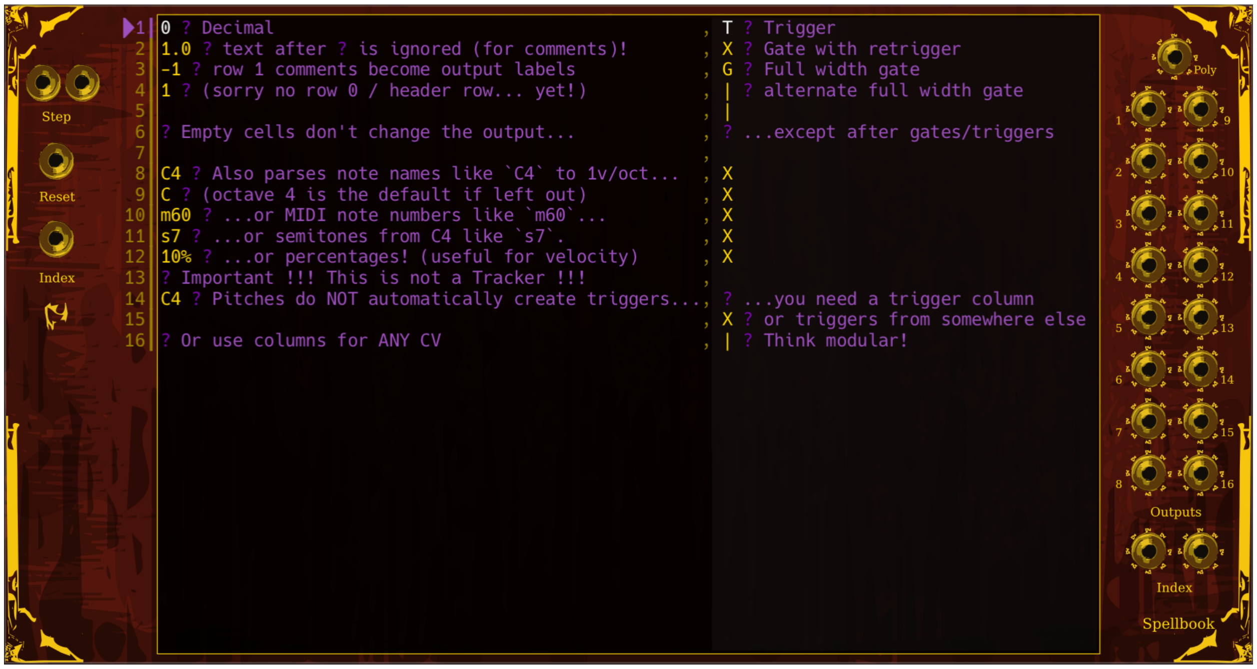
Contents

Spellbook



```
0 ? Decimal                                         , T ? Trigger
1.0 ? text after ? is ignored (for comments)!       , X ? Gate with retrigger
-1 ? row 1 comments become output labels            , G ? Full width gate
1 ? (sorry no row 0 / header row... yet!)           , | ? alternate full width gate
                                                    , |
? Empty cells don't change the output...            , ? ...except after gates/triggers
                                                    ,
C4 ? Also parses note names like `C4` to 1v/oct...  , X
C ? (octave 4 is the default if left out)           , X
m60 ? ...or MIDI note numbers like `m60`...         , X
s7 ? ...or semitones from C4 like `s7`.             , X
10% ? ...or percentages! (useful for velocity)      , X
? Important !!! This is not a Tracker !!!           ,
C4 ? Pitches do NOT automatically create triggers..., ? ...you need a trigger column
                                                    , X ? or triggers from somewhere else
? Or use columns for ANY CV                         , | ? Think modular!
```

Figure 1: Spellbook

Spellbook is a module for VCV Rack to sequence pitch and control voltage (CV) patterns in a eurorack-style environment using the plain text RhythML syntax. It has 16 outputs, each of which outputs a voltage controlled by the corresponding column in RhythML-formatted text input.

Inputs

- **Step Forward**: Advances to the next step in the sequence on the rising edge of a trigger.
- **Step Backward**: Advances to the prior step in the sequence on the rising edge of a trigger.
- **Reset Input**: Resets the sequence to the first step on the rising edge of the input signal.

- **Index Input**: Set the current step to a specific index, where 0v is the first step through to 10v for the last step, like a phasor controlling a “play head”.
- **Index Mode Toggle**: Toggle the yellow glyph to switch to “absolute address” mode, where 1v is step one, 2v is step two, etc.

Outputs

- **Poly Out**: Outputs all columns as channels of a polyphonic cable, for convenience.
- **Out 1 thru Out 16**: Individual outputs for the first 16 columns specified in the RhythML sequence.
- **Relative Index Out**: Outputs the current step as 0v = step 1, through to 10v = last step.
- **Absolute Index Out**: Outputs the current step as a voltage, e.g. step 3 outputs 3.0v.

Guide

If plaintext and spreadsheets make sense to you, you might like Spellbook. Very loosely inspired by oldschool music trackers, where it tries to capture that same sense of fast, keyboard-focused editing, adapted to the the modular eurorack/VCV Rack environment instead of MIDI. It also tries to capture the benefits of working in plaintext, like being able to copy and paste text snippets in Discord, or work with it in any text editor.

If other sequencers don’t jive with you, give Spellbook a try, but be warned: it is weird.

Spellbook sequences are described in plain text using the RhythML format, a syntax to define pitch and CV patterns in plain text. Sort of a “tablature” or “markup” vibe, rather than “scripting” or “coding” (there is no conditional branching, or calculations, or loops, or anything like that). Each row in the sequence represents one step, typically triggered sequentially by the “Step Forward” input. Text written in each “column” (defined by the commas) is parsed to determine what voltage to send to the corresponding output jack (one per column) for the current step.

RhythML Syntax Quick Start

Ultimately, every output is merely a simple voltage, but the syntax allows you to “express” a desired voltage in a variety of ways; whichever works best in context and for your brain. You don’t even have to stay consistent from row to row or column to column.

Control Voltages

- **Decimal Voltages**: Type a normal decimal number, with optional decimals or negative sign, and you get that number as a voltage.
- **Percentages**: Numbers ending in `%` (e.g. `50%` or `12.5%`), are normalized to eurorack CV standards so that 0% = 0v and 100% = 10v.
  - This is the range most modules expect for CV inputs, but of course you can scale them to other ranges if needed using the core Rescale module.
  - The values are not *clamped* to 0v-10v, you can also enter `-12.5%`, `300%`, etc.

Pitch Representations

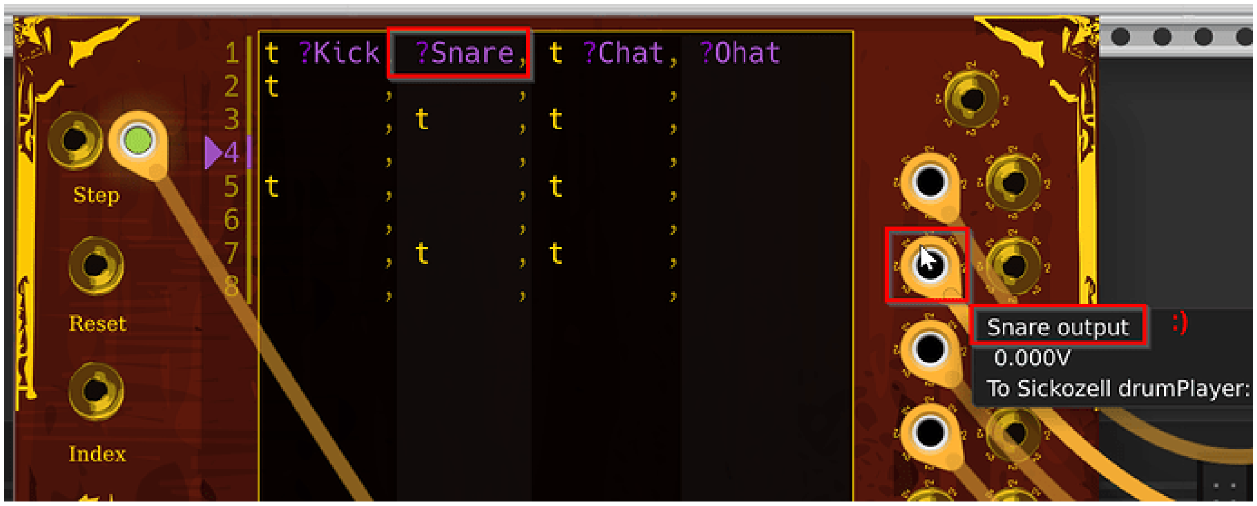
These formats are all parsed and translated into the equivalent 1v/octave control voltage. Decimals are allowed for all of them, but microtones may not be supported by all things you send those signals to (for example VSTs, DAWs, and MIDI mostly round to the nearest MIDI note, unless you do a little *mad science*, but most VCV Rack modules will accept microtonal pitches no problem). Case is NOT sensitive. Errors or undefined values become 0v outputs.

- **Scientific Pitch Names**: Specify pitches by name, accidental(s), and octave (e.g., `C4`, `G#3`, `Ab4`, `C#4`). `C4` and `C` = 0v.
  - You can stack and combine as many accidentals as you want: `C##4` parses to the same voltage as `D4`, for example.
  - Microtonal accidentals are also available: `$` for half-sharp, and `d` for half-flat (e.g. `C$4` for "C half-sharp 4")
- **MIDI numbers**: Numbers prefixed with `m` (e.g. `m60`) are parsed as MIDI note numbers. `m60` = 0v.
  - Decimals are allowed, but most MIDI devices will round it back to the nearest MIDI number anyway, so for microtonal MIDI you could send a pitch bend from a second column to get those microtones, and make sure your MIDI instrument handles pitch bending.
- **Semitones**: Numbers prefixed with `s` (e.g. `s7`) are parsed as semitones relative to C4. `s0` = 0v.
- **Cents**: Numbers ending with `ct` are parsed as cents relative to C4. `0ct` = 0v.
- **Hertz**: Numbers ending with `Hz` are parsed as frequencies. `261.63Hz` = 0v.

Gate and Trigger Commands: These are just shorthand so you don't have to type out a bunch of numbers for things like drum sequences which don't care about pitch or the exact voltage.

- `W` or `|` for a full-width gate; this one has no rising edge, so there will be no gap between this step and the prior step. This is identical to simply writing "10" or "100%" in the cell. The basic use case is to hold a gate open from the prior step for multi-step gates.
- `T` or `^` for a 1ms trigger pulse (this also guarantees a rising edge, so you'll get 0v for 1ms, then 10v for 1ms, then 0v until the next step), so that the output *doesn't* stay high for the entire step. This is usually what drum or clock-related modules will prefer.
- `X` or `_` for a 10v output which guarantees a rising edge when the step begins even if the output was already at 10v, by dropping to 0v for the first 1ms.

Comments: A `?` in a cell will begin a "comment"; it and all text for the rest of that cell will be ignored and highlighted in a different color. You can use these for labels, in-line comments and notes, or anything else where seeing a little text might be helpful.



```
1 t ?Kick  ?Snare, t ?Chat, ?Ohat
2 t       ,       ,       ,
3         , t     , t     ,
4         ,       ,       ,
5 t       ,       , t     ,
6         ,       ,       ,
7         , t     , t     ,
8         ,       ,       ,
```

Snare output
0.000V
To Sickozell drumPlayer:

Figure 2: Spellbook shows tooltips based on first row comments

- If there are comments in the first row, Spellbook assumes they are columns labels, and they will be shown in the tooltips for the output jacks.

Refer to RhythML for comprehensive guidelines on the syntax. Check out the manufacturer presets in the module's right click menu for examples and templates of various types of sequences, including some mad science like using Spellbook as a wavetable. If you think of any other number/voltage/pitch formats for which there's a good (mathematical) way to translate them into 1v/oct, let me know and we can add it to RhythML!

Timing with Spellbook

Sequences in Spellbook can be played step by step (e.g. using an incoming clock or other trigger source), or by "index". Importantly, RhythML itself has no concept of "time" or "duration", only "steps in a sequence". It's up to you to decide how to clock or index Spellbook to actually play a RhythML sequence, and what each "step" means- is it going to be clocked on 8th notes? Whole notes? Bars? None of the above because you're doing some modular mad science?

You might step a Spellbook once per bar, for a sequence which controls the chord progression, or you might clock it on 16th notes for a drum loop.

It's often useful to have one Spellbook for each musical part in your patch, so they can be sequenced and timed independantly based on the overall need of the music and the patch, and kept in sync using traditional modular methods such as a master clock and clock dividers. Spellbook makes working with multiple clock speeds really easy because RhythML sequences can be any arbitrary length; just add or remove rows.

Steps

- Step Forward / Step backward: Acts the most like a basic "clock in". Simply advances the sequences to the next or prior row each time they're triggered, wrapping around to the first step at the end. Ignored if anything is connected to the Index input.

Index

- The Index input is in Relative mode by default, where it acts like a phasor input: If you send a smooth rising sawtooth control voltage, Spellbook will set the "currently active step" as the *first* step when the Index voltage is 0v, and the *last* step when the Index is 10v, and the proportional step for every voltage in between. If you sync two Spellbooks with different length sequences to the same Phasor for their Index, this is a great way to get easy polyrhythms or polymeters.
- Click the small gold symbol to change Index to Absolute mode. In this mode, Spellbook expects an Index voltage representing exactly which step to be on like an address: 1v sets Spellbook to step one, 2v is step two, 14v is step fourteen, and so on. If you send a higher voltage than you have number of rows, it will wrap around (for the nerds: modulo sequence length). Unlike Relative mode, even if the length of the sequence changes, the same index voltage always takes you to the same row.

Controls and Hotkeys

The Spellbook module offers a variety of hotkeys and controls for managing its interface and functionality effectively. Here is a comprehensive list of controls and hotkeys available for the Spellbook module:

Text Field Behavior:

- Click anywhere inside the text box on Spellbook's panel to enter "editing mode". You'll see a text cursor when focused. The prior sequence will continue playing "in the background", unchanged as you edit, until you "commit" your changes.

- Edit your sequence, making sure to follow the syntax rules for RhythML.
- Click anywhere outside the text box to leave editing mode, or press `Ctrl+Enter`, to "commit" the text without losing focus (so you can keep editing).
- Spellbook will trim and/or pad cells so that columns align visually, and parse the updated RhythML as a new sequence.
- The parser evaluates each cell to convert it into an appropriate output voltage. Errors default to 0 volts.
- It tries to stay on the same "current step" if it can, but will modulo the current step into the new sequence length if the new sequence is shorter, which should hopefully help live-editing stay in sync without relying on frequent Resets, if you're careful.

Special Keyboard Shortcuts:

- `Ctrl+Enter`: Commit and parse the current text, but stay in editing mode.
- `Ctrl+[` or `Ctrl+]`: Decreases or increases the text size.

Additional Notes:

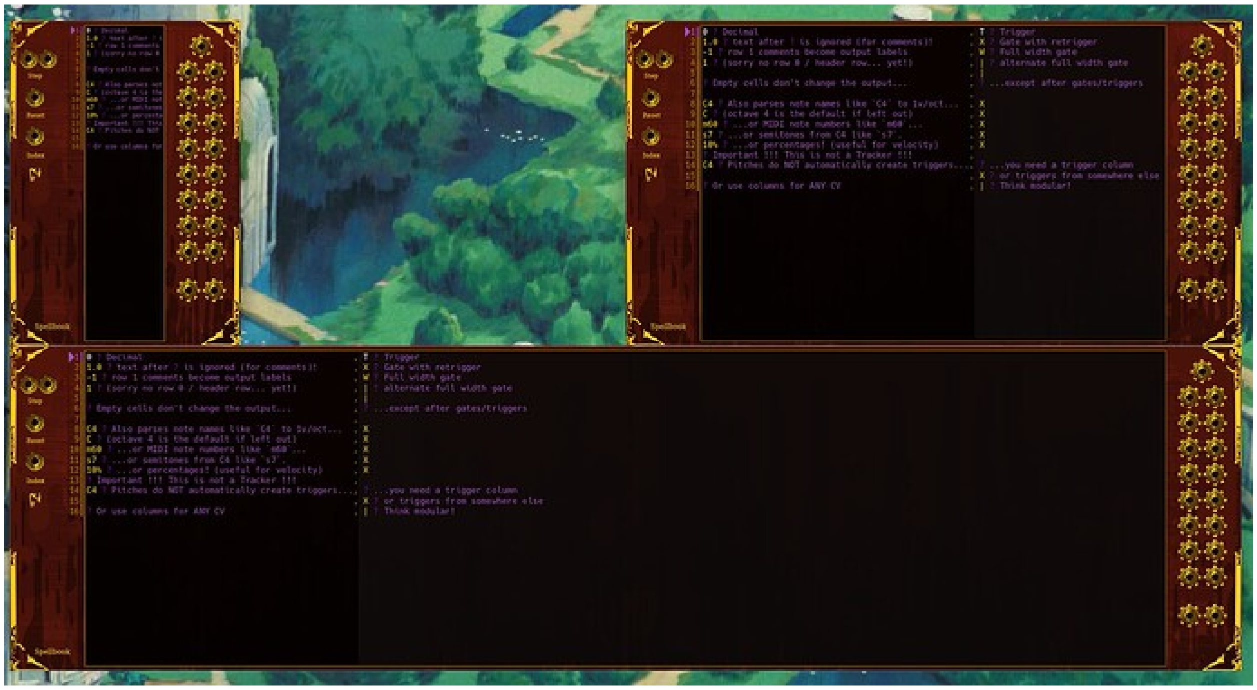

```
0    ? Decimal                                          , T ? Trigger
1.0  ? text after ? is ignored (for comments)!          , X ? Gate with retrigger
-1   ? row 1 comments become output labels              , W ? Full width gate
1    ? (sorry no row 0 / header row... yet!)            , | ? alternate full width gate
                                                        , |
? Empty cells don't change the output...                , ? ...except after gates/triggers
                                                        ,
C4   ? Also parses note names like `C4` to 1v/oct...    , X
C    ? (octave 4 is the default if left out)            , X
m60  ? ...or MIDI note numbers like `m60`...            , X
s7   ? ...or semitones from C4 like `s7`.               , X
10%  ? ...or percentages! (useful for velocity)         , X
? Important !!! This is not a Tracker !!!               ,
C4   ? Pitches do NOT automatically create triggers...  , ? ...you need a trigger column
                                                        , X ? or triggers from somewhere else
? Or use columns for ANY CV                             , | ? Think modular!
```

Figure 3: Spellbook at various sizes

- **Resizing:** You can resize the module by dragging the right edge of the panel, to handle different numbers of columns in your sequences. I place minimized Spellbooks with one-column sequences all over my patches for short simple loops all the time.
- **Autoscroll:** When not in editing mode, the text field autoscrolls to keep the currently "playing" step centered, so you can see what the sequence is doing as it plays.
- **Scrolling**: While in editing mode, you can scroll up and down using the mouse wheel, or in any direction by moving the text cursor until it touches the edge of the viewport.

Appendix

Basics of VCV Rack

VCV Rack is virtual modular synthesizer platform for Windows/Mac/Linux that simulates Eurorack modules, in addition to original modules that go beyond hardware. You place modules on a grid, interact with their knobs, sliders, and buttons with your mouse, and connect their input and output jacks with virtual cables to create synth patches just like a physical eurorack.

Spellbook is a module for VCV Rack, and RhythML was designed with VCV Rack, and modular synthesis in general, in mind.

All signals in VCV Rack are virtual voltages (really just “a number”, a fact which many modules in T’s Musical Tools play with), but they can roughly classified into:

- Audio signals are audible if played through your speakers. They contain audio-rate frequencies typically between 20Hz to 20kHz.
- CV (control voltage) signals can modulate parameters of other modules. For example, an LFO (low-frequency oscillator) can oscillate the pitch of a VCO (voltage-controlled oscillator) or the volume level of a VCA (voltage-controlled amplifier).
- 1v/oct (1 volt per octave) signals are CV signals that represent a pitch or note. In this standard, an increase of 1V increases the pitch by 1 octave. Since there are 12 semitones in an octave, an increase of 1/12v increases the pitch by 1 semitone.
- Gate signals are treated like an on/off signal. A “low” signal represents “off”, and a “high” voltage represents “on”.
  - Typically you would send 0v for low, and 5v-10v for high. The standard in VCV Rack is to accept 1v or more as “high”, and send 10v by default, which lets you route gates directly into a VCA and treat that VCA as a Voltage Controlled Gate (for example).
  - For example, the core MIDI module outputs a separate pitch signal and gate signal when a key is pressed, which could be sent to the pitch CV of an oscillator, and the trigger/gate of an envelope generator, respectively.
- Trigger signals are very short gates (usually around 1 millisecond), typically used for cases where the “length” of the gate is unimportant, like a drum trigger, or a clock pulse, though many modules freely accept either gates or triggers in many use cases no problem.
- Clock pulses are gates or triggers played at a steady tempo, in order to set musical timing. Anything that sends a steady pulse can become a clock, such as an square wave LFO, or you could use a dedicated clock module with features such as creating multiple related clocks or aligning to traditional time signatures.
- Other: Some patches might intentionally “misuse” signals, or define their own specific uses within parts of the patch, etc.

Signals can be connected from module to module via patch cables. It doesn’t matter what type of signal a cable carries—you can connect any output to any input.

Polyphonic Cables

In VCV Rack, “polyphonic cables” refers to the ability of a single cable to carry multiple voltages simultaneously. This could be multiple pitches, in the traditional sense of “polyphony”, but really it can be ANY set of voltages in one cable. This is a powerful feature that allows for complex patches with fewer visible connections.

- A polyphonic cable in VCV Rack can carry up to 16 separate channels of voltage.
- Polyphonic cables appear thicker than monophonic cables in the interface.
- Each channel in a polyphonic cable is its own independent voltage, like a regular (monophonic) cable, and modules might process them in parallel, in sequence, or use them in

other ways.

- If sent to a monophonic module, they usually read only the first channel, but polyphonic modules can do all sorts of things with the extra channels.