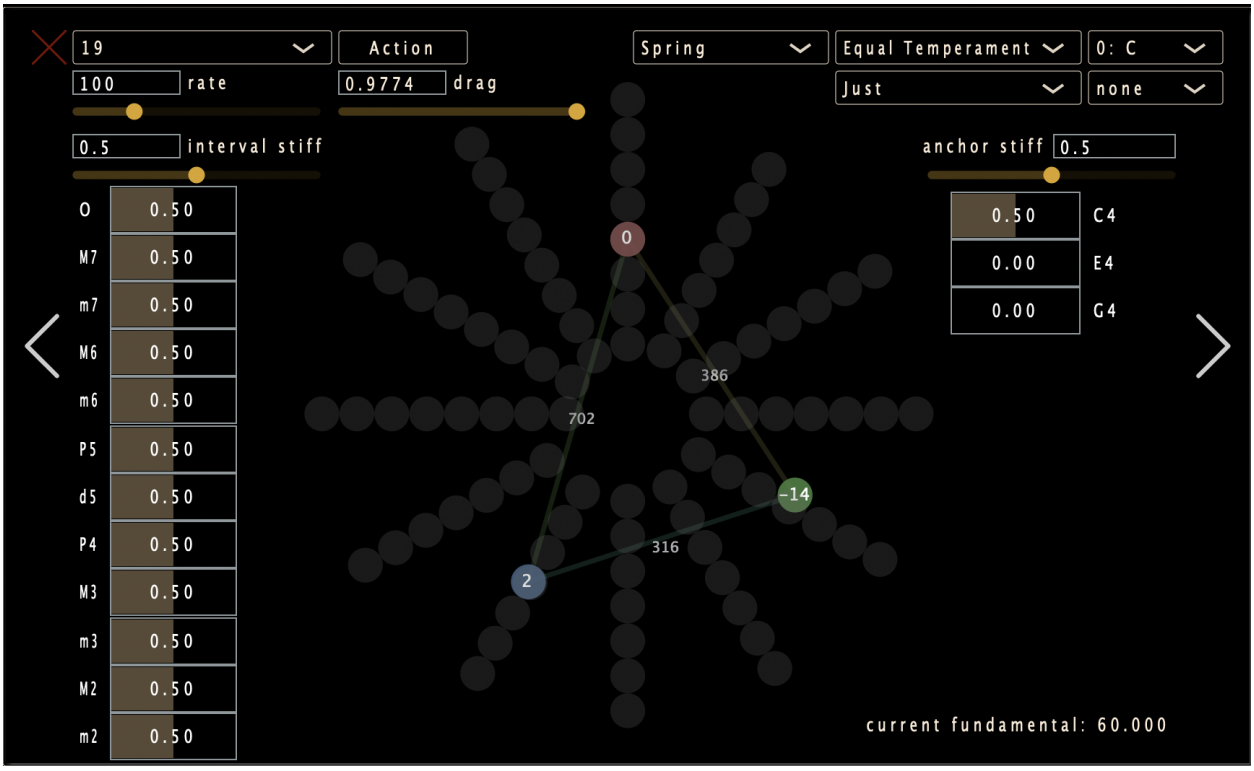



The key concept is that intervals are modeled by virtual springs with rest lengths determined by a chosen tuning system (usually just). So, a major third would typically have a rest spring length of 386 cents, a perfect fifth a rest length of 702 cents, and so on. When more than two notes are played, springs of the appropriate length are connected between them all, and they pull on one another until they reach a condition where each is minimally stretched or compressed, optimizing the tuning. The relative strengths of the various intervals can be set,

so, say, major third springs are stronger than, say, perfect fifth springs, and the system will respond accordingly.

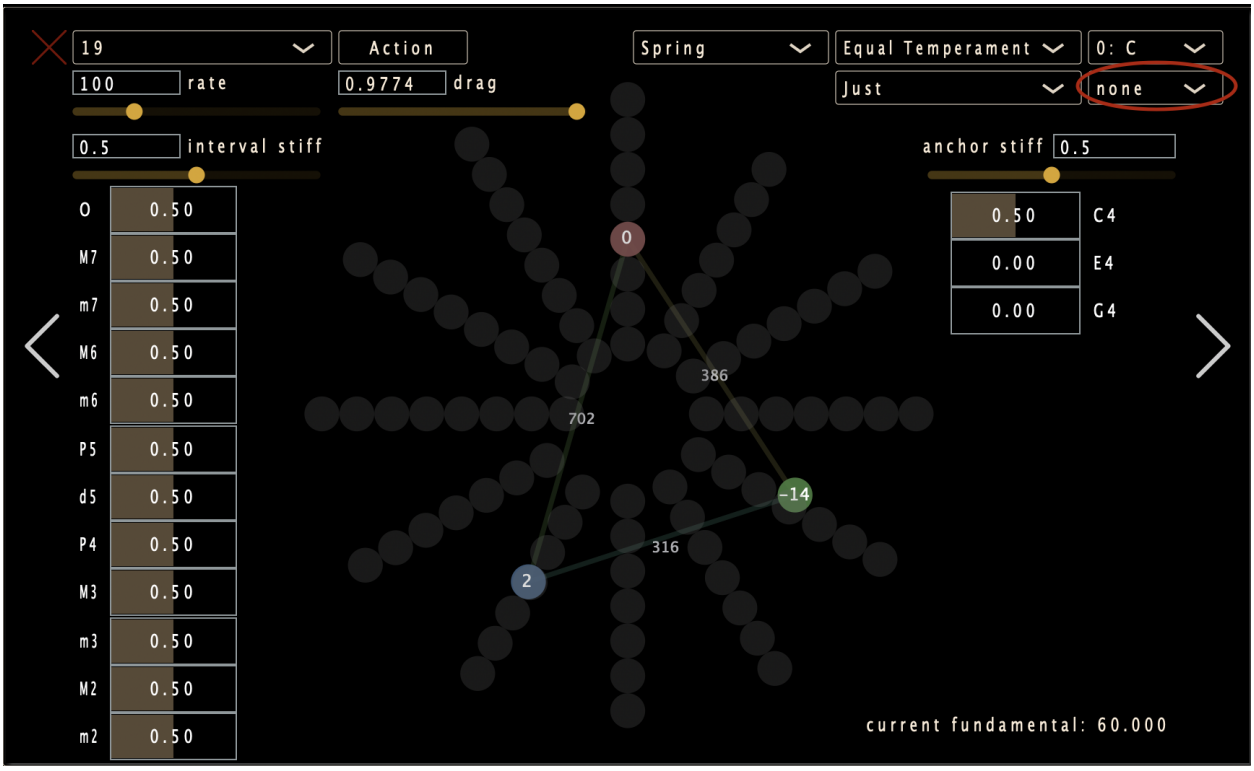



In addition, a background set of “anchor notes” (represented by the spiral) is connected to active notes via another set of springs, which help prevent the system from drifting. In the image above, the anchor scale is equal tempered (as set by the top-most menu) and the interval scale that sets the default lengths of the interval springs is just (as set by the menu just below the anchor scale menu). The relative strengths of the springs to the anchor notes can also be set, allowing some pitches to be favored and others to be totally unmoored, if desired; in the above image, all the interval and anchor springs are set to equal strengths of 0.5, slightly distorting the tuning of the intervals, while in the following image only the C anchor spring is “on,” so the triad is more perfectly tuned:




The overall *stiffness* of the interval and anchor springs affects the dynamics of the system, causing the springs to vibrate more or less quickly, and the *drag* models a kind of friction in the system (imagine the notes being dragged through molasses), damping the spring oscillations. Finally, the *rate* (in Hz) determines how often the spring model runs. All four of these parameters can be adjusted to create a wide range of different responses, from slowly oscillating systems to almost inaudible and instantaneous tunings.

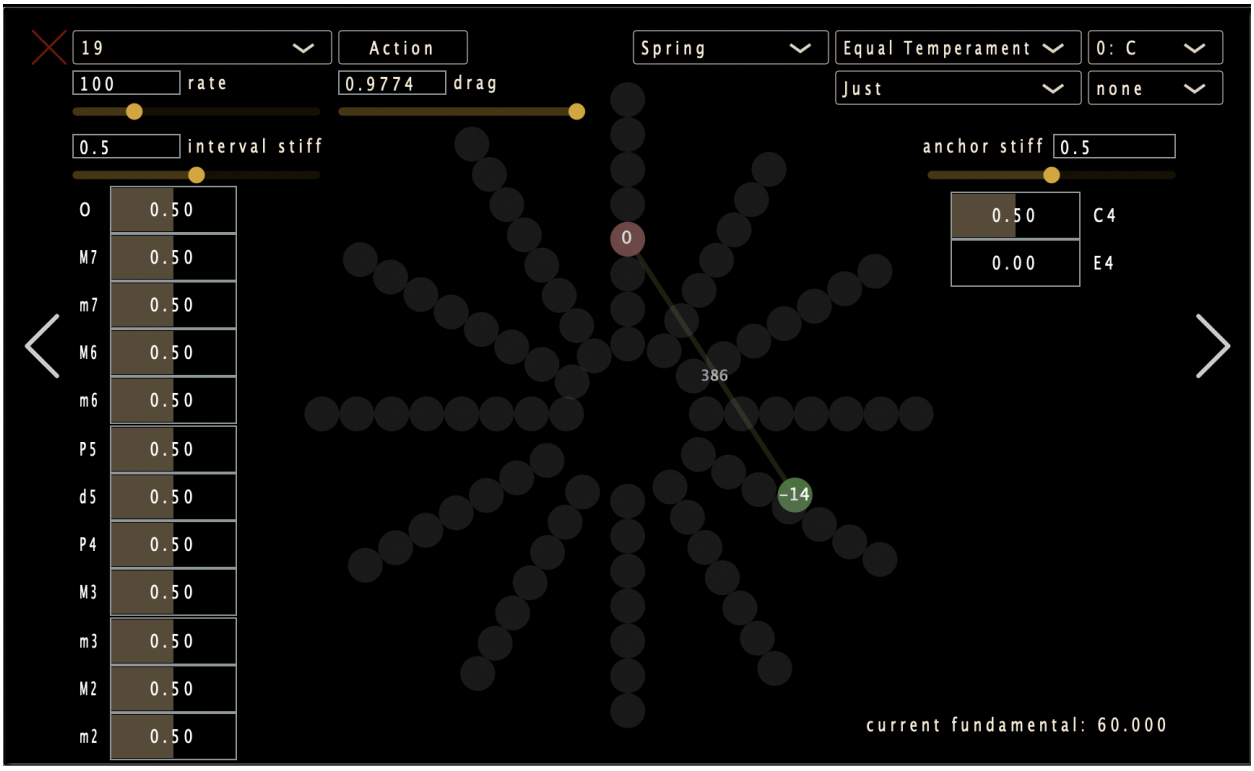
One of the more complex components of this system is the *interval scale fundamental:*



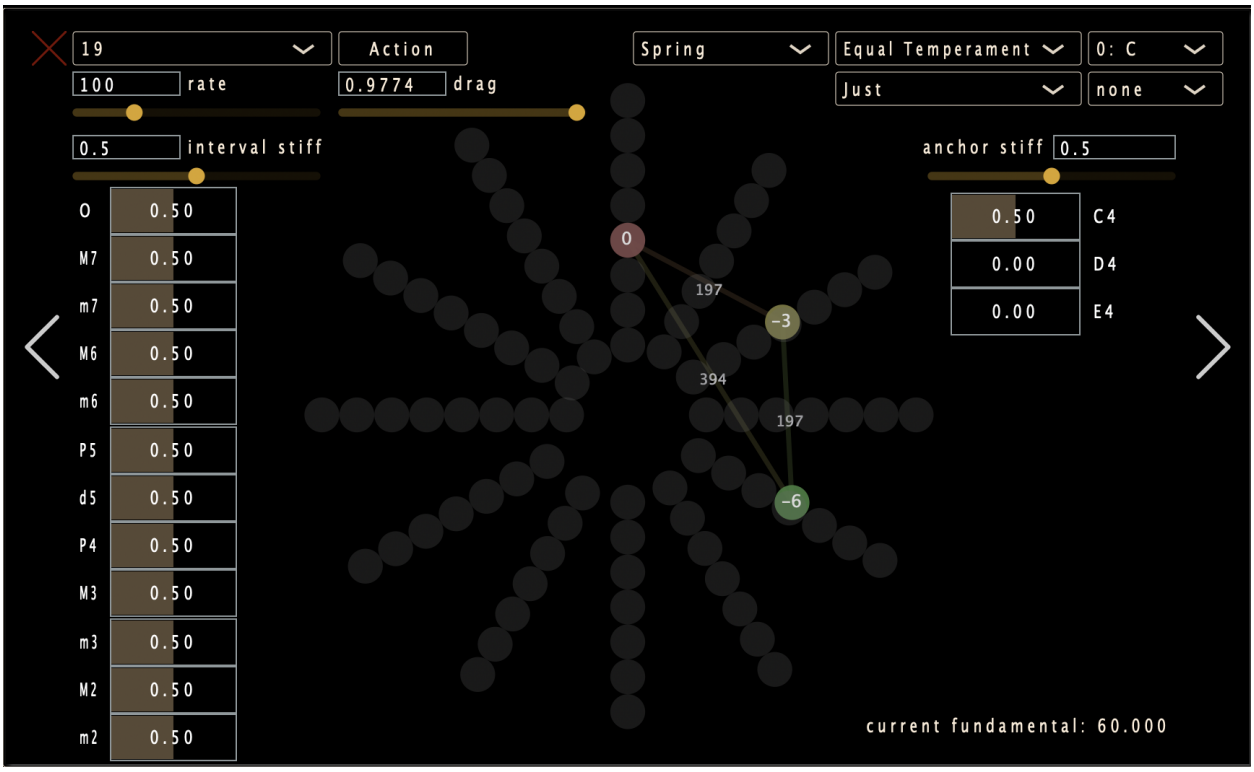

When set to “none,” the behavior is as described so far, where the resting length for a spring is set only with regards to the interval between its two pitches. So, for example, every M2nd spring will be set to 204c when using the just scale, which has a 9/8 M2nd between the root and M2nd:

| Interval | U | m2 | M2 | m3 | M3 | P4 | a4/d5 | P5 | m6 | M6 | m7 | M7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ratio | **1/1** | **16/15** | **9/8** | **6/5** | **5/4** | **4/3** | **7/5** | **3/2** | **8/5** | **5/3** | **7/4** | **15/8** |
| Length (cents) | **0** | **111** | **204** | **315** | **386** | **498** | **582** | **702** | **813** | **884** | **968** | **1088** |

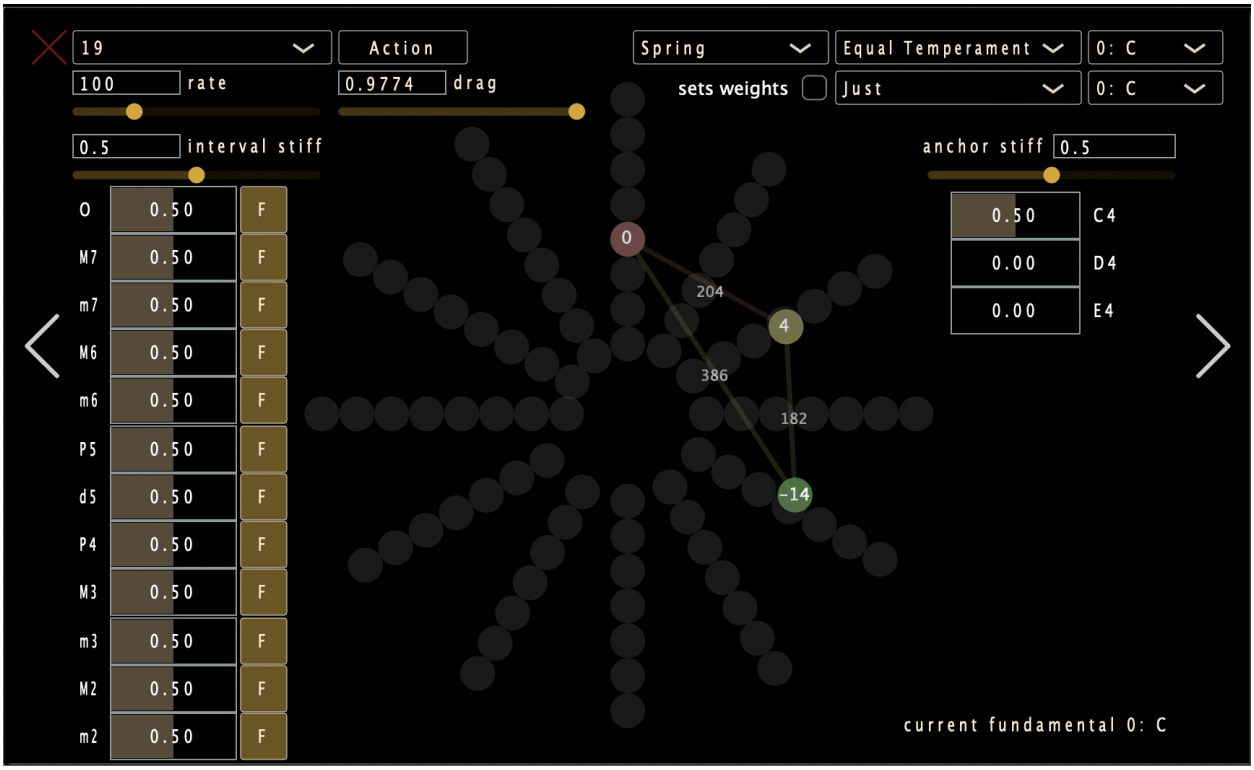
In this case, if you play a C-D-E trichord, the D will be squeezed by its springs on either side, and it will in turn push the C-E M3rd a bit wide. Here is the M3rd by itself:




And now with the D inserted between:



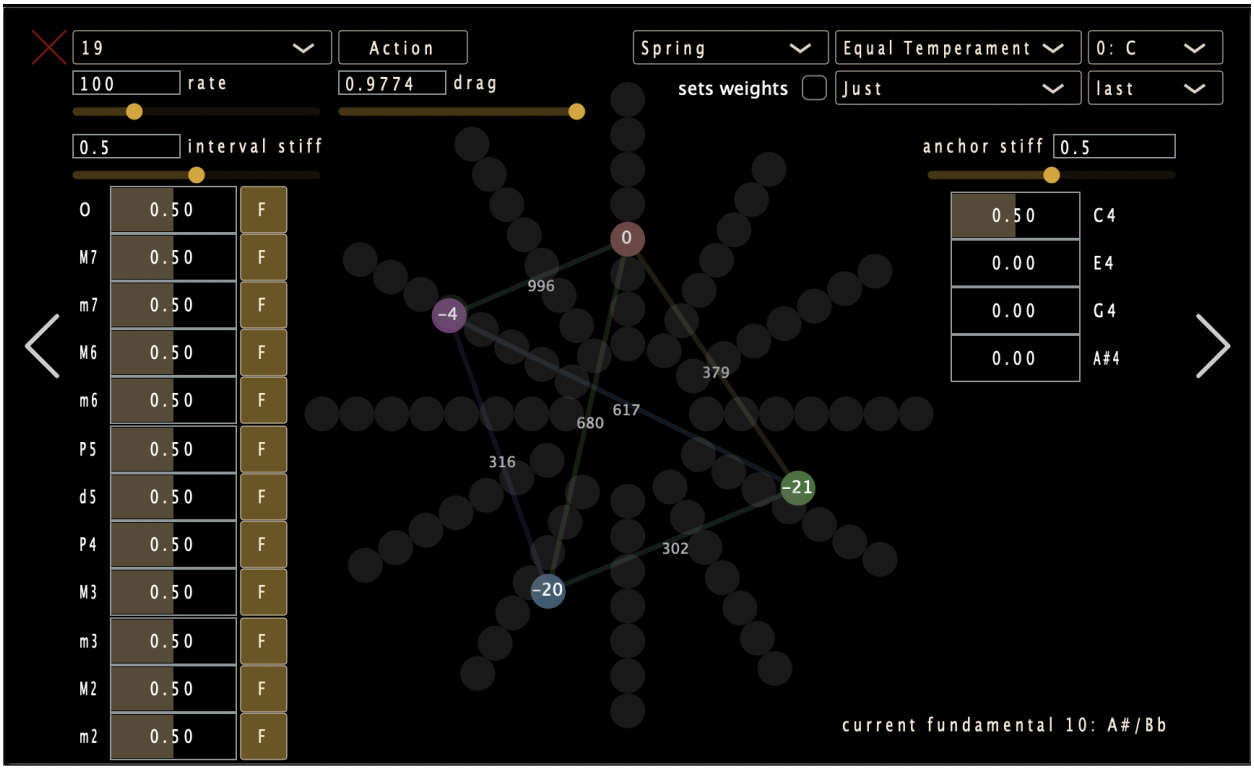
If we set the interval scale fundamental to C, however, the D-E M2nd will use the 10/9 ratio (386 - 204 = 182c, from the table above) of that just scale (which is the ratio between the major 2nd and 3rd in that scale; 9/8 * 10/9 = 5/4), so all the springs will naturally sit at their rest lengths:



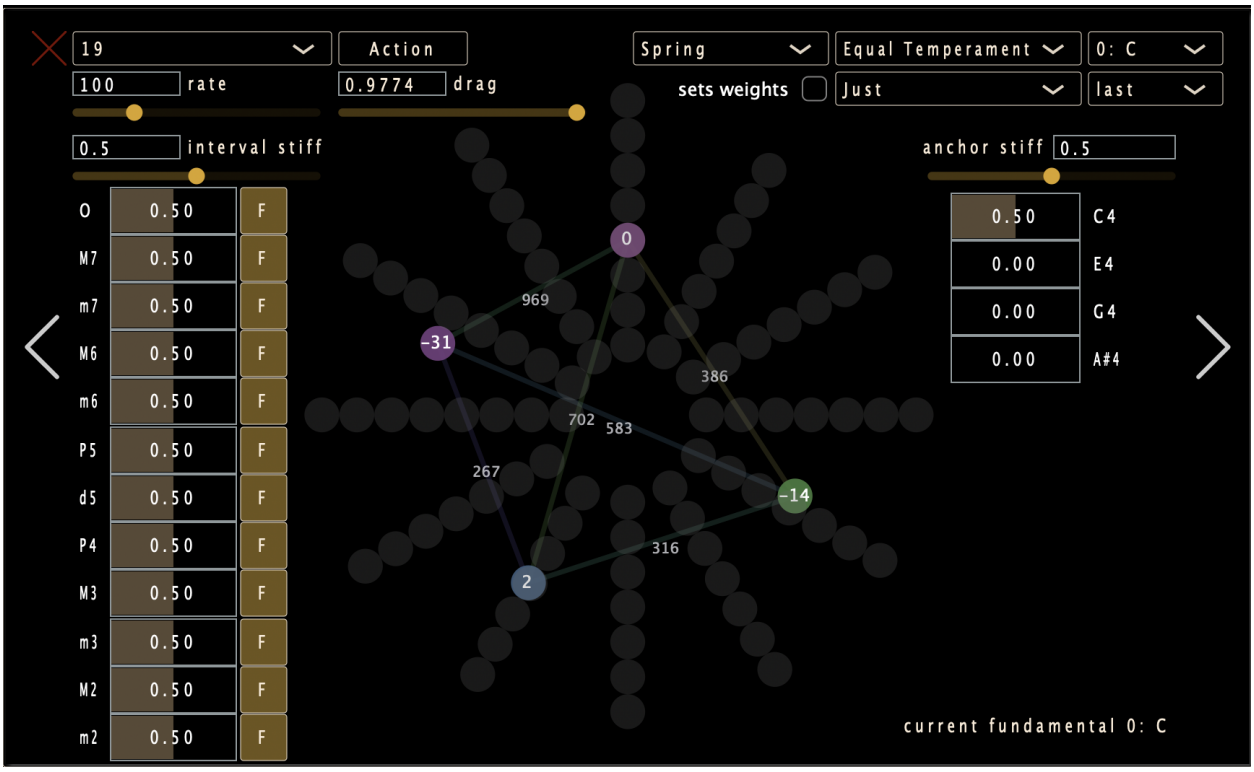

Setting the fundamental to “lowest” will use the lowest sounding note for the fundamental, and
“highest” the highest sounding note, while “last” will use the last played note; all can yield interesting behaviors with compositional possibilities.

Another example: try playing a C4 dominant 7th chord using “last” as the interval scale fundamental (and setting the anchor spring strengths to zero for the E4, G4 and Bb4), and compare the sound of the chord when arpeggiating it upwards vs downwards; can you figure out why these sound different?

Here’s what it should look like after playing the chord ascending:




And descending:



Notice the addition of the yellow F squares by each interval spring; these are toggles that allow you to decide whether to use the moving fundamental as a reference for setting the spring length, or whether to revert to the more “local” approach described earlier, as in “none” mode.

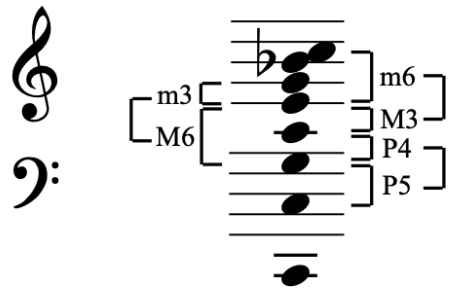
Why do this? bitKlavier’s movable fundamental allows us to retain some of the richness of the non-unique interval sizes (the M2, for instance) in a coherent fashion. But, one paradoxical consequence of this is that after notes have been added or subtracted from a sonority and the springs have stopped oscillating, they will all be at their rest lengths; this is essentially tautological, since using the spring length table this way guarantees that all spring lengths will be compatible, but not unique—we might have, for instance, two different spring lengths for the perfect 5th, one for the familiar 3:2 ratio, and another for a “wolf” interval (40:27 between D and A in Table 1, when the fundamental is C), or the two different M2nds as already described.

Why, then, use the springs at all? For one, they allow the system to gracefully adjust as springs are added and subtracted, which is essential in one of the moveable modes like *last.*

Another approach that the springs afford is the mixing and matching of spring lengths based on a fundamental with those determined locally, as in the core spring tuning (when the fundamental is set to *none*). Why might we want to do this? Consider again the case of the 40/27 5th between D-A, when the fundamental is C. Given the central role of the 5th in the overtone series, it would be reasonable to simply ask that all perfect fifths be 3/2, and similarly its complement, the perfect fourths, be 4/3. We might also want to avoid the Pythagorean major-3rds (81/64) the sometimes crop up, replacing them with 5/4 springs. To enable these sorts of combinations, bitKlavier (when in one of the “fundamental” modes) has a toggle next to each interval spring slider that determines whether the fundamental sets its spring length (F) or whether the length is set locally (L; so if all are set to L, then the system will behave identically to if it were in “none” mode). While this can be confusing, and in all likelihood most users will not make use of these options or even understand what they are, in the spirit of play and exploration we want to make this sort of configurability possible; it’s important to remember that we can’t anticipate all the musical situations that might arise in this world of dynamic tuning, and so while we are at risk of having too many features, we don’t want to overly prescribe what we think might be worthwhile.

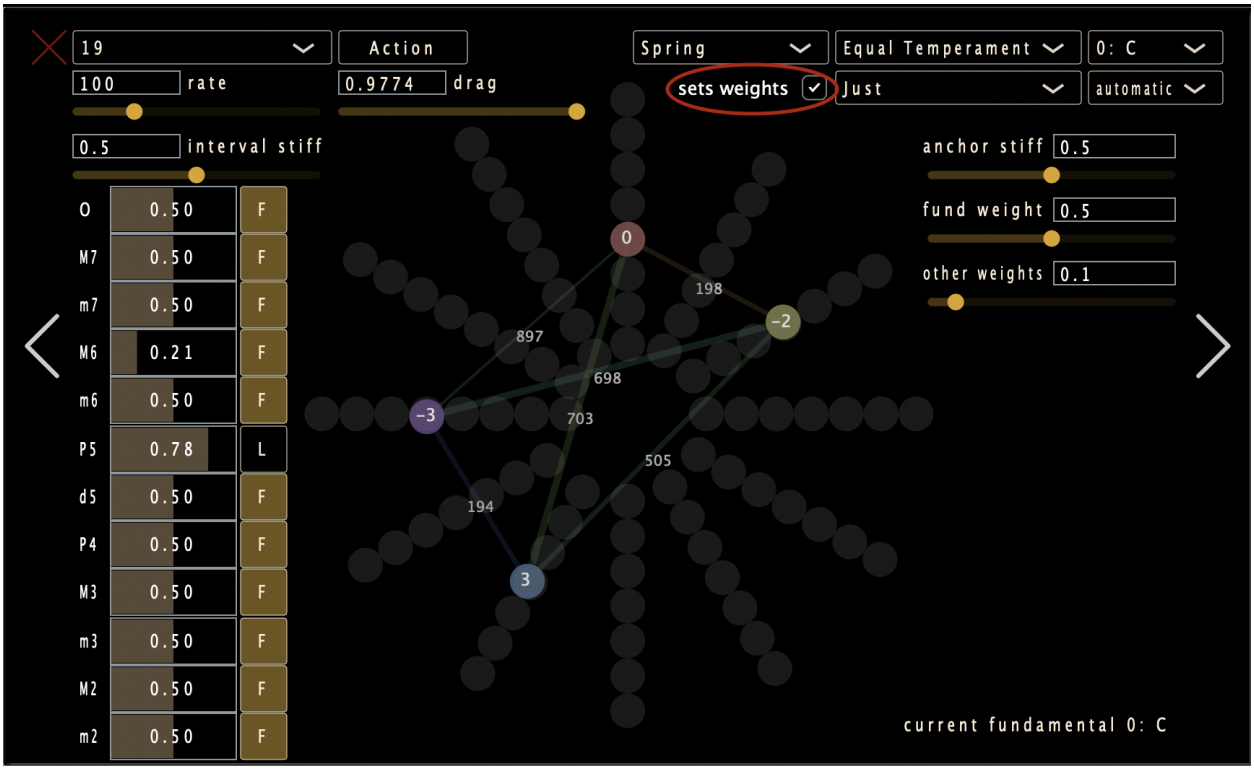
The *highest, lowest,* and *last* modes beg the question: can’t bitKlavier just figure out what an ideal fundamental would be to tune whatever sonority is play “as just as possible?” Of course, that is a qualitative question, but for the examples considered so far (a stack of two whole-steps, the dominant 7th chord) it’s a reasonable one. In *automatic* fundamental mode (new as of v2.4), bitKlavier looks at the sounding notes and tries to determine whether they suggest a particular overtone series, inspired in part by the psychoacoustic phenomenon of the “missing fundamental.” Below, we see the overtone series based on C, and then its first few intervals, in complementary octave pairings, so the P5/P4, the M3/m6, and the m3/M6.

The *automatic* algorithm works down from the top of whatever chord is sounding looking for these intervals; if it finds any of these intervals, it returns a fundamental of C. It prioritizes 5ths and 4ths over the M3/m6, and similarly the M3/m6 over the m3/M6, in the event that different intervals suggest different fundamentals, and by working from the top down, it prioritizes lower register voices; imagine a stack of perfect fifths, and the bottom fifth would determine what the fundamental is. If none of these intervals are found, then it leaves the fundamental unchanged.




This simple algorithm is inexpensive to calculate, even when many notes are sounding, and it yields predictable and intuitive results; the C-D-E trichord yields a C fundamental, as does a C dominant 7th chord, and so on.

One final component of the the “fundamental” mode in spring tuning is that the fundamental can also be used to determine the anchor spring weights; this is particularly useful in the mobile fundamental modes like *last* and *automatic*, enabling the anchor springs to track the changing fundamentals. Below, we see this and both the automatic fundamental finding and the local/fundamental setting for the interval springs, on a stack of perfect fifths, C-G-D-A.




Here, since C-G is the lowest fifth, the system determines that C is the fundamental, which will then have an anchor weight of 0.5, and all the other sounding pitches a weight of 0.1. Then, with the P5 set to “local” (L), the D-A P5 spring is set to the typical 702c, rather than the 680c that the “fundamental” (F) mode would define. This creates a tension between the C-A M6 spring and the D-A P5 spring, which is weighted to favor the P5, so the fifths in this sonority will be close to 3:2. Remember that this is a dynamic system, though, so the M6 won’t always be compromised in this way; for instance, if we simply release the D, the D-A M6 will shrink down closer to its normal 5:3 884c length.

We will be writing more about these design choices and tuning more generally in an upcoming paper about bitKlavier. We recommend exploring the example galleries for spring tuning to get a sense of how it might be used.

## Blendrónic

This preparation was introduced in bitKlavier v2.5 and is the first preparation that processes the sound coming from other preparations. In short, it is like a delay pedal that can be prepared to change its delay time and behavior according to a set of patterns, not so unlike Synchronic.


The primary parameters are as follows:

**Beat Lengths** - nearly identical to the Beat Length Multiplier in Synchronic, this parameter adjusts the actual length of each beat, in multiples of the pulse as set by Tempo, but unlike Synchronic, this does not then trigger a sound, but instead triggers a change in delay length.

**Delay Lengths** - sets the delay length, in multiples of the pulse (as set by Tempo).

**Smoothing** - sets how long it should take for the delay to change; 0 will result in sudden clicks on the beats, whereas non-zero values will result in chirps or scrubbing sounds, depending on the value and how big a change in delay is required.

**Feedback Coefficients** - sets how much of the output of Blendrónic should be fed back into itself.

> *Note the small circle-arrow at the bottom left of each pattern slider; pressing this will rotate the values, which can be handy when trying to establish particular relationships between the different parameters (new as of v2.5).*

## Beat Lengths

These values are multiples of the pulse, as set by Tempo, and they define how much time should pass before the delay length changes. In the picture above, these values are 4:3:2:3,

so if the pulse is 250ms, the beats will occur every 1000, 750, 500, and 750 ms. When one of these beat times is reached, the delay will change its duration, according to the pattern set in Delay Lengths.

## Delay Lengths

These values are also multiples of the pulse, as set by Tempo. In the picture above, they are 4:3:2:3, so if the pulse is 250ms, this will result in a sequence of delay changes of 1000, 750, 500, and 750 ms. In this particular example, the delay lengths and beat lengths are the same, so the delay times will always match the beat times.

## Smoothing

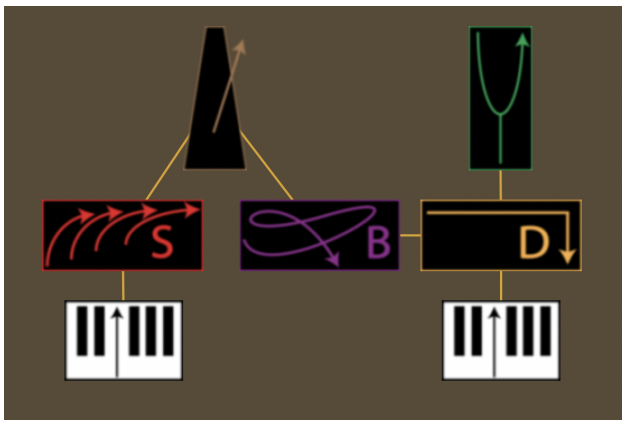
When the delay lengths change, the change can be instantaneous, or it can be smoothed over a set period of time. In the example above, it will take 50ms to change the delay length at each beat, resulting in a high pitched chirp.

## Feedback Coefficients

This determines how much of the output of Blendrónic is fed back into itself; a value of 1. Will allow Blendrónic to continue indefinitely, while smaller values will cause it to fade. A single value of 0 in the midst of much higher values can have the effect of punching holes in the delayed sound.

## Blendrónic Connections and Targets

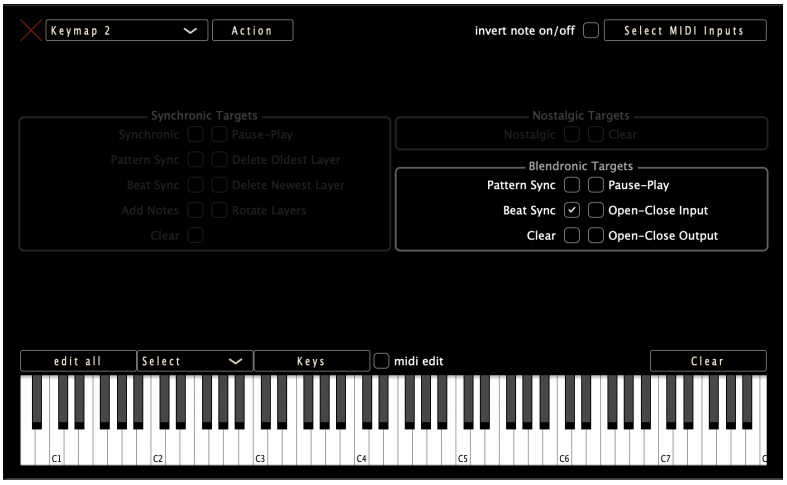
Blendrónic will only process the output of preparations to which it is attached; so, in the following piano, the Direct will be processed, but not the Synchronic.




Note also that Blendrónic does not need a Keymap; it will happily process away whatever it is attached to. However, if you want to synchronize its behavior, akin to Synchronic, or clear its contents, it will need an attached Keymap that targets the desired action.

As of v2.8, all the other sound-making preparations have a “blendrónic send volume” slider; this controls the gain of the signal being sent to Blendrónic, making it possible to balance the relative volumes of various preparations feeding a single Blendrónic.

As with Synchronic, it is possible to trigger particular events within Blendrónic. In the example below, an attached Keymap has Blendrónic Beat Sync activated:




and then in the Blendrónic Target Triggers window (reached by pressing the left/right arrows <> on the left/right sides of the Blendrónic preparation window), it is set to respond to noteOn messages, resetting the timing of the beat changes.

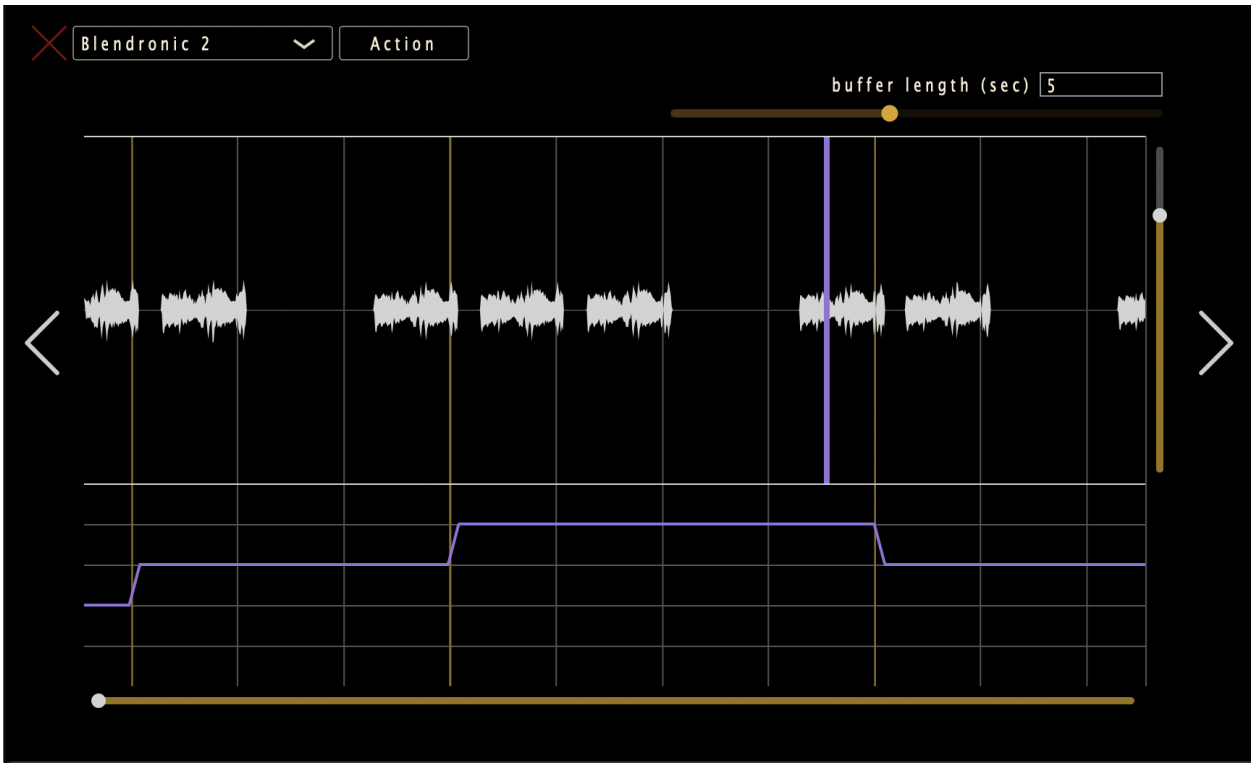

These targets behave as follows:

**Pattern Sync** - restarts the patterns (Beat Lengths, Delay Lengths, etc…) on the next beat; the pulse itself is NOT reset.
**Beat Sync** - the pulse is reset, but the patterns are not.
**Clear** - clear the delays.
**Pause-Play** - pause playback and without changing the contents of the delay-line; then play will restart it, as it was before pause.
**Open-Close Input** - stops and starts “recording” into the delay line; it will still playback, it just won’t add anything more to the delay..

**Open-Close Output** - when closed, will continue to “record” into the delay line, but nothing will be played back until it is opened again.

All of these are best understood by playing with them!

Blendrónic can be confusing! We have one additional window that can help elucidate what is happening:




It’s hard to describe, but time moves from right-to-left, with “now” being the right-most part of the window; we can see new sounds entering the delay there, and old sounds being fed back into it. We can also track the delay time changes, shown in the lower part of the window. The vertical purple line is the output of the delay, showing what is currently being played back; it moves, relative to “now,” as the delay lines change. Again, this is all best understood playing with it!

There is one additional slider, the “buffer length” slider, which sets the maximum delay that Blendrónic can handle, and it also affects how much time is represented in the window above (the width of the window shows the full delay time).

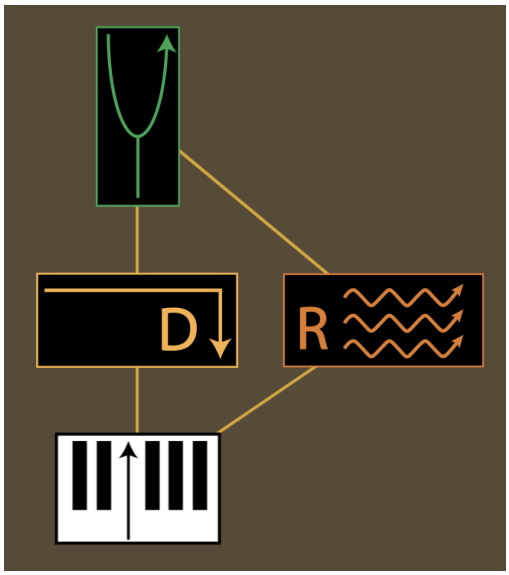
• • •

Like the rest of the preparations in bitKlavier, Blendrónic was originally developed as part of Dan Trueman's *neither Anvil nor Pulley,* and featured in the "Feedback" movement. It was later used in pieces like *Clapping Machine Music Variations* and others, but it has been expanded and made more flexible in its current instantiation. A short paper about the algorithm can be found here.

## Resonance

The acoustic piano is an instrument with many complex resonances: hold one key down silently, and it's string will ring in sympathy with other notes that are played, mostly due to overlapping overtones between all of the sounding strings. Introduced in v3.0, the Resonance preparation models that sympathetic resonance in a flexible way.

To hear this preparation in a way that most closely matches what happens in an acoustic piano (albeit *louder* than it is in an acoustic piano), simply create a piano as follows and play:



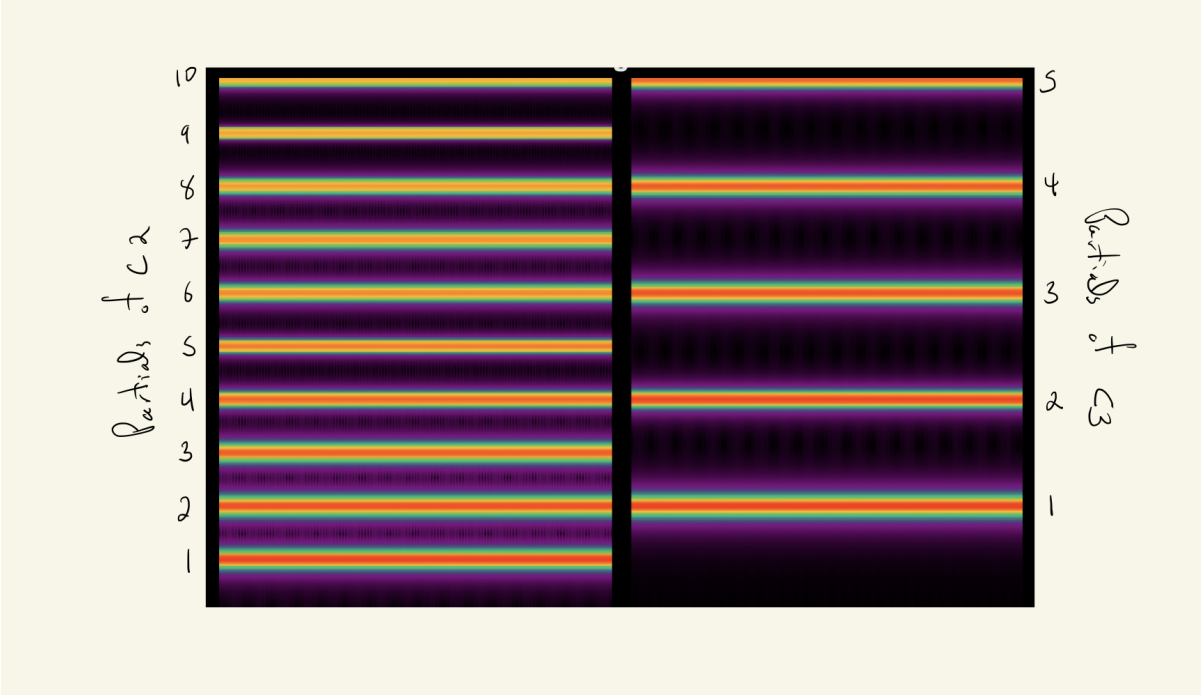

Try holding down a low C2 and then striking and quickly releasing its overtones one after another (C3, G3, C4, E4, G4, Bb4, C5); all of those pitches ring out, though some will be tuned differently than the struck note, because the overtones are not equal-tempered (the Bb4, for instance, will ring noticeably flatter than the struck note).

Then, invert that process: quietly hold down the overtone pitches as above, and strike/quickly release C2. Note how the held keys ring, because the virtual dampers are not

dampening those strings, so the overtones of C2 cause them to resonate (and note that in this case the ringing notes ARE equal tempered).

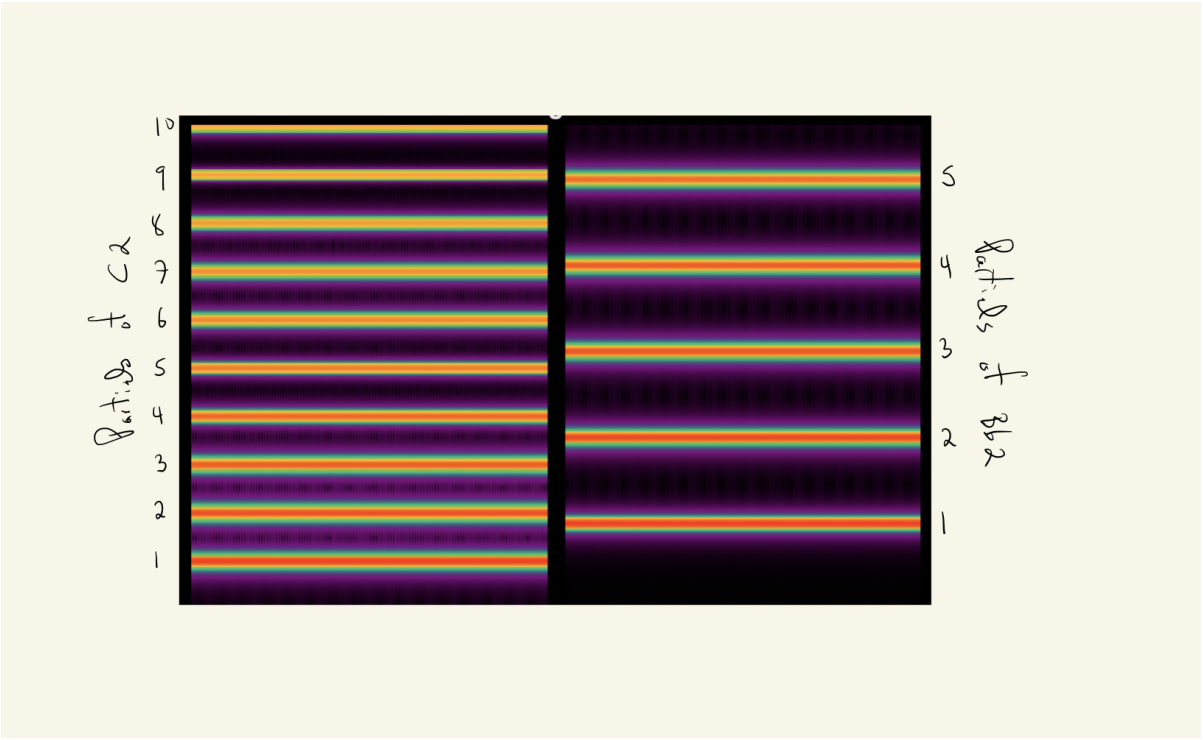
Those two examples illustrate the core functionality of Resonance; everything else is tailoring its behavior, perhaps to make it sound more realistic (add, for instance, the 11th partial, which doesn't come by default in the above), or *less* realistic.

To illustrate further, consider the overtones of two pitches an octave apart, say C2 (left) and C3 (right):




The main point is that these overtones (partials) overlap and will interact whenever the dampers for those strings are released. If the C3 key is held down and the C2 key is struck and quickly released, all 5 of those C3 partials will ring sympathetically. Conversely, if the C2 key is held down and the C3 is struck and quickly released, only the partials numbered 2, 4, 6, 8, and 10 will ring sympathetically, because those are the ones that overlap.

Another example, this time with C2 and Bb2:




Here, partial 7 of C2 overlaps a bit with partial 4 of Bb2, so they will interact, though not as strongly as the first case with C2 and C3, where the partials lined up exactly. To hear this most clearly, hold down the C2 key and quickly strike the Bb4 key. Partial 9 of C2 lines up quite well with partial 5 of Bb2, so they will interact as well (though in the bitKlavier default, the 9th partial is not present so will NOT ring; it is easy to add though!).

Note that the C2 and Bb2 strings are presumed equal tempered here, which is why the 7th partial of C2 is flat to the 4th partial of Bb2, since the 7th partial is quite flat to equal temperament.

Here is an overview of the parameters you find when you double-click on the Resonance preparation, first in panel 1:

**Volume** - controls the output volume of the full preparation.

**Blendrónic Gain** - controls how much of the Resonance is set to any attached Blendrónic preparations

**Start Time** - controls the range of start times within each sample to create the resonance; depending on the velocity a note is played with, the resonance will either start closer to the lower value (higher velocity) or the higher value (lower velocity; see below for a complete explanation)

**Max Sympathetic Strings** - constrains how many sympathetic strings can be ringing at a time; this is primarily to control how taxing this preparation is for the computer

**ADSR** - Attack/Decay/Sustain/Release envelope settings for the resonant notes.

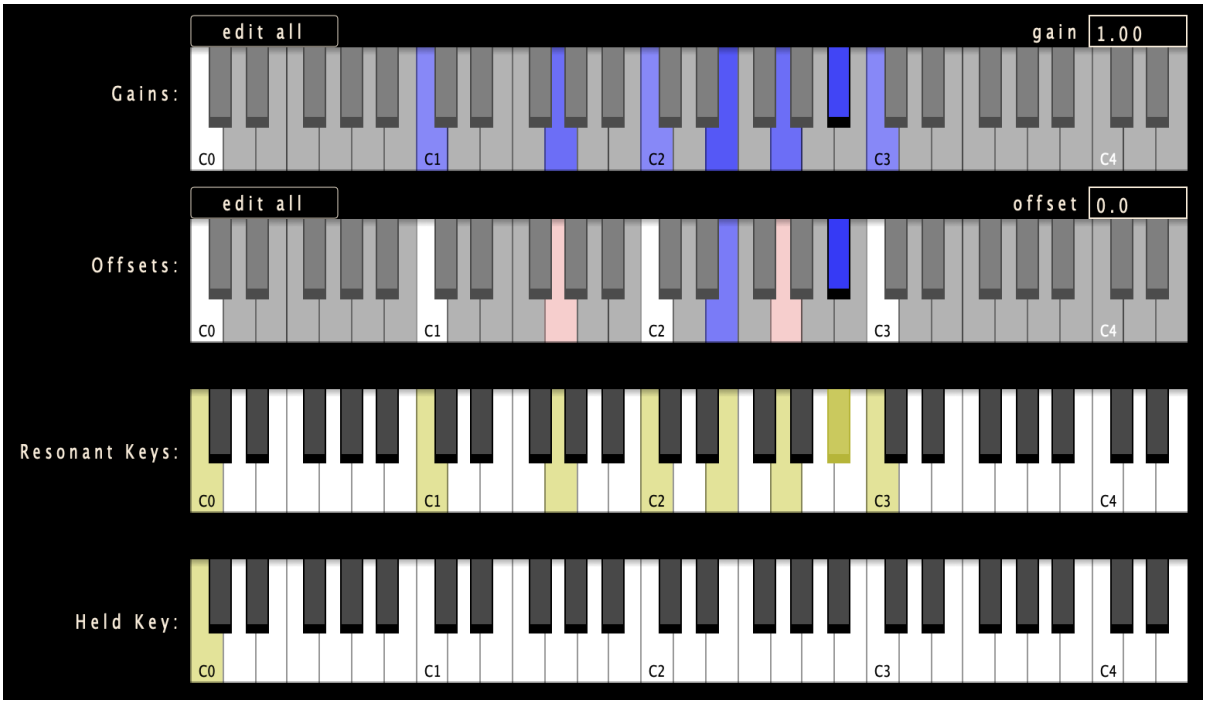
Then in panel 2:

**Held Key** - only a single key can be selected here, and all of the “resonant keys” are relative to this held key, the “fundamental.”
**Resonant Keys** - indicates which keys will cause the “held key” (its string) to ring sympathetically when struck; these are essentially the “overtones” to the “fundamental.”
**Offsets** - indicates any tuning offset (from equal temperament) in the resonant keys.
**Gains** - indicates relative gains/loudnesses of the corresponding resonant key

## Held Key and Resonant Keys

The “held key” is the key we use to set the fundamental for the partial/overtone structure we want. In the default setup, it is set to C0 (imagine we are holding down the key for  a low low C, below the bottom of the familiar 88-key keyboard). The “resonant keys” are the ones that, when struck, will be closest to the overtones of the “held key,” causing it to ring. In the default setup, these are set to the first 8 partials of C0. Of course, that overtone structure is not limited to C; all “held keys” will have a similar, transposed set of “resonant keys.”

This arrangement allows for the creation of all sorts of “overtone” structures, including “undertones” and totally unnatural “inharmonic” structures. Try, for instance, setting the “held key” to anything other C0 and leaving everything else the same, and then try playing various combinations of held and struck keys and listen to the resonances.



## Offsets

These values (in cents) set how far the tuning of the “resonant key” string is from the tuning of the actual overtone of the “held key” string. As in the example above, striking the equal tempered Bb4 while holding down C2 will result in two quite different Bb’s; the 7th partial of the “held key” C2 is 31 cents flat to the equal tempered Bb4 “resonant key.” Mousing over that Bb (Bb2 in the Offsets keyboard) will reveal the -31c offset; the “keys” on this graphical keyboard are sliders that you can drag on to change the offset, and you can also “edit all” to change by typing.

## Gains

The various partials of a vibrating string have varying strengths, some naturally louder or softer than others. The Gains keyboard enables building that into the resonant structure of Resonance

## Start Time and ADSR

Internally, Resonance creates the sound of resonance by playing the tails of the piano samples in the current sample set. While of course this isn’t what happens in an acoustic piano, bitKlavier isn’t an acoustic piano, and this approach is flexible and uses already accessible resources (the samples!). If there is significant overlap between partials, the resonance will be louder than if there is less overlap (as in the 7th partial example above). Of course the resonance will also be louder if the struck note is struck with greater velocity. The start time for the playback of the tail of the sample roughly models this:

- High velocity notes with lots of overlap will cause the playback to start near the lower range of the “start time” parameter (so, closer to the beginning of the sample, so louder, with longer time to continue ringing).
- Lower velocity notes (or notes with less overlap in their partials) will cause the playback to start near the higher range of the “start time” parameter (so, closer to the end of the sample, so quieter, with less time to continue ringing).

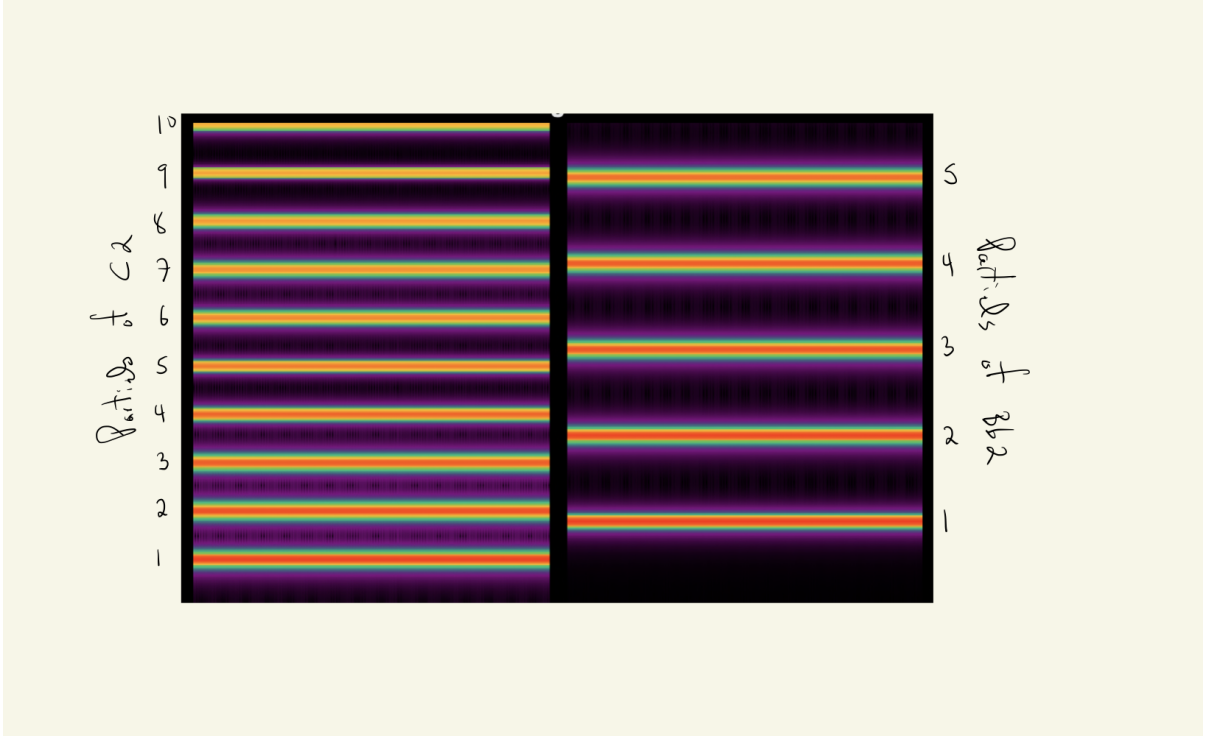
The ADSR values set the envelope for the playback of this tail. “Start Time” then is another way to think about the intensity of the resonance that you want to create; if the lower value is very low (even 0), then the resonance will sound almost like (or totally like) a struck note. If it is quite high, then the resonance will be soft, and also mellower, as the higher frequencies will have faded later in the tail. ADSR can also have a major impact on the quality of the resonance, making it “bloom” more slowly, fade more quickly, and so on.

## Max Sympathetic Strings

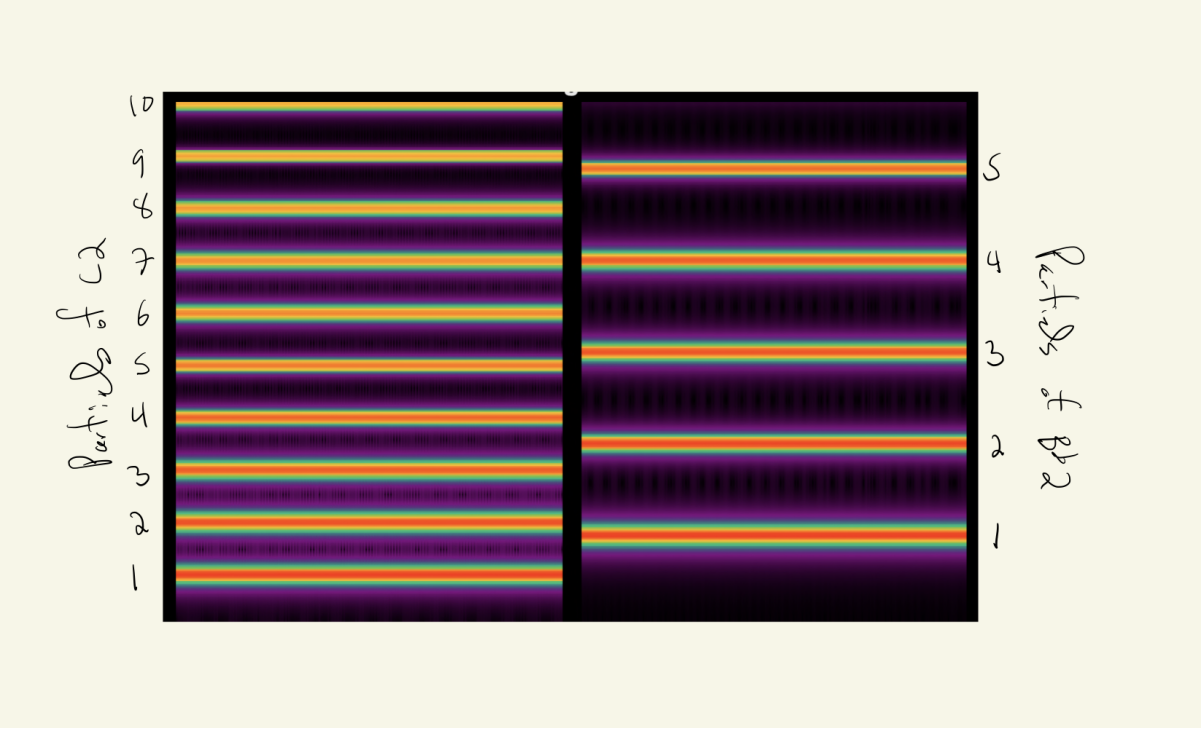
The more notes you play, and the more “resonant keys” there are activated in the Resonance preparation, the more taxing it is for the computer to play; it can explode pretty quickly actually, so this parameter helps constrain that, allowing more partials to be enabled in the resonant keys while keeping a lid on how many can actually be sounding at once.

## Attached Tuning

If a Tuning preparation is attached to Resonance, that will set how the fundamentals (held keys) are tuned. So, for instance, in the example above considering C2 and Bb2:




both the C2 and Bb2 are equal tempered, so the 7th partial of C2 is noticeably lower than the 4th partial of Bb2. If, however, the Tuning preparation is set to “C partial,” then the Bb2 will be tuned like the 7th partial of C:




and now the 7th partial of C2 will line up precisely with the 4th partial of Bb2. This can get pretty complex quickly, especially if, say, the Direct and Resonance preparations are using different Tunings.

As should be clear, the aim here is to allow for a range of explorations of sympathetic resonance; it should be possible to make something that sounds quite “natural,” which could be of use when making a recording of bitKlavier, or just to give the instrument a kind of warm glow; but it is also possible to create a range of other kinds of sympathetic structures, perhaps modeled after the spectral qualities of different kinds of instruments (gamelan, for instance), undertones, or…. who knows. Overall, this is a keyboard-centric approach to thinking about sympathetic resonance, where depressed keys allow for a range of resonant pitches that can be excited by other keys.

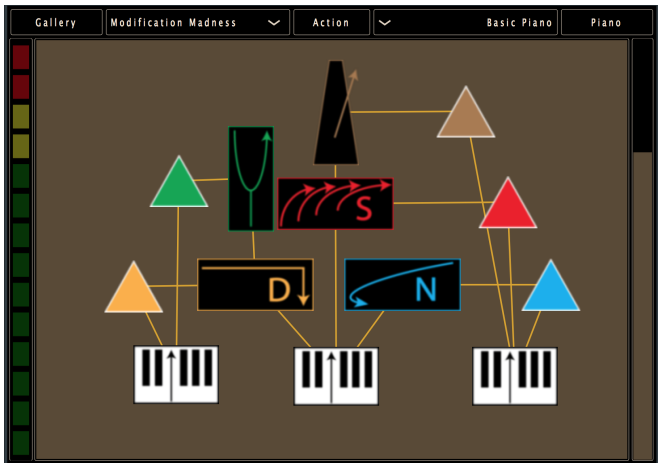
## Modification

We can make temporary changes to preparations based on Keymap input. Creating a Modification generates a neutral gray triangle, which can be connected to any of five preparations (Tuning, Tempo, Synchronic, Nostalgic, Direct). The color of the Modification will change to reflect the preparation it is modifying.

**Simple Modifications**

In the Basic Piano, create a Modification and connect it to the Direct preparation (it will turn orange). Double-click the Modification and you will see that it is essentially a Direct preparation with all of the parameters grayed out. If you make a change to any parameter, it will become highlighted indicated the modification you've made. This modification will apply its new settings to the attached preparation only when it receives an initial Keymap signal.

As with all bitKlavier preparations, you can save Modification presets within the preparation interface. Examples using Modifications include "Tuning 1" as well as "Undertow," "Marbles," "Wallumrod," and "Systerslaat" in *Nostalgic Synchronic.*




As of v2.8, Modifications have some new capabilities:

**Smoothing**
We can now have many modifications take place over a set period of time. In this example, the modification will set the gain of Direct to -10dB:


But, clicking on the button with the three dots at the right end of the slider will reveal this pop-up:

Where we can see that the “mod time” has been set to 2000 ms, so the gain will change smoothly over 2 seconds.

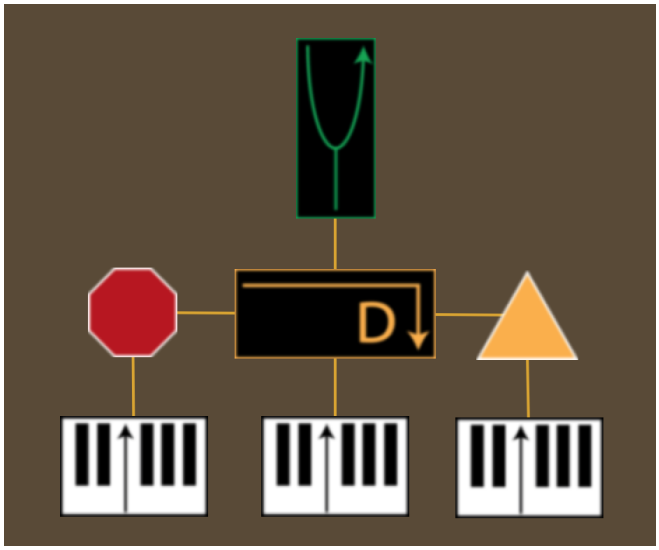
**Incrementing**
In this case, a parameter can be incremented by a set value, after first being modified. So in the example above with the gain slider set to -10, we can set the pop-up as follows:


And when we trigger this Modification, the gain will first be set to -10dB, and then every subsequent press will increment the gain by 1dB. If you want to limit how many times that can happen, set “max times to increment” to whatever ceiling you want.

**Alternation**
In this case, if “alternate mod” is toggled on, then triggering the Modification will cause the desired change, and triggering it again will return the changed value back to its original state. So, in the above example, with the smoothing and incrementing cleared out, the Mod will alternate the gain between -10 and 0dB. This essentially combines Mod and Reset (covered next) into one function so one key can be used to both modify and reset. Note that this can be combined with smoothing.

## Reset

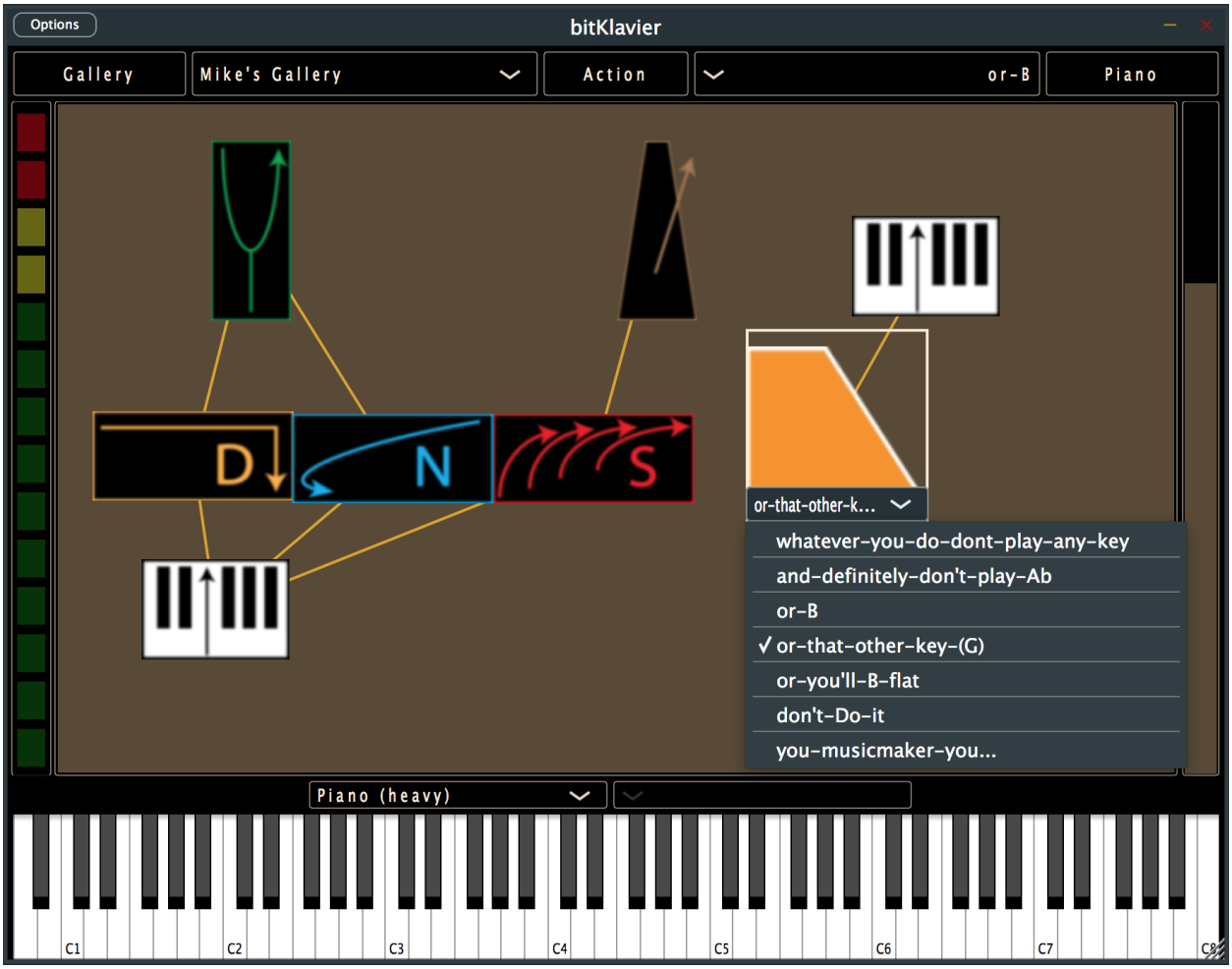
Reset is essentially the reverse of Modification: Upon receiving a signal from an attached Keymap, a Reset will bring an attached preparation to its default state. “Tuning 2” in the Example Galleries uses a Reset.




## Piano Map

The Piano Map preparation allows us to move through different Pianos in the same Gallery purely by using Keymap input. This could be used mid-performance to switch between movements, or while experimenting in bitKlavier with two different Piano presets.

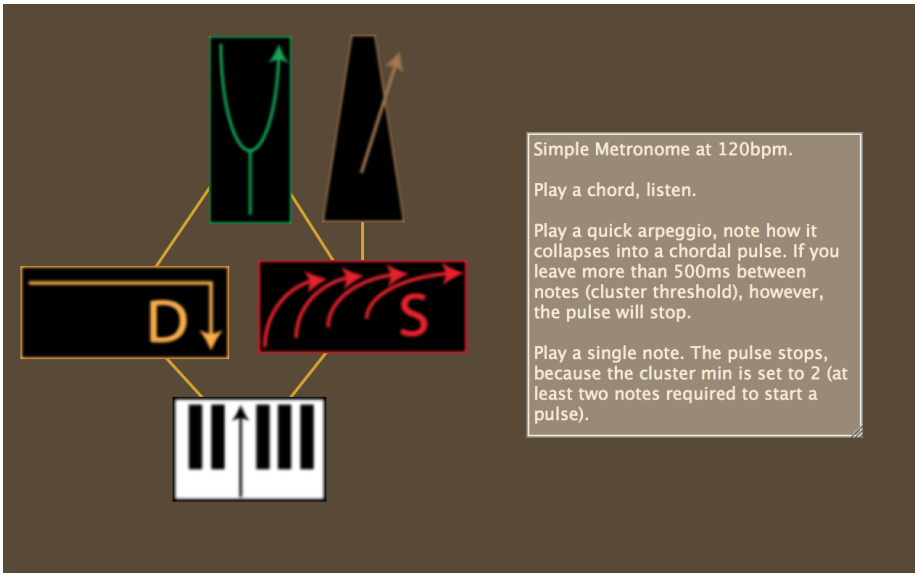
When a Piano Map is created, double-click to reveal a drop-down menu including all of the Pianos in that Gallery. When the Piano Map is set to a specific Piano, bitKlavier will automatically switch to the selected Piano after receiving *any* Keymap input to that Piano Map. A Piano Map is not a global control and will only exist in the Piano in which it is created. For a basic example of how the Piano Map can be implemented, take a look at “Mike’s Gallery.”




For another example of how to use the Piano Map, look at the “Piano Map” Gallery in the Examples folder (Example 27). This example uses the keys C3, C4, and C5 to cycle through three different preset Pianos.

## Comment

Hit (q) to add a comment to any Gallery, and double-click the text box to add notes, instructions, or any other text. Text boxes can be moved around and resized, and will save with the Gallery.




# Example Pianos

bitKlavier comes pre-programmed with a number of example Pianos that introduce, flesh out, and combine different preparations and settings. Here is a guide explaining a few select Pianos, many of which have explanatory text within the construction site itself.

Example Pianos:

- Synchronic (#1-9)
- Nostalgic (#10-15)
- Tuning (#16-20)
- Direct (#21-22)
- Adaptive (#23-26)
- Piano Map Gallery (#27)

## Synchronic 1

This first example Piano provides a simple demonstration of how Synchronic can be used to create a metronomic pulse in a bitKlavier Piano. In this example, the **cluster minimum is 2**, meaning it takes at least two notes played within the cluster threshold time to trigger a Synchronic sequence. The **cluster threshold is 500**, meaning Synchronic is looking for notes spaced less than 500ms apart to begin a Sequence.

## Synchronic 4

The **cluster min is 2** and **cluster max is 3** in this example, meaning any 2 or 3 notes played within 0.5 seconds will trigger a Synchronic sequence. A single note or any cluster of 4 or greater will stop the Synchronic sequence. This example also includes three sequenced **accents** of diminishing length, which cycle indefinitely.

## Synchronic 6

Example 6 includes two tuning preparations. The direct preparation is routed to Equal Temperament, and the Synchronic preparation is routed to Just tuning. This means that if you play a chord, such as a G major chord or arpeggio, the initial chord will be in Equal Temperament, and the triggered Synchronic triad will be Just-tuned.

This example also includes three sequenced **sustain length multipliers** of diminishing value, which cycle until the end of the sequence.

## Synchronic 8

In this example, none of the sequenced values have the same length, so the settings begin to phase and overlap.

## Nostalgic 3

This example features a **Nostalgic wave of equal length** to the duration of the note held down. Try playing a triad then lifting one note at a time to hear each note's corresponding length.

This example also includes an **Undertow** of 1000ms that begins 200ms before the beginning of the sample, which provides a crest and fade rather than bouncing back and forth from the head of the sample. Watching the blue lines over the waveform in the Nostalgic settings panel should clarify what exactly is happening here.

## Nostalgic 4

There are a few layers to look at with this example. At its core, the Synchronic pulse is determining the length of the Nostalgic swell. There are three Keymaps, each attached to a different preparation.

The notes C4 and E4 are routed to the Synchronic preparation. This means that playing C4 and E4 together will begin a Synchronic sequence. The Keymap attached to this Synchronic preparation includes only C4 and E4.

This Synchronic preparation is connected to the Nostalgic preparation, which is set to "Synchronic Sync KeyDown." The Keymap attached to the Nostalgic preparation includes all notes *except for* C4 and E4. This means that any note other than C4 and E4 will have a Nostalgic wave.

Because of the way these preparations are routed, every Nostalgic note will end as soon as the next Synchronic beat begins. The subsequent two Nostalgic Example Pianos (Examples 14 and 15) expand the functionality of these features.

## *Mikroetudes*

The *Mikroetudes* are a collection of etudes written by fourteen composers and performers in 2015 as a means of exploring and uncovering some of the many musical opportunities presented by bitKlavier. Inspired by the way Bartók's "Mikrokosmos" explore the identity of the piano from the ground up, these bitKlavier *Mikroetudes* were created by a variety of musicians who became acquainted with bitKlavier through their own composition/performance practices.

bitKlavier includes a preset Piano corresponding with each of the etudes in the *Mikroetudes* book. These Pianos can be used to perform the *Mikroetudes*, or can simply be used as jumping-off points of exploration. Some are wild and detailed, such as "Juxtaposed Weather" and "Keep It Steady," while others are simple, like "Petite Gymnopedie" and "Southwing."

For the full list of contributing composers, more information about the etudes, and to purchase the book of sheet music accompanying the bitKlavier *Mikroetudes* Galleries, please visit the Mikroetudes page through Many Arrows Music.

## Further Resources

For an example of how to use bitKlavier in live performance context, take a look at *Nostalgic Synchronic*, a series of etudes written by Dan Trueman for prepared digital piano. The *Nostalgic Synchronic* website includes descriptions of the compositional process, live performance videos and recordings, and scores for the etudes. The eight *Nostalgic Synchronic* etudes are written with corresponding bitKlavier Galleries, each of which is preprogrammed in bitKlavier.

To learn more about tuning and temperament, check out bitSuite, a collection of apps built as learning tools to help understand the physics and process of tuning a keyboard instrument. The bitSuite apps were designed by Dan Trueman and use many of the same interfaces as bitKlavier.

A new online course—Reinventing the Piano—presented by Kadenze makes extensive use of bitKlavier to teach about the history and ongoing development of the piano.

See these papers for more details about the history, design, and motivations for bitKlavier, and for a deep dive into tuning with bitKlavier:

- "bitKlavier: A Prepared Digital Piano." Dan Trueman and Michael Mulshine. Computer Music Journal, Vol. 43, Nos. 2-3 (2019)
- "Playing with Tuning in bitKlavier." Dan Trueman, Aatish Bhatia, Michael Mulshine, and Theo Trevisan. Computer Music Journal, Vol. 43, Nos. 2-3 (2019)

Finally, see the growing collection of short video tutorials.

# About & Credits

bitKlavier is an open-source project developed by Dan Trueman, Mike Mulshine, and Matt Wang, with Theo Trevisan, Jeff Gordon, and Katie Chou, at Princeton University.

The bitKlavier manual was written by Noah Fishman and edited by Dan Trueman, Noah Fishman, Mike Mulshine, and Matt Wang.

bitKlavier is the software engine that drives the prepared digital piano, running on MacOSX, iOS, and Windows, and has been used by dozens of performers—from young students to renowned professionals—in a range of compositions by Trueman and others.

Development of bitKlavier has been sponsored by the American Council of Learned Societies, the New Jersey State Council on the Arts, and Princeton University's Center for Digital Humanities, Council on Science and Technology, Council of the Humanities, and Department of Music.

bitKlavier is an open-source project, created in C++ using JUCE. Developers interested in joining the project can of course download the source from the GitHub site. Don't hesitate to contact us.

# Appendix

bitKlavier Temperament Tables by Ratio to Fundamental:

| Temperament | U | m2 | M2 | m3 | M3 | P4 | a4/d5 | P5 | m6 | M6 | m7 | M7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Just | 1/1 | 16/15 | 9/8 | 6/5 | 5/4 | 4/3 | 7/5 | 3/2 | 8/5 | 5/3 | 7/4 | 15/8 |
| Partial | 1/1 | 16/15 | 9/8 | 7/6 | 5/4 | 4/3 | 11/8 | 3/2 | 13/8 | 5/3 | 7/4 | 11/6 |
| Duodene | 1/1 | 16/15 | 9/8 | 6/5 | 5/4 | 4/3 | 45/32 | 3/2 | 8/5 | 5/3 | 16/9 | 15/8 |
| Otonal | 1/1 | 17/16 | 9/8 | 19/16 | 5/4 | 21/16 | 11/8 | 3/2 | 13/8 | 27/16 | 7/4 | 15/8 |
| Utonal | 1/1 | 16/15 | 8/7 | 32/27 | 16/13 | 4/3 | 16/11 | 32/21 | 8/5 | 32/19 | 16/9 | 32/17 |
| Common Just | 1/1 | 16/15 | 9/8 | 6/5 | 5/4 | 4/3 | 45/32 | 3/2 | 8/5 | 5/3 | 9/5 | 15/8 |
| Symmetric | 1/1 | 16/15 | 9/8 | 6/5 | 5/4 | 4/3 | Sqrt(2) | 3/2 | 8/5 | 5/3 | 16/9 | 15/8 |
| Well Tuned Piano | 1/1 | 567/512 | 9/8 | 147/128 | 21/16 | 1323/1024 | 189/128 | 3/2 | 49/32 | 7/4 | 441/256 | 63/32 |
| Harrison Strict | 1/1 | 28/27 | 9/8 | 32/27 | 5/4 | 4/3 | 112/81 | 3/2 | 128/81 | 5/3 | 16/9 | 15/8 |

For the historical temperaments, visit Carey Beebe's wonderful website, which describes all of the systems in bitKlavier, including how they are actually executed. See Kyle Gann's page for information about the La Monte Young "Well Tuned Piano" tuning, and this article by Giacomo Fiore and Larry Polansky for more about the "Harrison Strict" tuning.