Junior

If you don't know what chipmusic is, it's the practice of using (mostly vintage) video game systems to make music.

One of the most popular platforms was (and still is) the Nintendo Game Boy™, both due to its charming soundchip but also fantastic software such as nanoloop and Little Sound DJ which enabled artists all over the world to make original chipmusic in the palm of their hand.

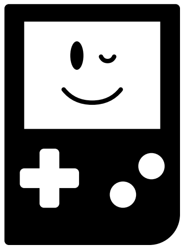
This is how I started making music in 2005, and 20 years later I still think there is nothing that sounds quite like it.

For the longest of time I've wanted to bring this raw and characterful sound into synth-form, augmenting its core with a modern feature set and extended capabilities while still staying true to the original.

This is Junior, it's a 4-bit wavetable synthesizer full of charm and history.

I hope you like it!

/ Ess



Junior is a plugin instrument and requires a host to run.

It's available in the CLAP, VST3 and AUv2 formats and is compatible with any host (such as a DAW) that supports any of these formats.

Junior is compatible with macOS (Universal), Windows and Linux.

Minimum supported OS versions are

macOS 10.13 (High Sierra)
Windows 10
Ubuntu 22

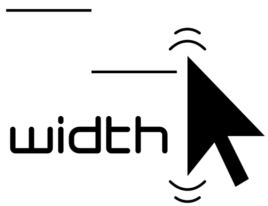
MIDI Control & tuning support

Junior supports Pitch Bend (+/- 2 semitones) and Sustain Pedal (CC64) control as well as continuous MPE pitch bend data (+/- 48 semitones)

In the CLAP version of Junior, Note Expressions are used for continuous pitch bends instead of MPE where available.

Junior also supports ODDSound MTS-ESP for on-note individual key tuning.

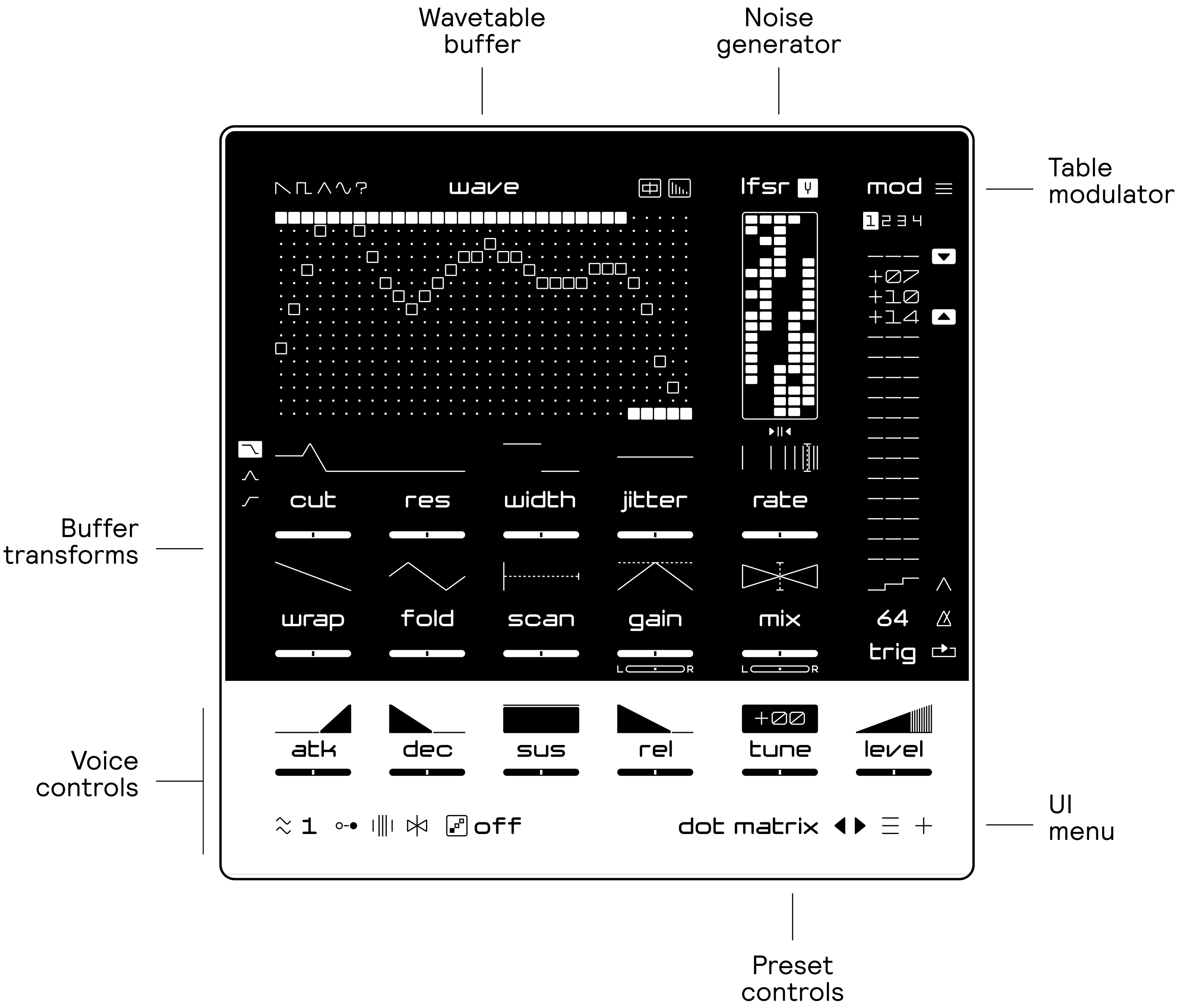
The parameters in Junior can be adjusted with your cursor, scrolling and via direct keyboard input.



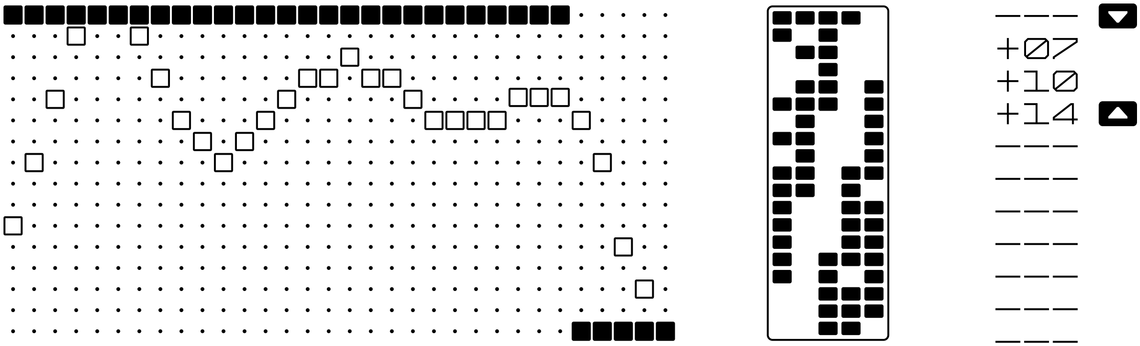
Click and drag the icon above the parameter name to change the value. Hold Shift to increase the sensitivity.

Click on the parameter name to enter a value by typing.

Double-click on a parameter icon to reset the value.



Wavetable buffer
Noise generator
Table modulator
Buffer transforms
Voice controls
UI menu
Preset controls
wave
lfsr
mod
1 2 3 4
+07
+10
+14
cut
res
width
jitter
rate
wrap
fold
scan
gain
mix
64
trig
atk
dec
sus
rel
+00
tune
level
1
off
dot matrix



Junior consists of three distinct modules:

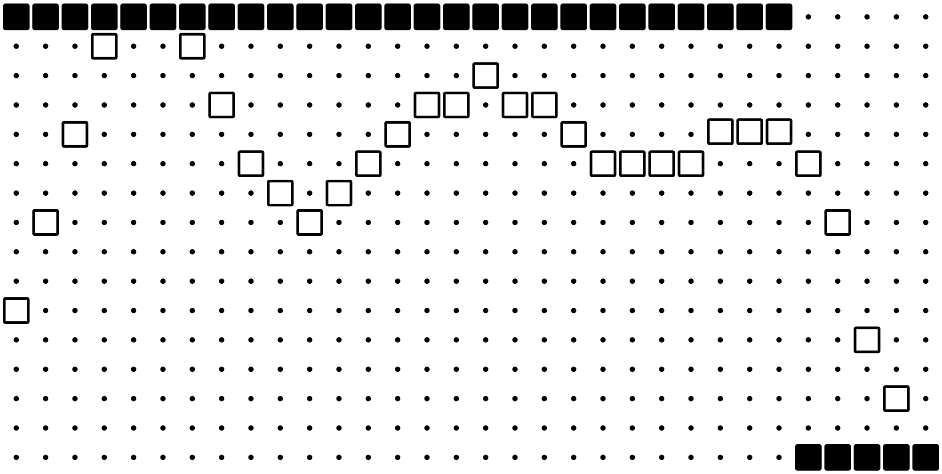
A 4-bit wavetable buffer, played back by an oscillator,

A 7-bit LFSR noise generator,

and four identical “Table” modulators.

All together they can be used to create all sorts of lo-bit sounds, from percussion to leads, sound effects and more.

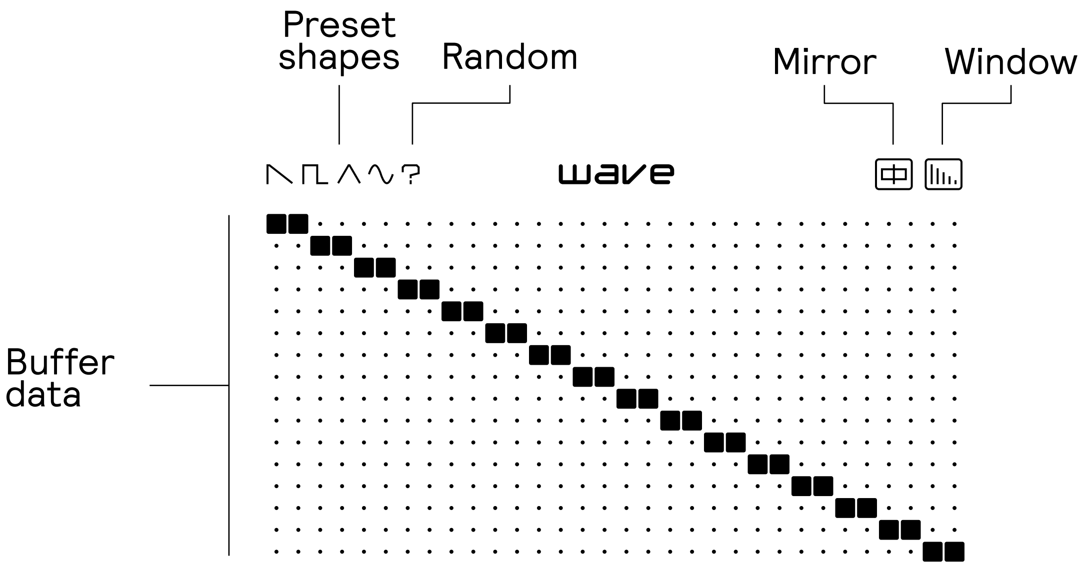
The basis and inspiration for these modules is partly the Game Boy™ sound hardware and partly tracker software, harnessing the strengths and characteristics of both while extending the capabilities to serve both the veteran chiptune musician and the curious synthesist.



Junior’s tonal sound generator is centered around a 4-bit wavetable buffer.

This is based on the “Wave” channel of the Game Boy™ which plays back chunks, a “buffer”, of 32 4-bit values. This has historically been used in games for wavetable synthesis, e.g playing back a set of values with an oscillator to create a periodic wave, or PCM playback.

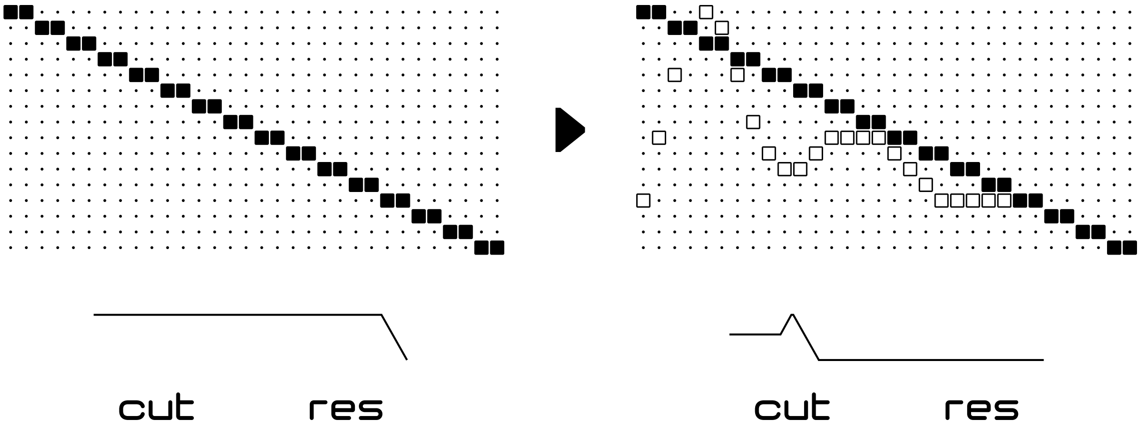
In Junior the buffer is treated as a custom waveform drawable by the user which then gets transformed by various controls that directly alter the buffer data in realtime.



The wavetable can be drawn into directly by clicking and dragging anywhere on the dot matrix canvas, or you can load common waveforms by clicking on the waveform icons above.

The wavetable can also be randomly generated by pressing on the question-mark icon.

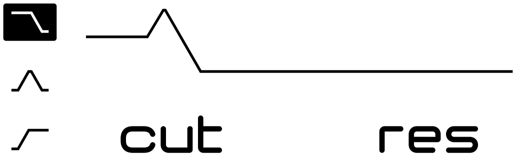
Pressing the top-right buttons will enable two different window modes which alters the buffer. Mirror will mirror the wavetable horizontally and Window will apply a fade-out of the wavetable which results in a smoother sound.



After drawing your waveform you can apply so-called Transforms which alter the buffer contents. These apply directly to the buffer data and changes the waveform itself.

For example, the filter pictured will apply a lowpass filter to the buffer data and then write it back into the buffer, resulting in a cyclic waveform.

Buffer transforms are not destructive, they can be altered back and forth while keeping the original data intact. The difference between the input data and the transform is shown as hollow blocks.



The filter transform applies a resonant multi-mode filter effect to the buffer. The left-hand buttons selects between Lowpass, Bandpass and Highpass.

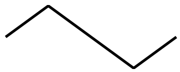
The width transform collapses the buffers towards the left side, which can be used to simulate pulse width modulation etc.

jitter

Jitter applies random values to each value, this is updated per sample and works essentially as a tuned noise.

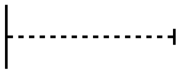
wrap

Wrap repeats the buffer contents horizontally up to an extreme number of harmonics which can be used to simulate oscillator sync.



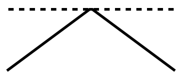
fold

Fold repeats the buffer vertically and folds it back, resulting in an effect similar to wavefolding.



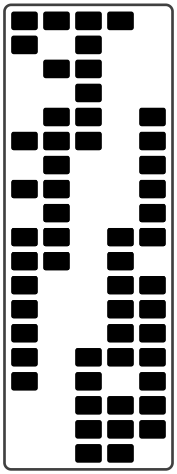
scan

Scan offsets and overlaps the buffer contents forwards and can be used for interesting phase effects.



gain

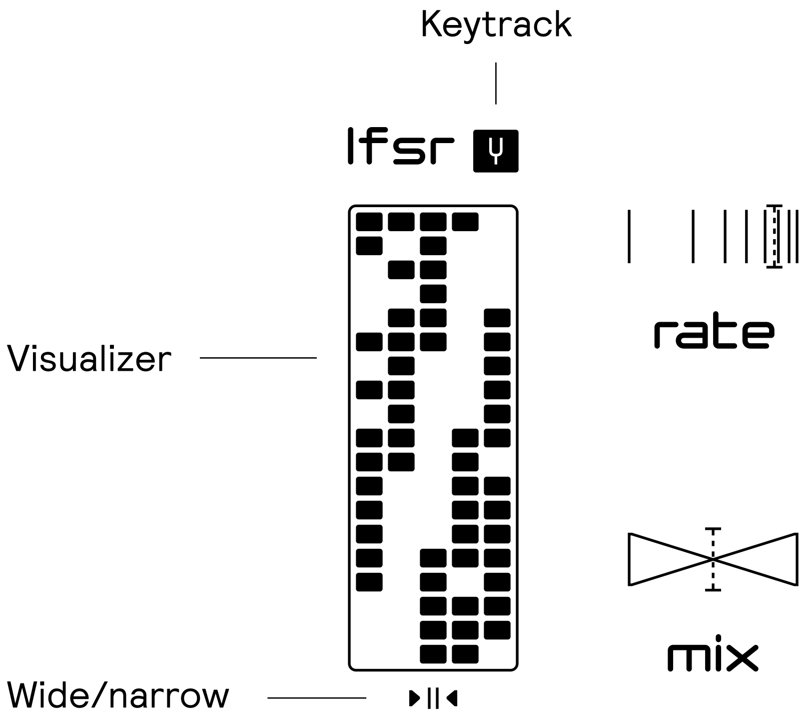
Gain adjusts the vertical height of the buffer and can be used to either clip or attenuate it. For purists, this is the way to achieve 4-bit envelopes that are true to the original.



The noise generator in Junior is a 7-bit Linear Feedback Shift Register (LFSR) which is a way to generate many kinds of deterministic sequences and is commonly used to generate pseudo-random noise patterns.

The actual register itself is 7-bit, but its output is either on or off resulting in a very rich lo-fi noise timbre.

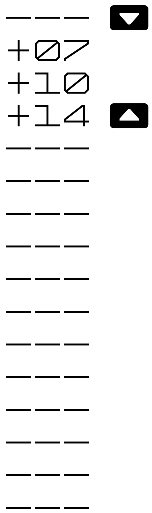
This can be used for all kinds of sound effects, percussion, textures and even tonal layers.



While it may sound quite intricate, the LFSR noise is very simple to interact with.

Rate controls the pattern of the LFSR, from darker to brighter tones. The default Wide mode sounds typically “noisy”, but the Narrow mode can produce more tonal/metallic results.

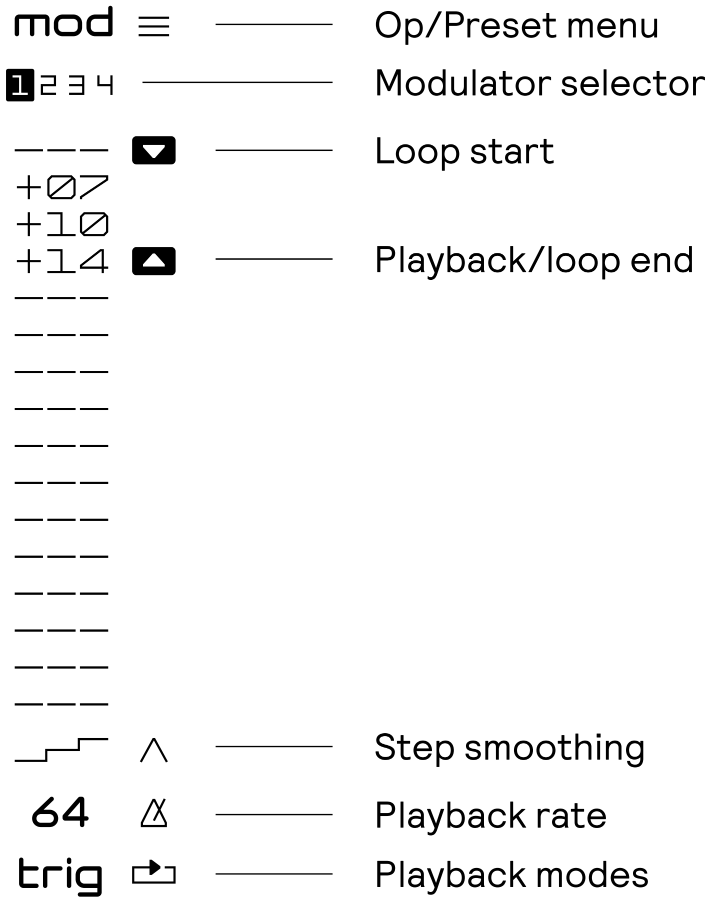
Keytrack will tie the LFSR frequency to the pitch of the synth, and lastly Mix controls the balance between the LFSR and the wavetable oscillator.

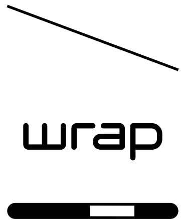


These two sound sources can be controlled by four “Table” modulators. Tables are essentially stepped modulation sequencers that are prevalent in tracker music software such as LSDj, Defmon and M8.

In Junior they are incorporated more as a synth-style modulation source with different playback modes, smoothing and MIDI control.

Tables are very flexible and can be configured as an LFO, envelope, arpeggiator, pseudo-random source and more.


mod
Op/Preset menu
1 2 3 4
Modulator selector
Loop start
+07
+10
+14
Playback/loop end
Step smoothing
64
Playback rate
trig
Playback modes



All parameters that have a slider underneath them can be modulated by the tables. The depth is increase by clicking and dragging the slider, and can be set both positively and negatively. The modulation is always relative to the value.

Each table has an individual set of depths per parameter, so when you select 1, 2, 3 and 4 the sliders will also change. This allows for very complex sound design.

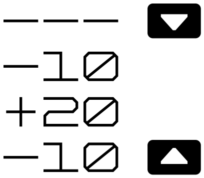
There are four unique table modulators that can affect the synth parameters individually, however they all share the same type of controls.

64

20 Hz

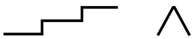
The rate of the table playback can be set either in Hz or tempo-synced note values such as sixteenth notes (16), dotted (16d) and triplets (16t)

You can switch between Hz and Rhythm by pressing on the button on the right side of the rate value.



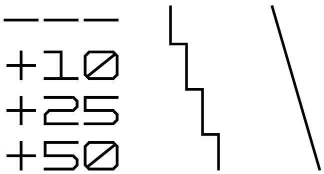
The 16 steps hold values between -100 and 100. To change the value, click and drag on any line.

The two arrows in the sequencer determine the region of the table that will be looped once it reaches the end.

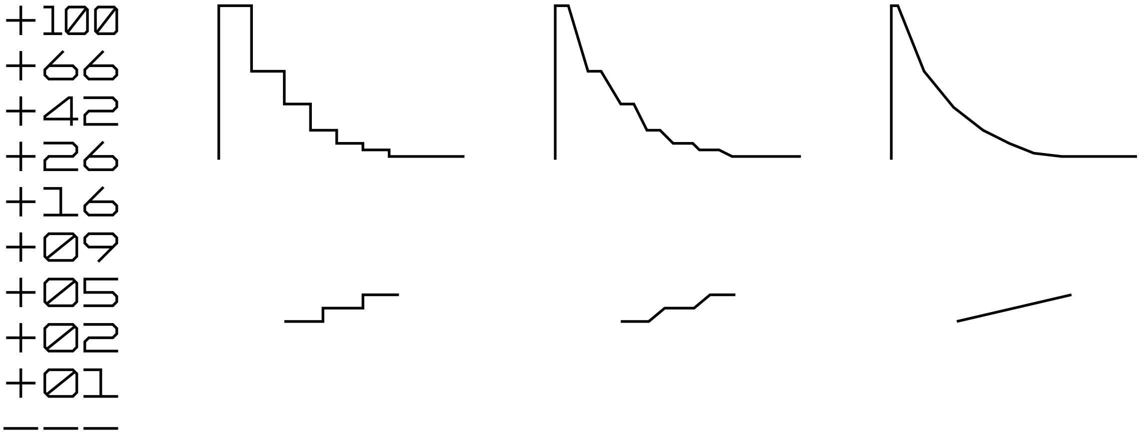


The table plays back stepped values by default, but they can be also smoothed out with the interpolation control.

Full interpolation will result in a linear step from one value to the other, which allows you to basically build a break-point envelope generator among other things.



For example, building an exponential envelope could look something like this:



With varying degrees of smoothing it’s also possible to build pitch slides, subtle ramping and so on.

The triangular button next to the smooth control determines whether the interpolation should be continuous when looping or if it should reset.

Resetting can be useful for looping envelopes for example.

There is a handful of different modes which can greatly affect how and what the tables do.

The Playback control changes the table traversal:

Once: this plays the table once and then stops, this is useful for envelopes.

Loop: this plays the table from the start, then loops the looped region indefinitely,

Ping-pong: like loop, but reverses the direction each time the loop point is reached.

Random: like loop, but randomizes the position each step.

The table Mode changes the way the table is iterated through:

trig	Trig will reset the table playback per each new note trigger. Useful for envelopes etc.
free	Free will run the table indefinitely, useful for LFOs for example.
hold	Hold will iterate the table one step per each new note trigger. Like a sample and hold.
midi	MIDI will index the table using an external MIDI source ... more about this on the next page!

The MIDI mode in the tables lets you use the table as a "lookup" with external MIDI sources. This means that you can map for example velocity or the mod wheel to index into the 16 steps of the table.

This lets you create complex MIDI mappings and take control of the modulation with external sources.

Note maps the incoming MIDI note.

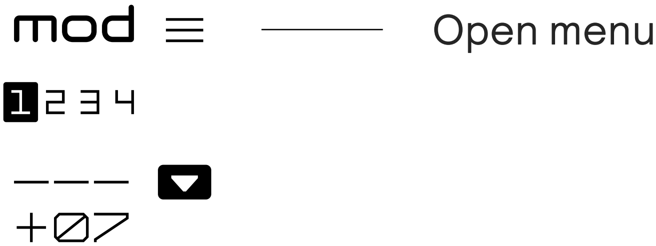
Velo maps the incoming MIDI velocity.

Wheel maps the incoming MIDI mod wheel.

Slide maps MPE slide (Y-axis)

Slide maps MPE pressure (Z-axis)

Last but not least, the tables have a menu which can be used to transform the table data and load/store presets that can be recalled from any session.



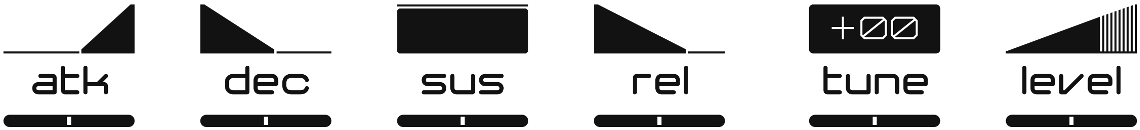
From this menu, you can save a whole table configuration as a preset, or perform actions which affects the values in the current table.

The “op” submenu contains operations that manipulate the values such as normalize, randomize, etc ...

Clear will remove the step data while Reset will reset all table values to default.

There is also a handful of presets of arpeggios, waves, ramps and so on...

These elements are all tied together by the voice controls in the bottom row. These control the amplitude, pitch and output level of the synthesizer voice.



Junior has an ADSR envelope which controls its dynamics, meaning the amplitude over time.

Attack (atk) controls the fade-in time of the envelope.

Decay (dec) controls the time it takes from where the attack stops to settle into the Sustain (sus) level.

Release (rel) controls the final fade-out time.

Tune and Level controls the semitone offset of the synth and output level respectively.

Underneath the main rows of voice controls lies the global controls. These control things that affect the whole patch, such as the voice count, main volume, effects and preset switching.

From left to right, the two waves denote Voice Count which chooses how many voices of polyphony are used, with a max of 16 voices. If this is set to 1 voice Junior will operate in Mono mode, which allows legato play.

The two dots with a line in between represents the Note Glide control, which will introduce a portamento / glide between each note being played.

The three lines denote the Unison control, increasing this will add two oscillators above and below the fundamental pitch. The control next to it sets the stereo spread.

In Mono mode it’s possible to use a chip-style mono arpeggiator which activates on overlapping notes.

dot matrix

Presets can be browsed by pressing the arrows next to the preset name, or you can open up the menu (lines) to select a preset manually.

In the preset menu you can initialize the current sound, or save it as a new preset in the User folder.

To name your sound, simply click on the preset name and enter a new name for it.

It’s also possible to generate a new randomized preset by clicking on the Rand option.

Presets that have been created with the Lite version of Junior will show up in a sub-menu (Lite) and are fully compatible with the full version of Junior.

Press the folder icon in the Preset Menu to view the preset folder in your system's file browser.

Presets are stored in a common XML format in these locations:

macOS:

`~/Library/Application Support/Junior/Presets`

Windows:

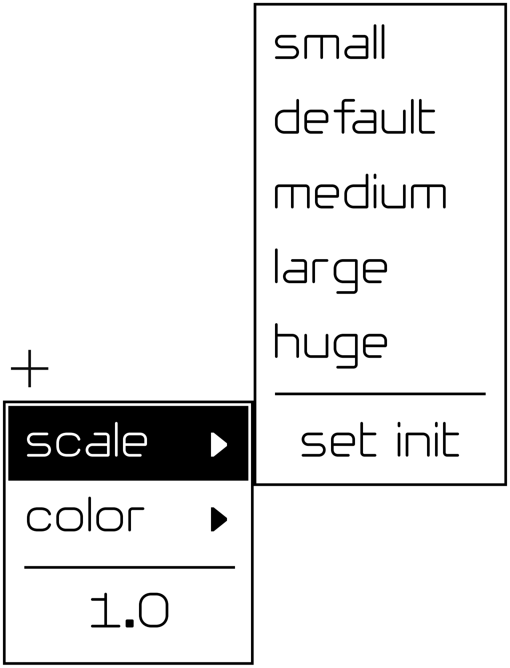
`C:\Users\[name]\AppData\Roaming\Fors\Junior\Presets`

Linux:

`~/.config/Fors/Junior/Presets`

For Junior-Lite the location is the same, but of course in a folder called Junior-Lite instead.

UI Menu



You can resize the window by dragging any corner, but if this is not available in your host you can change the size from this menu. Here it is also possible to set the initial size of the plugin when it's first loaded.

The Color sub-menu lets you change the color theme. It comes with a handful of factory themes, but it's also possible to make your own.

The color themes in Junior are stored in a simple XML format using hex color codes, e.g #000000 for black. Please see the factory theme files for reference.

Color themes are installed in these locations:

macOS:

`~/Library/Application Support/Junior/Themes`

Windows:

`C:\Users\[name]\AppData\Roaming\Fors\Junior\Themes`

Linux:

`~/.config/Fors/Junior/Themes`

You can of course manually make a custom theme, but we suggest using our Junior Theme Creator on our website:

`https://fors.fm/junior-themes`

Ok, that's all!
We hope you enjoy Junior.

Factory presets by

Beautiful Goblins
mikey303
Zabutom
laamaa
sm0hm

Special thanks to

cancel
Nullsleep
Salkinitzor
Johan Kotlinski

Fors

100% Digital Synthesis

Made in Sweden