

virta

Sound-controlled
synthesis and effects

MADRONA LABS

This manual is released under the Creative Commons Attribution 3.0 Unported License. You may copy, distribute, transmit and adapt it, for any purpose, provided you include the following attribution:
Virta and the Virta manual by Madrona Labs. http://madronalabs.com.

Version 1.0, March 2016. Written by George Cochrane.

Illustrated by David Chandler.

Typeset in Adobe Minion using the TEX document processing system.

Any trademarks mentioned are the sole property of their respective owners. Such mention does not imply any endorsement of or association with Madrona Labs.

Introduction

Synthesizers and effects. In this crazy mixed-up world, what could be better? On and on, we tweak and twist, searching for something. Smiles creeping across our faces all the while. The roiling sea of knobs and buttons and lights are magnetic. We seek, pure of heart, to merge with the world of sound.

Alas, we sometimes find ourselves stopped short.

O, that this too solid flesh could melt... And, like, seep into the jacks of my modular.

It can be a tall order to get the level of expression and immediacy that you crave with traditional synthesizers and processors. This goes double if you've got to work quickly, but want to maintain a satisfying amount of complexity and variance in your sound. Perhaps, in the heat of a creative moment, you have wished to simply dive into a synth or effect and command it by thought. There is no shame in that.

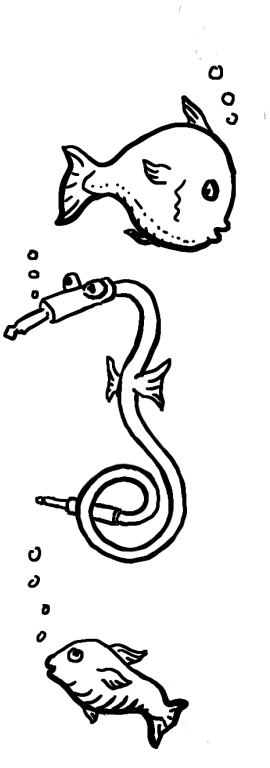
I mean, people talk to their *PETS*, for crissake.

This, to a large degree, is why we've created Virta—a modular synthesizer and effects processor that responds directly to *you*.

When you route your voice (or any kind of audio) into Virta, the sound analysis engine precisely extracts information about its pitch, timbre, and dynamics over time. These new signals can control nearly any parameter in Virta, just the same as you can with MIDI or OSC.

This makes for an instrument that can easily follow your every twist and turn. Living the dream, really. Then again, Virta is modular, so audio-controlled synthesis is just one road you can travel. You can also make stuff like:

- Expressive vocal processing chains
- 4-voice MIDI-controlled pitch shifters
- MIDI/OSC-controlled synths (with or without audio control)
- Drones for days
- First-ever faithful emulation of the mating call of the Space Pony



A Quick Overview

Virta's interface has four main sections, from top to bottom:

- The *header* section, where you'll find the patch menu, along with the version and registration status of your copy of Virta.
- The *signals and shapes* section, where AUDIO and MIDI/OSC (KEY) signals flow into Virta, and where the oscillators (OSC1 and OSC2) and modulators (LFO, and ENVELOPE) reside.
- The *patcher* section, where connections between modules are made and unmade.
- The *processing and output* section, where the sounds you make are massaged and processed to perfection by the FORMANT, GATE, DELAY, and OUTPUT modules.

Of course, there is much more to learn about the ways of Virta. Let's go on a little cosmic journey together.

A Little Book of Power

If Virta is your first foray into anything modular, or your first audio-controlled alien communicator, let this manual be your guide. Whatever your perspective, we mostly hope that you have fun exploring this new instrument, making sounds you've never heard before.

This manual is arranged in three sections:

- Where Keyboards Fear to Tread—A little background on the concepts behind Virta.
- Getting to Know Virta—Info on how Virta fits into your system, and how to use the various types of controls and connections that you'll find.
- Module by Module—A guided tour through the different parts of Virta, module by module.

1 Where Keyboards Fear to Tread

Virta is all about *immediacy of expression.* We all want to make evocative synth parts and process sounds in novel ways, but oftentimes, doing so takes a little more thinking than we'd like—a lot of futzing with controllers, note and parameter values, automation, and so on. That's all well and good, but sometimes you just want to let fly, make bold moves, and get compelling results. This can be a challenge in a world where the most responsive control mechanisms we've got are keys, knobs, and faders. Virta to the rescue!

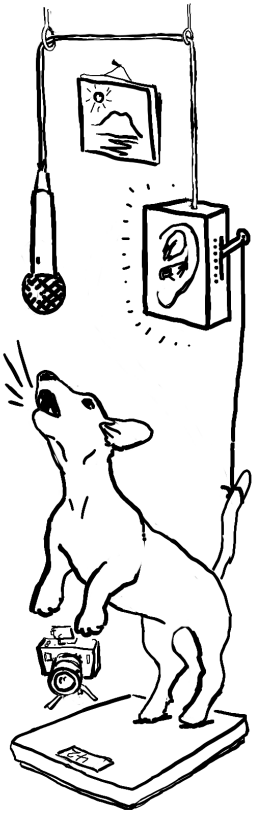
You feed sounds into Virta (often voice, but it could be *anything*), and they are analysed by the AUDIO module. This derives pitch, amplitude, on-off state, and some timbral details from your audio, and outputs them as signals you can use to control anything in Virta, just like you can with MIDI or OSC. When you're doing audio-controlled synthesis, you'll often patch these signals first into the pitch and timbre control inputs for the oscillators, which are only too happy to follow along.

The AUDIO module is optimized for monophonic input signals, but feel free to try anything you like, if you're okay with unpredictable results.

Then, you can use the FORMANTS and GATE modules to apply the timbre & amplitude of your input source to this new synthesized sound in a wide variety of ways. Add on a pitch-shifting delay effect (also chock-full of control inputs), and you've got quite a hot musical pie.

In other guises, you could patch up a keyboard-controlled synth, and use FORMANTS like a really nice vocoder to shape the sound with incoming audio. Use your voice to vary delay time or control the intensity of modulations—or move a dozen things at once, all in whatever range of motion you want. The creative avenues afforded by being able to treat audio as both a multi-parameter controller and useful musical signal are legion. Even without audio control, Virta is still a powerful, patchable synth and processor.

Virta is quite modular, so the way you patch it dictates the way it will act. May you patch with verve and reach new horizons.



2 Getting to Know Virta

This chapter shows you how Virta fits into your computer ecosystem, and gives you a working knowledge of the various types of controls you'll find throughout. Some of Virta's controls act just like you would expect, but some have more personality than that.

First Things First

Virta is a software *plug-in* that comes in both VST and AU formats. A plug-in does not run on its own. It runs within an application (known as a *host* or *Digital Audio Workstation*). Some good hosts include Ableton Live (on Mac and Windows), Logic and Numerology (Mac), and FL Studio (Windows). All of these hosts provide easy ways to use plug-ins like Virta.

Every host handles instruments a little differently, so for more information on using instrument plug-ins in your own system, please see the user guide that came with your DAW.

The rest of this guide assumes that you've got Virta up and running just fine. If you have problems with installing or operating Virta, please search our forums at http://madronalabs.com or, if that fails, send us your questions at support@madronalabs.com. We, and the growing community of Virta users, are here to help you.

Routing Audio & MIDI or OSC

If you're using Virta with audio input only (no MIDI or OSC input), you can go ahead and simply drop Virta onto an audio track in your DAW. Once you've done that, you can enable recording or input monitoring for that track, start pumping in the audio of your choice, and you're off to the races. However, for many uses, you'll want both audio and MIDI or OSC to be able to flow into Virta.

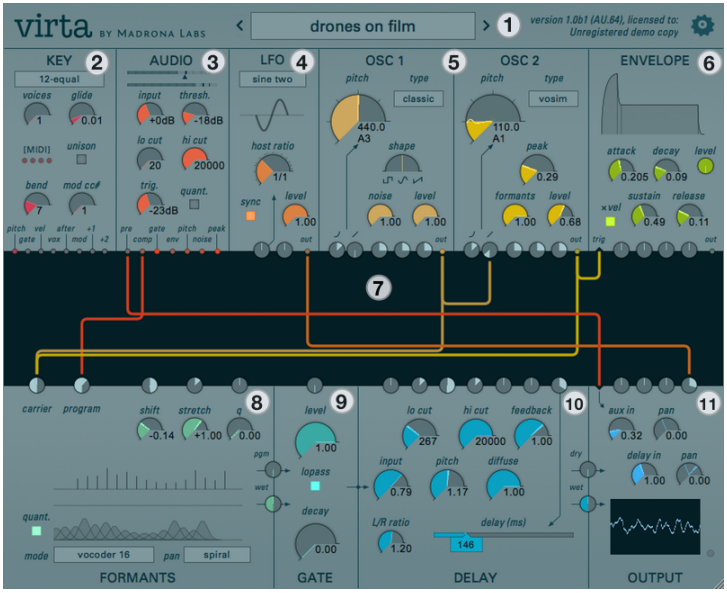
The procedure for doing this will differ from DAW to DAW, but as an example we'll show you how to do this in Ableton Live:

As much as we might like to, we can't explain how to use every DAW here. But between your DAW's manual, your DAW's friendly support people and ours, hopefully you can get Virta happily munching down audio and MIDI data without too much trouble.

- Create an audio track and a MIDI track. Assign the intended inputs to both tracks
- Insert Virta on the audio track.
- Using the output selectors on the MIDI track, set its output to the audio track containing Virta, and then to Virta itself.
- Record-enable both tracks by holding the Command key and pressing the record arm buttons on both tracks.



An Annotated Map of Virta



This section gives a quick description of each of Virta's areas, to give you a little familiarity before we go deep. For in-depth information on each module, control, and feature, see Chapter 3, *"Module by Module."*

1. Header

   This area's most vital item is a nice big preset selection menu with back and forward buttons for fast switching. On the right, you can find your license information as well as the plug-in format for the current instance of Virta. Click the gear-shaped icon to access the global settings that affect the way Virta displays info and responds to MIDI & OSC.

2. KEY

The KEY module receives the note and control signals you send Virta over MIDI or OSC, and makes them available to the rest of Virta's devices. If you have experience with CV-controlled synths, you can think of the Key module like a MIDI-CV converter box that outputs digital control "voltages."

3. AUDIO

The AUDIO module is the key to Virta's powers. It receives incoming audio, analyses it, and outputs a slew of useful signals that you can use to control and interface with the rest of the modules. It also outputs the raw audio input signal, so you can run it through Virta's processing tools.

4. LFO

The LFO module creates low-frequency oscillating control signals, useful for modulating all sorts of parameters. A variety of traditional LFO waveforms are available, as well as a couple that may surprise you.

We apologize about the shirt. Is it dry-clean only?

5. OSC 1 & OSC 2

Now arriving at Grand Waveform Station. OSC 1 & 2 create synthesized signals that can form the core of your synth sound, or act as carrier signals to feed the FORMANTS module (and a lot more besides). With a wealth of control inputs, these OSCs are all about responding to you, whether you use audio or MIDI/OSC to get things going. Each oscillator has two modes: one for making traditional analog waves, and one for expressive vocal tones.

6. ENVELOPE

The ENVELOPE module creates time-based control vectors, commonly used to control volume or timbre as notes are struck and/or held. Envelope is a fairly standard ADSR (Attack-Decay-Sustain-Release) type, but with fully mod-able settings, and a trigger input that lets you get creative about how to fire things off.

7. PATCHER

The Patcher is the dark central strip in the plug-in window, surrounded on all sides by the Modules. The patcher lets you connect signals from the outputs of modules to the inputs of modules. It is notable that multiple inputs can be fed from a single output, or multiple outputs to a single input. We think this is way more powerful and easy to use than a ton of menus.

8. FORMANTS

The FORMANTS module acts a lot like a vocoder, but with several big surprises up its sleeve. Several modes of multi-band filtering are available, all designed to coax interesting, expressive, vocal-like sounds out of whatever waveforms it touches.

9. GATE

If you're familiar with modular synth jargon, the GATE module can be described as as a VCA (Voltage Controlled Amplifier/Attenuator) or an LPG (Low Pass Gate) depending on the mode you choose. Either way, GATE works in concert with signal sources like ENVELOPE and LFO to change the level (VCA mode) or the level and low-frequency cutoff (LPG mode) of the input signal.

10. DELAY

Love crazy pitch-shifting delay effects? Fancy controlling one with *your mind*? You'll adore the DELAY module. This delay includes the aforementioned pitch shifter and a diffusor in its feedback path, and a variety of filtering and panning options.

11. OUTPUT

The OUTPUT module is where all of your audio comes together before venturing out into the wilds of your DAW. It includes controls for mixing and panning the output of the DELAY module (and thus, the whole bottom row of processors) along with any signal(s) you care to patch into the provided Aux input. It also contains a handy oscilloscope, which gives you visual feedback about the sounds you're making.



Presets

We labored heavily to make Virta simple and inviting to use, but flipping through the preset sounds first is a good way to hear what it can do, and see how it's done. Virta comes with both *user* and *factory* presets. The user area is where you'll keep your own creations, and where we put contributions from other Virta users that we include. The factory presets are meant to be a small and well-rounded set of sounds that you'll come back to often.

Using Dials

So, you'd like to go beyond the presets? Of course you would! Meet *dials*. They're found in every module. Like knobs on any piece of gear, dials are mainly good for two things: manipulating signals and giving you information. However, whereas most knobs inform you merely about a single unchanging value they've been adjusted to, Virta's dials act as tiny signal viewers as well. This means they not only show you the value you've adjusted them to, they also show you the values they're being pushed and pulled to by incoming modulation signals.

To modulate a dial's signal, just make a connection to the dial's signal input in the patcher. Every signal that can be modulated has a signal input next to it—this is how Virta can provide so much control without using menus. Signal inputs are like small dials without displays, or regular knobs, if you like. We'll cover the patcher and signal inputs thoroughly in a later section.

Dials as Controls

To set the position of a dial, you can do any of the following:

- Click in the dial's track (the dark area within it) to set the value to the click position. While still holding, drag up and down to adjust the value.

- Hover over a dial and use the scroll wheel to fine-adjust the position. At slow speeds, each click of the scroll wheel corresponds to the smallest currently visible increment of the dial. Scrolling faster accelerates the change.

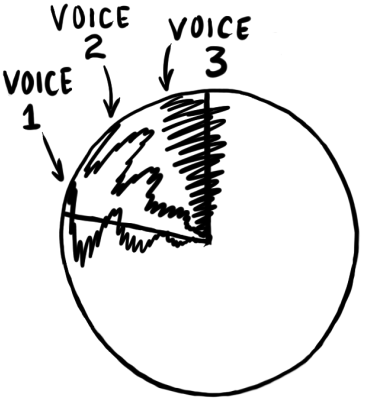
- Click and drag vertically on a dial outside the track area to adjust the dial from the current position.
- Double-click or command-click a dial to return it to its default value.

Holding down the shift key before any of these motions are done will modify the motion to be a fine adjustment. This allows particular values to be set precisely.

Dials as Displays

Each dial controls a signal that can be modulated, and shows the most recent sample values of that signal every 1/30th of a second. That means that static or slow-moving modulation will cause the pointer to stay still or move slowly back and forth.

A faster modulation signal will cause the pointer to show a waveform of the dial's position under modulation. Modulation is shown cumulatively, so a dial receiving more than one modulation signal will show the sum of the incoming signals.



The dial's display is just like a classic oscilloscope display, but wrapped around the center of the dial in what are called polar coordinates. Time moves outward from the center of the dial, and every value of the signal is a straight line going outward from the center. So, a constant value creates a straight-line image in the dial.

Whether a MIDI note is being sent to Virta or not, it always calculates as many voices as the ***voices*** dial in the KEY module is set to. When animation is on, each voice is displayed as a separate line in every signal dial. So, if you set the number of voices to four, play a four-voice chord and send just the steady pitch output of KEY to OSC 1 OR 2 pitch, you will see four straight lines in the pitch dial.

If you send more complex modulations to the pitch dial, you will see multiple scribbly lines, all animated.

Detents

Some dials, such as the OSC 1 & 2 pitch, have *detents.* Detents are useful default positions. For example, the pitch knob has a detent at an A note in each octave (110Hz, 220 Hz, 440 Hz...) to keep the oscillator tuned to MIDI notes. Normal use of these dials makes them stop only on the detents. By shift-clicking a dial with detents, or holding down shift and dragging it, you can adjust it to any position in between the detents.

Numeric Displays

All of Virta's dials show their current value both in the (often changing!) pointer position, and in a numeric display below each control. The numeric display does not show the modulated value, only the center value that you have set on the dial itself. The numeric displays are not directly editable, so just get that crazy idea out of your head.

We tried it the other way, and all those flashing numbers were a bit much.

The ***Show numbers*** toggle in the global settings on the header lets you turn off all the numerical displays, if you'd rather not see them.

Dial Scales

While many dials are linear (the change per degree from high to low is constant), some dials have logarithmic scales where the change is much larger as the value gets higher. This was done in cases in which a logarithmic scale matches the changes you perceive better than a linear one, as in oscillator pitch, for example.

In a logarithmic scale, equal movements of the mouse in different positions on the dial will produce differently-sized changes. For example, 55, 110, 220 and 440 Hertz are all equally spaced apart on the *rate* dial in the oscillator modules.

Using Buttons and Switches

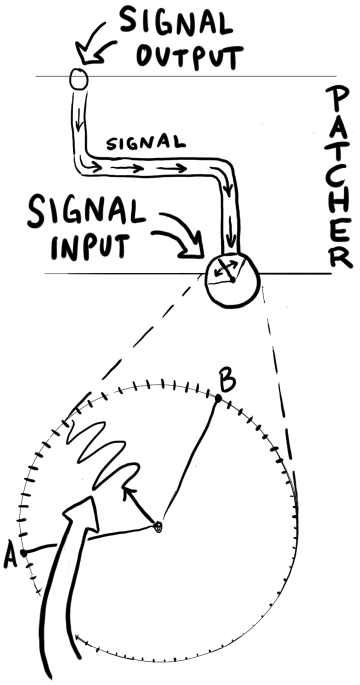
There's not a lot to say here, only that switches need only be clicked to be toggled (you'll see the little dark switch move back and forth) and the same goes for the buttons. Dark is off, bright is on.

Using the Patcher

The patcher is the *grand connector.* With it, you'll bring together the tools Virta offers, to do really fun stuff. The patcher is both the place from which much of the joy of working with Virta springs, and the part of Virta most likely to confuse you at first.

The Patcher is the large dark central area surrounded by all the modules. It lets you patch signals from the outputs of modules to the inputs of modules by drawing patch cords. Each patch cord has an arrow on it that shows which way the signal is flowing. Note that though signals tend to flow down, from ENVELOPE to GATE, for example, this isn't always the case, because inputs can be found on both the top and the bottom of the patcher.

There are no signals underneath the patcher that can flow up, but signals from above the patcher can go to other modules on top. And remember, modulation and audio signals are both the same thing, just made up of different frequencies, so it's perfectly fine to experiment by connecting any output to any input.



Some signals are *bipolar*, meaning they can have negative as well as positive values. Negative signals light up the outputs just like positive signals. In other words, the absolute value of the signal controls the output brightness.

Signal Outputs

These are the tiny circles on the edge of the Patcher; the places from which all patch cords start. They light up to show the current value of the signal.

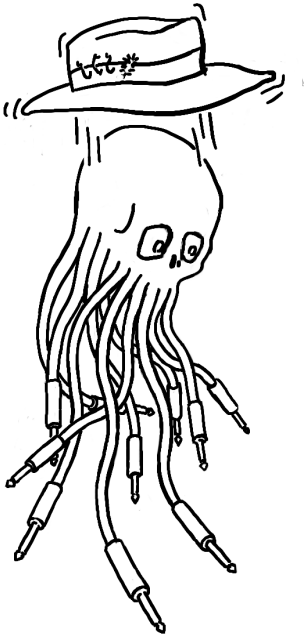
You can use the LFO to see this, even without using any patch cables, as follows: turn the *rate* dial on LFO to 1.0. Double-clicking the dial will do the same thing, because 1.0 is the dial's default value. Now, turn the *level* dial up towards 1.0. You can see the LFO signal output lights pulsing more and more brightly.

Signal Inputs and Modulation

These are the small dials bordering the Patcher; the places where all patch cords end. Each signal input connects to just one dial. When you connect a varying signal to an input, it modulates the dial's signal just as if you were moving the dial itself, but possibly at much faster rates.

Signal inputs are also knobs that let you adjust the amount of modulation applied to the dial. They do not display incoming signals themselves, because you can always see the effect in the dial. Some inputs are bipolar, meaning the value by which they multiply the signal can be either positive or negative. Like the bigger dials, each signal input dial has a default value, and returns to this value when double-clicked.

For example, the exponential pitch inputs to OSC 1 & 2 have a default value that corresponds to standard tuning when the pitch output from the KEY module is connected. Changing this input dial makes nice music into weird tones very quickly. But by double-clicking to restore the default value, normalcy can be quickly restored if desired.



Envelope Trigger Input

Virta's ENVELOPE module can accept *trigger* signals, and it does so through a triangle-shaped input labeled "trig." You patch signals into this input just as described above. The big difference is that there is no dial to set the level of trigger input—a signal either triggers the intended response, or it doesn't. For this reason, you'll want to make sure that the signals you patch into the trigger input are appropriately "triggery," whether they're purpose-built trigger signals like the ***gate*** output of the KEY or AUDIO module, or a particularly spiky audio or control signal.

Patching

To make a patch cord, drag from an output to an input. As you drag, you'll see a glowing line with an arrow at the end stretch from one to the other. This shows the new connection you are making. This is a pretty nifty thing, since such routing does not involve deciphering menus or matrixes of things you can't see—you only need to look at what's on the screen, which is everything. This ease of use is intended to keep Virta feeling like an instrument; something you can grab, pull, and mess with fluidly.

You can make patch connections while holding a note down, and they will affect the currently playing note just as patching a hardware modular would. By holding a note and touching a patch cord end to various signal outputs, you can even get intermittent glitchy sounds that are reminiscent of playing with a live electrical circuit, or *circuit bending*.

Multitudes Within Multitudes

Almost every module panel in Virta's interface is really a controller for as many copies of the module as there are voices. And each voice has its own internal patcher. When a patch cord is made using the patcher UI, it is made simultaneously in the patcher within each voice. For example, if you connect the output of the LFO to OSC 1 pitch, you are connecting LFO of voice 1 to *pitch* of voice 1, LFO of voice 2 to *pitch* of voice 2, and so on. The KEY module is the exception: it is more like one module with a signal output for each voice. When a note is played repeatedly, the KEY module sends the note signal to each voice's patcher in turn to create polyphony.

Since they are controlled by the common patcher UI, and one set of dials, the patch created for each voice is identical. But the signals that flow through each voice's patch can be very different. Thus, each voice is separately controllable, in timbre, modulation, and all of its parameters.

A patch cord always takes on the color of the module it is coming from. This helps you see at a glance what is going where. To modify a patch cord after it's made, first you must select it. To select a single cord, just click directly on it. When multiple cords are running over the point you click, clicking repeatedly will rotate through all the cords at that point. You can also select a group of cords in the patcher by clicking on an empty part of the patcher, then dragging over multiple cords.

Removing or Repatching a Cable

When a cord is selected, its *handles* are visible. Handles appear as circles at either end of the cord—you can drag them to move the ends. If multiple cords are selected starting or ending at the same place, clicking the handle there will move all of the selected cords.

You can also delete a patch cord or multiple selected cords by dragging either end to a place with no input or output. An X will appear instead of the handle at the end, and POOF. It's gone. Again, the changes happen in real time for any currently held notes.

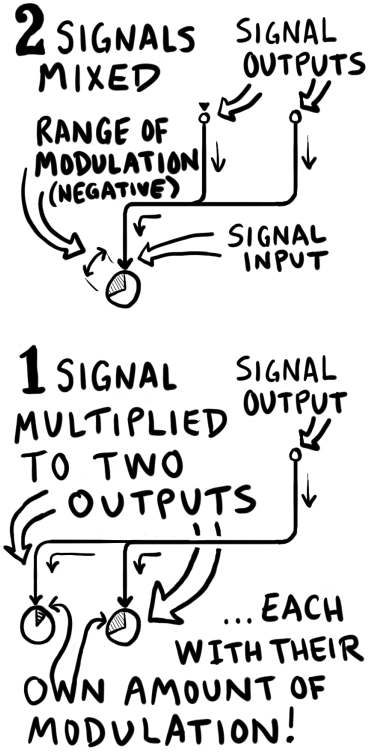
Mixing and Multing

If multiple cables go to a single input, the signals are *mixed* together. The sum of all these signals is then multiplied by the input dial value. If multiple cables go from one output to more than one destination, the signal has been multiplied, or *multed.*

That's not terribly important information, but it's good to have your terminology straight, especially if you move on to other modular instruments.

Unipolar vs. Bipolar

Some output signals, such as the envelope outputs, send only positive values, and are *unipolar.* Others, like the *pitch* output on the KEY module, and the LFO, swing both positive and negative. These are *bipolar.* Negative and positive signal values follow all of the same rules that real numbers do in algebra.



For example, if a negative-valued signal is multiplied by a negative signal input dial, its effect on the modulation will be positive.

Default Signal Routing

We've seen how easy it is to patch Virta's modules together, but it's important to know that some connections in Virta are pre-made for you.

The FORMANT → GATE → DELAY → OUTPUT signal path, consisting of all the modules below the patcher, is pre-routed. The dials between those modules let you set the wet/dry balance of the FORMANT and DELAY processes.

The pre-patched modules below the patcher can be called the processing modules. While the modules above the patcher generate signals, the ones below are where the sounds you hear are shaped. Of course, this being a patchable instrument, there are exceptions to this rule. How you proceed is up to you! The *Aux in* input in the OUTPUT module gives you a chance to send a signal directly from a top-row module to the output, bypassing all bottom-row processing.

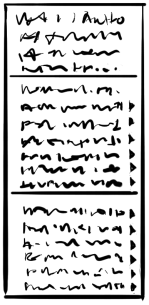
3 Virta: Module by Module

This chapter will take you on a more detailed tour of the various modules that compose Virta, one by one.



The Header

Apart from reminding you which amazing plug-in you're currently using, Virta's header mainly deals with patch access and management, and user interface options. All the things that don't affect the sound, in other words. The big drop-down menu in the middle displays the current patch name, and lets you select patches from a hierarchical list. The menu is refreshed each time you click on it, so new patches you save or import will show up right away.



The patch menu

The drop-down patch menu has three main sections. The first section holds the Copy, Paste, and Save commands. When you select "Copy to clipboard," the current patch is saved in a text-only format that you can paste into other text documents.

This lets you send a patch to a friend in an email, or post it on a forum, for example. "Paste from clipboard" does the reverse.

"Save as version" lets you quickly save a new version of the currently loaded patch (with whatever tweaks you may have made since loading it) without having to enter a new patch name. The new patch is named after the current patch, followed by a revision number in brackets, incrementing with each new version you save.

"Save" permanently updates the current patch with any parameter changes you've made since loading the patch. This is, by definition, a bit risky, unless you're sure of the changes you've made. In many cases, you'll be safer using "Save as version" or "Save as..." when making incremental changes to an existing patch.

"Save as..." brings up a file chooser from which you can create a new file to save the patch to, or choose an existing one to overwrite. Patches from the Audio Units version of Virta are saved in the .aupreset format. This is a compressed XML format, compatible with Logic, Live and other Audio Units-friendly applications. Patches from the VST version of Virta are saved in the .mlpreset format. This is the same XML format, but uncompressed.

"Revert to saved" returns all parameters in the currently loaded patch to their original, saved settings. You can also activate the Revert to Saved feature by sending MIDI program change 128 to Virta. This can be useful when recording multiple takes of Virta dial-twiddling as audio in Ableton Live. In the Clip View for the MIDI clip you're working with, set the "Program" parameter to 128. Each time that MIDI clip is launched (with its launch button or a stop-and-start of the transport), Live sends program change 128 to Virta, reverting the patch to its saved value. This gets you back to a consistent starting point for the next recording pass.



Below the commands are all of Virta's patches in two more sections: the so-called "factory presets," in directories all starting with "Virta," followed by storage space for your own personal patches. Some user presets, contributions from fellow Virta users, are installed here by default.

Thanks, wonderful sound-crazed Virta users, for all your contributions!

Presets are all stored on your hard disk in the right place for user data on your operating system of choice. For simplicity, factory presets are stored in the same place as your own patches. In the unlikely (but perfectly valid) scenario that you have multiple people with accounts on the same computer all using Virta, each person can have a copy of the factory presets along with their own patches in their home directory.

On Mac OS, the user directory is in *(your home directory)* / Library / Audio / Presets. On Windows 7, it's in *(your home directory)* / UserData. If you're trying to copy your preset files using Windows Explorer, be aware that even though it's the recommended path for user data, Windows makes this directory invisible by default. Likewise, Mac OS 10.7 Lion and more recent versions hide the Library folder. You can navigate to the Library folder in the finder by choosing the "Go" menu and holding down the option key.

Selecting patches via MIDI

If you'd like to be able to load patches by sending MIDI program changes to Virta, create a folder titled "MIDI Programs" (note the capitalization and the space between words) in one of the following locations, depending on your platform:

- Mac OS: /Library/Audio/Presets/Madrona Labs/Virta
- Windows: (Your home directory)/UserData

Copy the patches you'd like to access with MIDI program changes into the "MIDI Programs" folder you've created. The folder is scanned by Virta on startup, and the presets in it are assigned numbers, in alphabetical order. To rearrange the programs, give them new names so they are in a different alphabetical order. Send a program change to Virta that corresponds to your chosen patch, and Virta will dutifully switch to that patch.

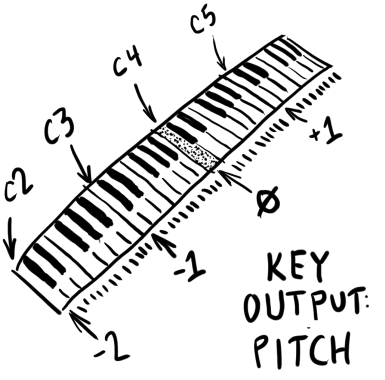
If you look at the MIDI Programs folder in Virta's preset menu, you will see each preset listed, followed by its MIDI program change number.

Global Settings

Click the gear-shaped button on the right of the header to access the global settings. ***Show numbers*** lets you toggle the numeric displays that sit next to each dial on or off. ***Animate dials*** lets you toggle the animated scope-like displays inside the dials on or off. ***Reset editor size*** returns Virta's window to its default size. The ***Input protocol*** menu lets you specify whether you wish to use MIDI or OSC to control Virta. The ***OSC port offset*** menu lets you specify the OSC port you wish to use for control. For example, if your base OSC port is 6000, and you set the offset to 5, Virta looks for OSC control signals on port 6005.

Version and registration

The right top corner shows the version of Virta you are running, as well as your registration info. When you purchase a copy of Virta for yourself, we encode your name and account information into it. This shows to you and the world that you are supporting what we do—from our end it means we have agreed to help you out with Virta if any problems arise, and to maintain and improve it.



KEY

The KEY module receives all the MIDI data you send Virta's way, and turns it into useful control signals that you can route to other modules with the Patcher. If your MIDI controller was a basket of fresh fruit, you could think of KEY as a robot that bakes pies that the other modules long to eat.

Tuning menu

The menu on the top selects the tuning table that Virta uses to map incoming MIDI notes to specific frequencies. *12-equal,* the default tuning, is short for 12-tone equal temperament. It is the basis for most modern Western music, but there are around a hundred others to try, included with Virta. There are too many scales to describe here, but if you open up the .scl file for a scale you're interested in, you can read it as text, and often find a bit more information that can lead to an article on the subject. As a start, we've selected some of the public-domain scales from the Scala archive and sorted them into the tuning menu according to what musical culture they're from.

You can also add tuning files in .scl format to the scales directory yourself, or make your own using the free software Scala, available at http://www.huygens-fokker.org/scala/.

Voice controls

- ***voice:*** This control lets you set the number of voices of available polyphony, from one to four. Monophonic, duophonic, triphonic, quadrophonic. The choice is yours.

- ***bend:*** This control lets you set the amount that the ***pitch*** output varies when MIDI pitch bend messages are received. It is calibrated in semitones from zero to 24. Yes, 24-count-em-twenty-four semitones, which really means a range of 48 (+- 24), or four octaves. Can you handle such power? No? OK, then set it to 7, or something.

- *unison:* This button lets you toggle Unison mode on or off. Unison mode combines all four voices into one monophonic voice, which can make for some very big sounds. If multiple oscillators at exactly the same pitch are added together, the result can sound quieter than a single oscillator because the waveforms cancel each other out. This is hardly ever what anyone wants, so Virta's sound engine applies a small random frequency drift to the pitch of each oscillator to maintain a nice, big sound.

- *glide:* This dial lets you bring a little or a lot of portamento (pitch glide between notes) to the party. Set it to the amount of time (in seconds) you wish Virta to take when sliding between notes.

  item*mod cc#:* This control lets you select which MIDI continuous controller signal to output through the Mod output. When set to 1, it will use the Mod wheel. The subsequent two MIDI CCs above the number you choose will drive the +1 and +2 outputs.

Outputs

- *pitch:* This output turns incoming MIDI notes into a pitch signal Virta can use. When MIDI note A3 is played, the pitch signal output is 0. This has the same result on a patcher input as when nothing is connected. A4, an octave higher, outputs the value 1, and A2 outputs -1. Another 1 is added or subtracted for each octave up or down. This scaling was chosen so that keyboard input maps naturally to all the various control signals.

It's like the 1.0 volt per octave standard of some modular hardware, but there are no actual volts involved. So we can call this just 1.0 per octave.

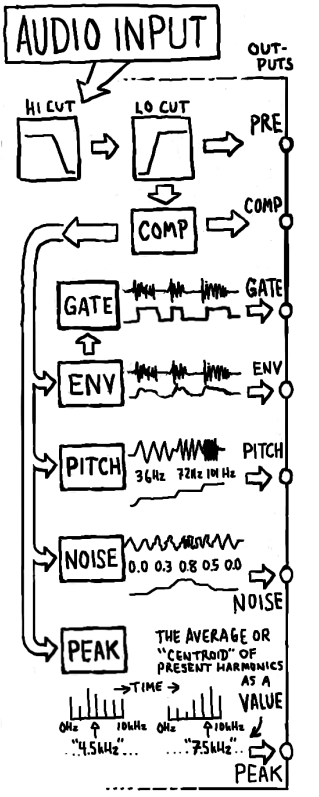
In the patcher, any input dials that control pitches (such as OSC 1 & 2 *pitch* and GATE *level*) are all calibrated so that when you connect a pitch input and set the default scaling (double-click), they will track the same frequencies or intervals according to the 1.0 per octave standard.

- *vel:* This output sends a signal proportional to the velocity of incoming MIDI notes. This signal maintains its value after each key is released, allowing neat things like whacking on a drum pad at different levels of velocity to set filter cutoff over time, and such.
- *vox:* This output sends a signal proportional to the number of each voice: 0.0 for voice 1, 1.0 for voice 2, and so on. This can be used to quickly make changes to the patch that are different for each voice, such as panning all the voices across the stereo field, or setting each lfo to a different rate.
- *after:* This output sends polyphonic aftertouch data for each key, added to the channel aftertouch value. There's nothing like routing aftertouch to a few parameters, and discovering a new dimension of control over notes you're already holding down! Really, do it. It's awesome.
- *mod:* This output sends a continuously-variable control signal, set by whatever MIDI CC (continuous control) you've specified in the *mod cc#* dial above.
- *+1 and +2:* These outputs send continuously-variable control signals, set by the two subsequent MIDI CCs (continuous controls) above the value you've specified in the *mod cc#* dial. For example, if the *mod cc#* dial is set to 10, the +1 output responds to CC# 11, and the +2 output responds to CC# 12.

Channel aftertouch sends one value for the MIDI channel, and polyphonic (or poly key pressure) sends a different value for each key. Very few keyboards have true polyphonic aftertouch, so we decided these two kinds of aftertouch could share a signal output. If you don't have a keyboard controller with aftertouch, you can still use the output in Virta by sending messages from a MIDI knob or fader controller.

AUDIO

The AUDIO module is the magical place where incoming audio is analyzed in real time, producing a wealth of control signals you can use to make the rest of Virta come to life.



Controls

- *input:* This dial sets the level of the input signal before it reaches the Compressor.

- *thresh:* This dial sets the amount of compression applied to the input signal. Gain is compensated automatically as you add more compression, so no output gain control is needed. The effect starts out rather gentle, and develops more and more attitude as you crank it.

- *lo cut / hi cut:* These dials set the corner frequencies of a pair of 24 dB/octave filters that act on the input signal. The filters cut frequencies quite sharply, and are useful for isolating certain areas of frequency in the input signal. Cut frequencies starting from the bass-end with *lo cut* and from the treble-end with *hi cut*.

- *trig:* This dial sets the trigger threshold for the *gate* output. When the input signal exceeds the threshold you set, the *gate* output flips to "1" (signifying that a signal has tripped the trigger) and stays there until the signal drops below the threshold again.

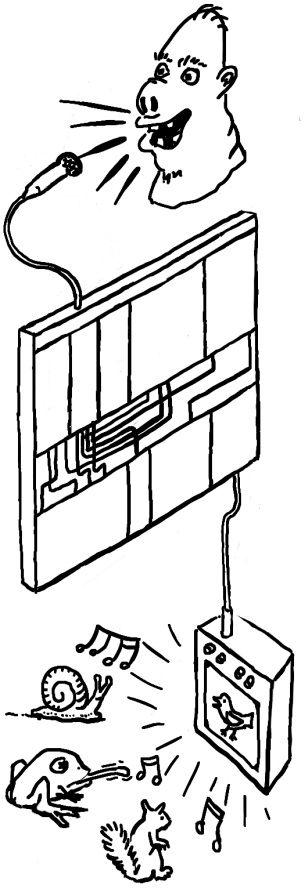
- *quant:* Enable this toggle to quantize the output of the pitch detector to the musical scale you've chosen in the KEY module. Disable it to allow continuous (non-stepped) pitch output.

Outputs

- *pre:* This output provides the raw audio input signal, for use in patching to other modules in Virta.
- *comp:* This output provides the audio input signal after it's been run through the compressor and hi/lo-cut filters.
- *gate:* This output provides trigger signals derived from the audio input signal, as directed by the *thresh* control.
- *env:* This output gives you an envelope-style control signal that follows the shape of incoming audio (after compression and filtering). You might send this signal to the GATE module to link the volume of a synth patch to the amplitude of incoming audio.
- *pitch:* This output gives you the pitch control signal derived from audio input by the pitch detector. You can send this signal to the oscillators to control the pitch of synth patches, or to any input you like, for fun real-time control.
- *noise:* This is the output of the noise detector, which, in a nutshell, listens to the input signal and determines how "noisy" it is. A sine wave has a noisiness of 0, whereas white noise would have a noisiness of 1. Most sounds you introduce will fall somewhere between those values. Patch this to various things for another level of timbre-based real-time control.
- *peak:* This output gives you a value that describes the overall brightness of your input signal, effectively tracking the highest center of frequency content. The brighter your signal is, the higher this value will be.

Also known as the "spectral centroid" in computer-music-speak.

LFO



The LFO module is a Low-Frequency Oscillator—one of the canonical modulation sources in synthesis. Practically every synth has an LFO. Those that don't tend to mope around a lot, thinking of all that could have been.

The drawing on top shows the currently chosen LFO shape. The menu below it gives you access to all available shapes. Currently, our list of noble shapes includes:

- *sine one:* A sine wave, the most harmonically pure of all shapes. This sine wave varies from zero to one, which is good for modulating another thing either up or down.
- *sine two:* Another sine wave, this time centered around zero. As a modulator, this wave is useful for making FM synthesis. Try it with the linear pitch inputs of the oscillators.
- *triangle:* A very triangular wave. It's all business.
- *square:* Either zero or one, a simple proposition.
- *saw up:* Up-going sawtooth wave.
- *saw down:* Down-going sawtooth wave.
- *noise:* Noise usually means randomness. But this noise is a little different. It simply loops a short *pseudo*–random shape, the exact shape, in fact, that you see in the drawing. The advantage of it not being truly random is that it will repeat itself reliably every time you play back your composition. When set to a low rate, the wavering value is not likely to be noticed as a repetition.

- *blip:* We call this one the blip. It's the shape that the envelope of a vibrating object makes, more or less, after it is hit—in physics terms, an *exponential decay.* It's useful as a kind of third envelope.

Controls

- *rate:* This dial sets the oscillation rate of the LFO. Set it low to go slow, high to go fast, or in the middle to just *cruuuuise*.

- *sync:* Enable this toggle to set LFO speed as a rhythmic subdivision of host tempo. Disable it to let the LFO fly free.

- *level:* This dial lets you set the output level of the LFO signal.

OSC 1 & 2

OSC 1 & 2 are formidable sound sources, providing the raw waveforms that then go on to be processed by the FORMANT module and other processors. Their inputs let you control the pitch and timbre of your synth patches in real time, with whatever control signals (from audio or MIDI or other modulators) you prefer. Each oscillator can be set to one of two modes: Classic mode makes juicy analog waveforms, and VOSIM mode makes even juicier voice-inspired sounds. Some control functions differ depending on which mode is selected. Most controls feature at least one Patcher signal input, so get modulating!

Controls

- *pitch:* This dial controls the resting pitch of the oscillator— A.K.A. the pitch produced when you plug in a signal from the Key module's Pitch output and hit MIDI key A3. Unlike other dials, the pitch dial has two Patcher inputs: one exponential and one linear. The exponential input is calibrated to match the one per octave standard of Virta's mod signals. The linear input has a much wider range and can be modulated for fast "thwips," inharmonic sounds and other effects.

- *noise:* (Classic mode) This dial fades the oscillator's output between your choice of waveform and noise that is bandpass filtered to match the sine wave frequency.

- *formants:* (Vosim mode) This dial lets you specify the number of formants used in the vowel synthesis. Higher settings introduce higher harmonics.

- *shape:* (Classic Mode) This dial lets you control an antialiased waveshaper that fades between a sine in the center, a square on the left, and a saw on the right. Patching LFO or Sequencer signals to this control can create pleasing PWM-like effects.

- *peak:* (Vosim Mode) This dial lets you control the formant peak of the vowel synthesis algorithm. This dial describes a trajectory through a two-dimensional space of vowels from "OH" to "'AH" to "EE." Controlling this parameter with the *peak* output from the AUDIO module can help to make a kind of synthesized speech. At the top end of the *peak* dial, noise is faded in. If the oscillator's signal is intended to be used as a carrier for the FORMANTS module, that added noise makes consonants clearer, and speech more intelligible.

- *level:* This dial lets you set the output level of each oscillator.

ENVELOPE

This module is an envelope generator with a standard ADSR control layout. The display shows the current shape of the envelope.

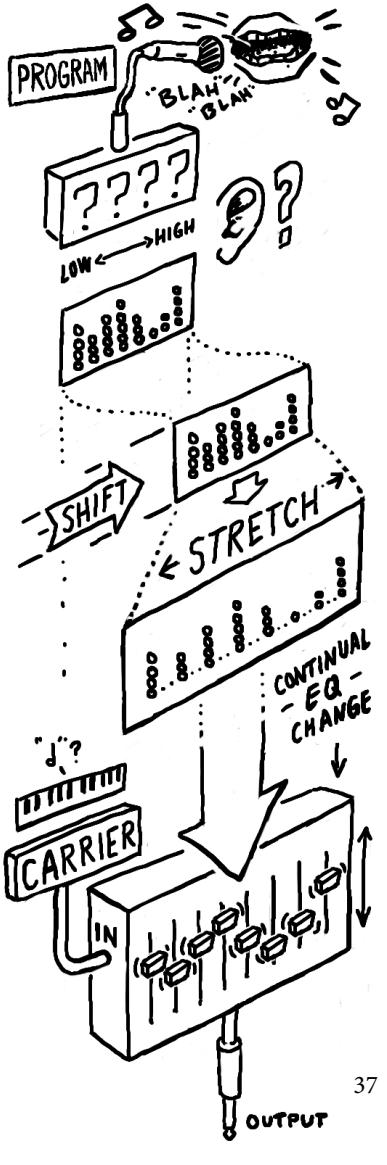
ENVELOPE is an ADSR generator, so your old pals Attack, Decay, Sustain and Release each have their own control. Each control has an input for control signals.

Envelope has a trigger input that lets you trigger it with signals such as the *gate* output from the AUDIO or KEY module... Or something trickier (so long as it's "trigger-y").

FORMANTS

The FORMANTS module is a big part of what makes Virta's vocal-ish processing special. Using one of four vocoder algorithms, it analyses a sound or signal (which enters through the "program" input), and applies multi-band filtering based on that analysis to another signal (which enters through the "carrier" input). In this way, it applies the timbral essence of one signal to another, over time.

If you're doing audio-controlled synth stuff, or a vocoded synth with synth pitches coming from MIDI/OSC you'll likely patch one or both oscillators into the *carrier* input, and your audio (from the *pre* or *comp* outputs of the AUDIO module) into the *program* input. That said, that's not the only way to do things.



If you're not planning on using FORMANTS at all in a given patch, you'll probably still want to patch your audio or oscillator signals into the *program* input, so that it will flow into the GATE and DELAY modules, to the output. In this instance, you'll want to turn up the *pgm* dial between FORMANTS and GATE, so that the audio is audible.

Some folks prefer it that way.

FORMANTS comes with a bunch of different filtering modes, all designed to do wonderful things while under the spell of vocal sounds. That said, you can pipe any sound you like into Virta, and it will do your bidding dutifully, often with unexpected results.

Formant Displays

The top bar-graph display shows you the detected frequency content of the program signal, spread across as many bands as you've specified. The display below it shows the actual position of the formant filters, which can differ from the detected positions, once you've put your grubby mitts all over the *shift* and *stretch* controls.

Voices to the Unvoiced

Because most carrier signals (even brighter waveforms like sawtooth or square) don't have enough high-frequency information to accurately synthesize the "unvoiced" parts of speech or singing (or the atonal, textural bits of other sounds), classic analog vocoders of yore often had what was called an "unvoiced detector." The idea is to detect when an input program is a consonant, or any sound with high frequency energy that requires a noisy carrier to reproduce. When such an input is detected, a noise generator, or sometimes a little bit of the program input itself, is injected into the carrier input.

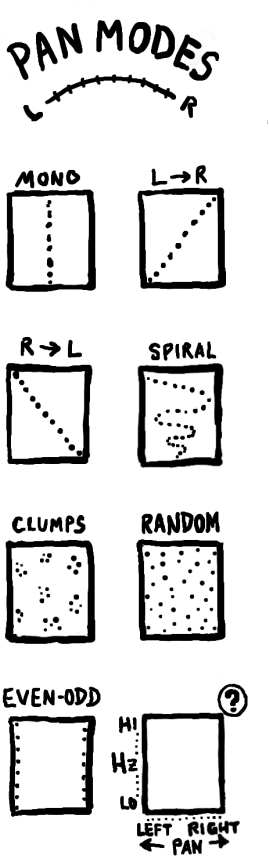
In Virta, we provide other tools which you can take your pick of, outside of the FORMANTS module, for handling this scenario. The *noise* and *peak* outputs of the KEY module provide different but complementary ways of detecting "unvoiced-ness." These signals can be sent to various destinations in order to make the carrier you send to the FORMANTS module more noisy. In particular, try sending the *peak* output to the *peak* input of the oscillator in VOSIM mode! They are made for each other.

Controls

- *shift:* This dial lets you shift the detected vocal formants up or down in frequency, before they are applied to the carrier signal.
- *stretch:* This dial lets you stretch the relationship between the detected formants like taffy before they are applied to the carrier signal. You can bring them closer together, or further apart, or even invert the relationships entirely (low on top, high on bottom).
- *q:* This dial sets the width of the formant filters. At lower settings, each filter passes a fairly wide band of frequencies. Higher settings pass narrower ranges of frequencies, and ring out over time, approaching the sound of oscillators.
- *quant:* This toggle turns quantization of formant frequencies on or off. When on, each frequency snaps to the closest note in the musical scale selected in the KEY module, instead of moving continuously.
- *mode:* This selector lets you choose the number of filters that FORMANTS uses. Choose from classic vocoder-like modes (from 8-32 bands), and the unabashedly digital FFT-based 256-band mode.

Vocoder 8-32 also offer a *quant* switch, that locks the formants to frequencies that correspond to the current scale selection in KEY.

- *pan:* This selector sets the way in which individual formants are spread across the stereo field. *Mono* is just that, everything is in the center. *L->R* places formants low-to-high from left to right across the stereo field. *R->L* does the opposite. *Spiral* places formants in an overlapping back-and-forth pattern across the stereo field. *Clumps* spaces groups of formants across the field. *Random* places formants at random. *Even-Odd* stacks even harmonics toward the left, and odd toward the right.



GATE

The GATE module is a dynamic volume control, akin to the VCAs (Voltage Controlled Amplifiers) found in modular synths. Its input comes from the FORMANTS module, with the wet/dry ratio between the raw carrier signal and the filtered output blended according to the small knobs in between the two modules. Its output is sent to the DELAY module. You send GATE control signals, typically envelopes, and it nicely increases and decreases the amount of input signals passed to the output. The control signals you send flow through a vactrol emulation, this time with a settable decay, which opens up a world of cool, percussive envelopes.

The GATE is an important tool used to sculpt the dynamic profile of Virta's sound, and it's pretty magical beyond that. That magic comes into play when you flip it into LPG (Low Pass Gate) mode, and commence synth-bongoing 'til dawn.

Controls

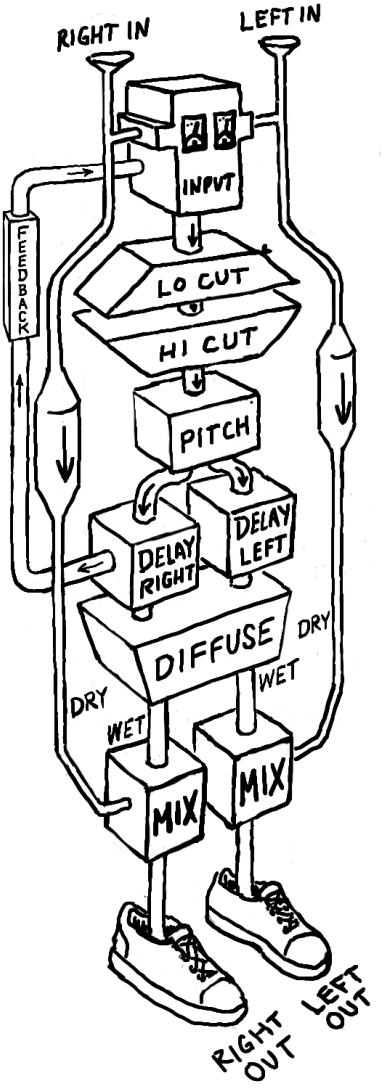
- *level:* This dial sets the static level of the Gate module's level attenuator. Signals from the *level* patcher input modulate the level, as well. For normal, keyboard-like playing, you'll probably keep this control at zero, so only incoming envelope signals triggered by key presses affect the sound's volume. For drones or reverse-enveloped sounds, try raising it higher.

- *lopass:* This toggle turns on low-pass gate (LPG) mode. In LPG mode, the gate's gain-changing cell is replaced by a low-pass filter, the frequency of which is modulated by the level signal. The modulation afforded by the vactrol emulation makes percussive envelopes with a very particular sonic signature.

- *decay:* Sets the decay constant of the vactrol algorithm. At low *decay* settings, the GATE will follow incoming mod signals very snappily. At higher settings, the decay of the vactrol rings out more and more.

DELAY

The DELAY module is a sprightly delay with a pitch-shifter and diffusor in its feedback loop. This means that delayed signals can be pitch-shifted and diffused again each time they repeat, with skyward-climbing or floorward-lurching results. This effect can be quite subtle, or can be the Thor's hammer you need to really smash things to bits.



Controls

- *input:* This dial lets you set the input level to the delay effect.

- *pitch:* This dial lets you set the amount of pitch-shift applied to the delayed signal, from 0 (no shift) to plus or minus an octave.

- *diffuse:* This dial controls the amount of diffusion applied to the delayed signal. Lower settings give you cleaner delays, whereas higher settings smudge the lens a bit, for more organic delay decay (as in an analog or tape delay unit).

- *lo-cut / hi-cut* These dials set the corner frequencies of a pair of shelving filters that act on the delayed signal. Cut frequencies starting from the bass-end with *lo cut* and from the treble-end with *hi cut*.

- *feedback:* This dial lets you specify the amount of delayed signal to feed back into the input of the delay. The higher the setting, the more repeats you'll hear.

- *delay (ms):* This slider sets the length of the delay line. Smaller values give shorter delays, and higher values, longer.

- *L/R ratio:* This dial tilts the delay ratio for the left and right channels. Lower settings shorten the left delay time and lengthen the right. Higher settings do the opposite. At the center, the delays are of equal length in each channel. This control comes in handy for ping-pong stereo delay effects.

Keep in mind that because the pitch shifter and diffusor processes live in the feedback loop, the signal is shifted and diffused again each time it repeats. Blast off!

OUTPUT

The Output module lets you mix and blend your lovely new signals before they're sent out into the cruel, cruel world. Here, you'll find an *Aux In* that lets you patch signals directly to the output, bypassing all the processing in the lower row of modules. You'll also find level and pan controls, and a nifty little oscilloscope that shows you the waveforms you're making. All controls have signal inputs on the patcher.



Controls

- *aux in:* This dial sets the output level of the signal patched to the *Aux In* input. If nothing is patched there, this dial has no effect. An accompanying *Pan* dial lets you set left-right balance for this signal.

- *delay in:* This dial sets the output level of the signal arriving from the DELAY module (and thus, all the processors in the bottom row of modules). An accompanying *Pan* dial lets you set left-right balance for this signal.

After the output

Well, that's a pretty thorough description of all the modules and how signals flow through them to make sound. But what happens to your sound when it leaves Virta? This is an important thing to consider. Virta has been designed to give you the purest possible sound from its oscillators, if you choose, and an extremely wide dynamic range.

Because Virta has a nice collection of vocoders at its heart, and because the output levels of vocoders can vary so much depending on the carrier and program signals, we have built a limiter into Virta after the output mixer. The effects of this limiter can be seen in the waveform display in the OUTPUT module. And when it's working, you will see the limiter light go on, at the bottom of the waveform display. When the light is off, the limiter is not affecting the signal. When it is on, the limiter is engaged, preventing the signal output from going over 0dB.

There's no harm in running your Virta patch with the output light stuck on, if you like the resulting sound. We don't judge! But if a sound with high dynamic range is your goal, you will want to back off the various levels we've discussed when you see that flashing light.

Finally, a high-quality audio interface is really a must to get the best sounds out of Virta. Part of the reason software synthesizers can sound like they are missing something in comparison to their analog brethren is that computers do not usually come equipped with good audio hardware. Turning a stream of digital data, in a noisy electrical environment, into accurate and stable analog voltages is a very demanding task, and it's a safe bet that the cheapest device you can find will not sound good. The good news is, the price of truly listenable interfaces is getting lower all the time. The landscape is constantly changing, but as of this writing we can recommend digital to analog converters from Metric Halo, Apogee, RME, and MOTU. As always, the only enduring advice is: use your ears and your own good judgement.

A Frequently asked questions

Why "Virta?"

It's the Finnish word for "current," as in both electrical power and a water stream.

Where is it? I installed it but I can't find it in the Start Menu / Dock / Applications.

Virta is a VST and Audio Units format plugin. To run it, you need a VST or AU host on your computer. Please see the Introduction to this manual for more info, and try asking on our web forums if you need advice finding a host to use.

Is Virta supposed to sound like this?

Probably, unless you hear an abrasive series of glitches. Here's a good way to check that Virta is functioning well: select the "default" patch from the factory patches section of the patch menu. Change the ***voices*** control if needed to get all four voices running.

Now, turn the *level* dial in the GATE module up a little bit. You should hear a mellow, slowly shifting drone. If there are any glitches in the audio, they will be readily apparent.

I hear the glitches, how do I get rid of them?

The most common thing that needs adjustment is buffer size. Your host gives you a control somewhere over the size of the small buffers it fills up with calculations, over and over, to generate a steady stream of sound. If this buffer is too small, the calculation takes much longer, and even the fastest computer won't be able to keep up. Try turning the buffer size up to some number greater than 256. This should let Virta run as fast as possible. In Ableton Live, the buffer size control is under "Preferences... / Audio / Buffer Size." For other hosts, it's probably something similar: please check your host's manual for details.

If the buffer size made no difference, it's possible that your computer is not fast enough to run all of Virta's voices. You can try turning the *voices* control down to 1, and turning up the audio again on the default patch. If this helps, then it's almost certainly the case that CPU power is the issue. You can try adding voices one by one to hear where the problems come up.

If you are running the 64-bit version of Virta in a 64-bit VST or AU host, you can expect to get around a 10% performance boost compared to the 32-bit version.

Finally, turning off animations with the *Animate dials* global setting or hiding the Virta interface altogether will increase performance, for those times you are trying to squeeze out that last few percent and get your mixdown to happen.

For reference, as of version 1.0, each voice of Virta takes around 10% of the CPU time on an 2.4 Ghz Intel i5 processor. Since all of the audio processing is constantly running, Virta will take this time whether or not a MIDI note is playing.

Performance is affected by many, many variables including choice of audio interface, drivers, host application and OS version. We can only give guidelines here. To tap into the collective wisdom of Virta users on this topic, visit the ongoing discussions at madronalabs.com.

I bought one license for Virta. Can I use it on my Mac and my PC too?

Yes. Virta's license is very simple, but different from some you may have encountered. One purchase gives you a license for both Mac and Windows. You are restricted to running Virta on one computer per license at any one time. If you want to run the software on more than one computer at a time, you must buy a licensed copy for each computer.

How does your copy protection work?

Virta does not have copy protection. Copy protection always creates hassles for legitimate users. Our approach is different.

What we do is stamp each copy of Virta securely with user data, consisting of your name and a unique ID. This is your own copy of Virta, and you are free to make as many copies as you want. But do so carefully. When you run a copy, it may unobtrusively check to ensure that this data is intact and no other copies with the same user data are running anywhere. Since another copy running somewhere else could stop yours from running, we assume you will be careful about where your watermarked copies go.

We imagine, for example, that you might put a copy on your studio machine as well as your home machine, or on a USB stick to take to a mixdown session.

Can I load presets made in Virta 1.0 in version 1.2 or 1.3?

Yes. Virta presets will always be compatible with future versions, even as we add controls and features.

On the other hand, if you try to load newer presets in an older version of Virta, you will get errors.

I've got a Madrona Labs Soundplane controller. How do I set it up to work with Virta?

Virta detects your Soundplane's presence (provided you've got it plugged in and set up), and automatically switches its control behavior to make full use of the OSC/t3d control data Soundplane provides. Just plug in, and play.

I'm not playing any notes, so why is Virta eating my CPU time?

We designed Virta as a very general-purpose sound-making machine that behaves very much like an analog modular synthesizer. So Virta has free-running oscillators that are updated whenever your DAW is processing audio. Just like on a modular, you can simply turn the ***level*** dial on the GATE up to hear the oscillator, even if no notes are playing.

Virta seems to be stuck on, how can I get it to stop?

Is the ***level*** dial on the GATE module turned up to a nonzero value? That's usually the problem.

If not, maybe there's a stuck note, even though we haven't heard one for a long time. Try turning the ***voices*** control to reset the KEY module.

The decay control on the gate module doesn't seem to have any effect. Why not?

The *decay* controls the time constant of the Vactrol emulation. This is a special kind of low pass filter applied to the level input itself, not the audio. So you probably won't hear it if the signal controlling the envelope already has a long decay. Try setting all the envelope controls to their minimum values to get a very brief tick, patching that into the *level* input, and adjusting the decay control. You should definitely hear a difference then. Or just fire up the "deep bongo" preset.

How do I make Virta's dials change in response to MIDI data?

The KEY module's *mod* and *+1 and +2* outputs turns MIDI continuous controllers into continuous signals, which can then be sent to any destination in the patcher. This is a very flexible way of using MIDI controller data, because you can route and scale it quickly as a signal. The *mod cc#* dial sets the control number sent to the Mod output, and the *+1 and +2* outputs are driven by the two subsequent MIDI CCs above that setting.

Virta does not provide its own interface for changing a dial's position directly from a MIDI controller, often referred to as *MIDI learn*. Most plugin hosts, such as Live, Logic and Numerology, provide good interfaces for MIDI learn that work well with Virta. Please consult the manual for your host for details.

We recognize that some hosts out there lack good MIDI learn features, or the particular ones you want. So, we plan to address this somehow in a future version.

B Index

Ableton Live, 10
AUDIO module, 12, 32
audio-controlled synthesis, 7
aux input, 43

bipolar, 22
buttons, 18

compression, 32
converters, D/A, 44
copy protection, 47

DAW, 9
DAW setup, 10
DELAY module, 14, 41
detents, 17
dials, 15

ENVELOPE module, 13, 37
envelope triggering, 20, 37

FAQ, 45
feature overview, 10
filters, 32, 41, 42
formants, 37
FORMANTS module, 13, 37

GATE module, 13, 40
glide, 30
glitches, 46

header, 11, 25

KEY module, 12, 28

LFO module, 12, 34
LPG, 40

MIDI CCs, 30, 31
MIDI program change, 27
mixing, 22
multing, 22

noise, 33, 36
numeric parameter displays, 17

OSC 1 & 2 modules, 12
oscillators, 35
OUTPUT module, 14, 43

patch menu, 25
Patcher, 13, 18
patching, 20
peak detection, 33, 36, 38
pie, temperature of, 8
pitch bend, 29
plug-in hosts, 9
polyphony, 29
presets, 14

quantization, 32

registration, 28

saving patches, 25
Scala, 29
signal inputs, 19
signal outputs, 19
Soundplane controller, 48
switches, 18

tuning tables, 29

unipolar, 22
unison, 30

VOSIM, 35, 36, 38

