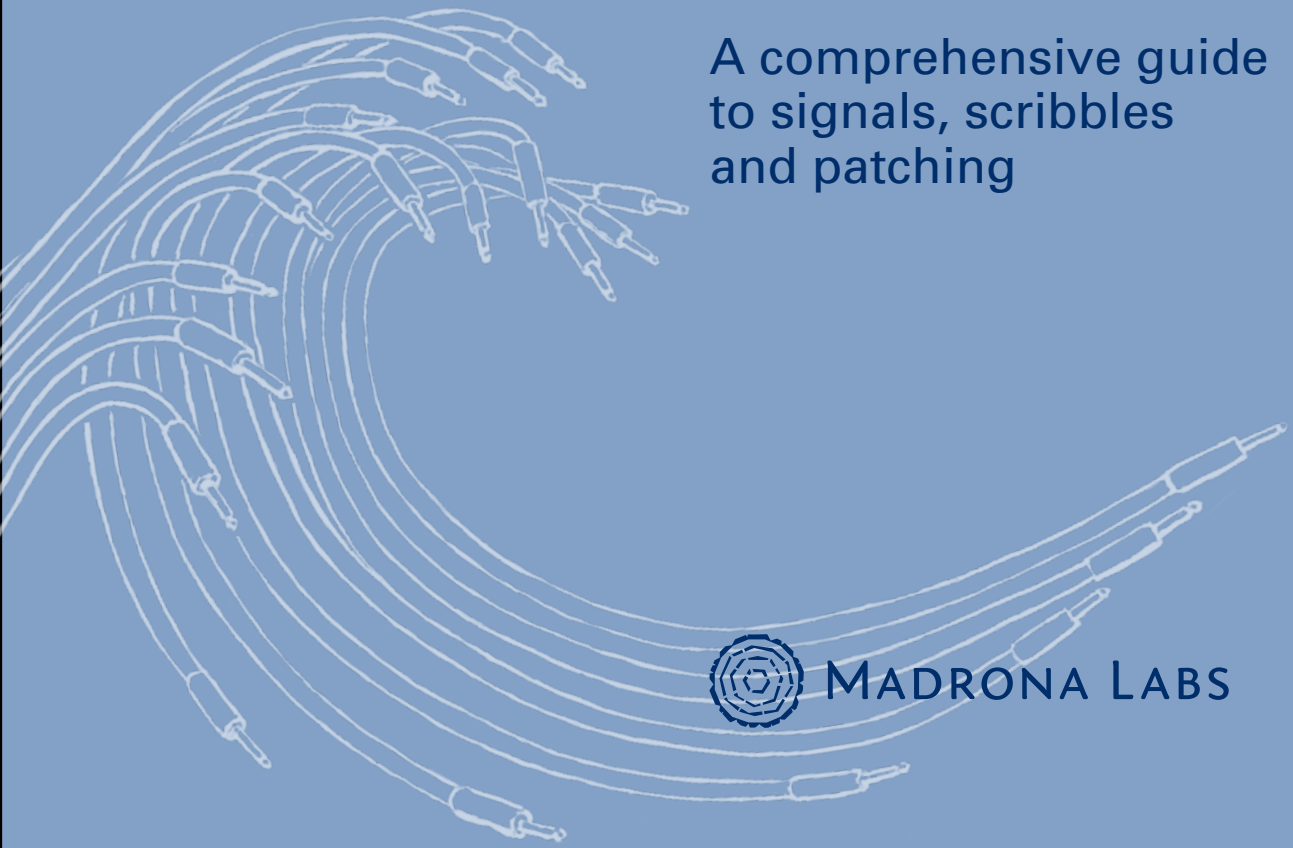
AALTO

A comprehensive guide to signals, scribbles and patching



Madrona Labs

Version 1.8, December 2016. Written by George Cochrane and Randy Jones.

Illustrated by David Chandler.

Typeset in Adobe Minion using the TEX document processing system.

Introduction

What is Aalto? It's tempting to think of Aalto as a mere software synthesizer, yet another sound source, huddling amongst the teeming masses of such instruments that lurk within the menus of your favorite audio program. However, that would be doing it a disservice, for Aalto is, we think, really special. We like to think of it as a carefully crafted box of sonic tools, few enough to learn easily, but flexible enough to combine in surprisingly powerful ways. Of course, it's also just a good everyday instrument, if that's what you want.

Aalto, like many modular synthesizers, comes stocked with oscillators, filters, envelope generators, and goodies such as a waveshaper section and an especially nice one-knob reverb. Aalto's twist (at least, the one we chortle about as we sip vintage armagnac in our secret lair halfway up the Space Needle,) is that thanks to the unique patching interface, making your own sounds with Aalto, even complicated ones, need not be a chore. One might even call it a *joy*.

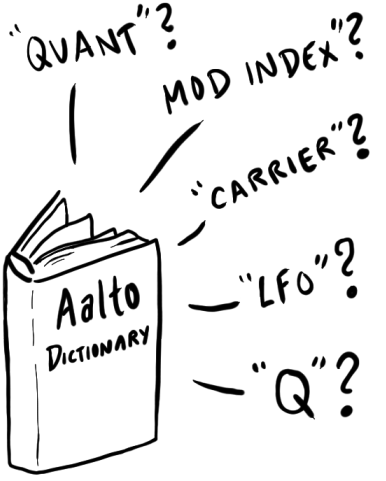
Aalto's interface has four main sections, from top to bottom:

- The *header* section, where the title of the plug-in sits proudly along

with several useful parameters pertaining to patches and the plug-in's interface.

- The *shapes* section, which presents tools for designing a wide variety of sonic shapes over time.
- The *patcher* section, where connections between modules are made and wrangled.
- The *audio* section, where most sound is created and processed.

Perhaps unsurprisingly, most of these descriptions are only loosely bound to reality. However, a manual's intro cannot be 30 pages long! It just wouldn't be right. For more information, read on.



What, you expect me to read?!

Ideally, Aalto would have no manual at all. You'd simply pull up the plug-in, which would fit your knowledge and sensibilities like a well-worn glove, and start making the best sounds of your life. For some of you, this has already happened, and you'll likely never see this manual, let alone natural light, again. For most of us, however, a new instrument requires a little study to get familiar with, so here we are. Luckily, we think you'll find that once you give Aalto a little bit of attention, it does its best to keep you informed and inspired as you work.

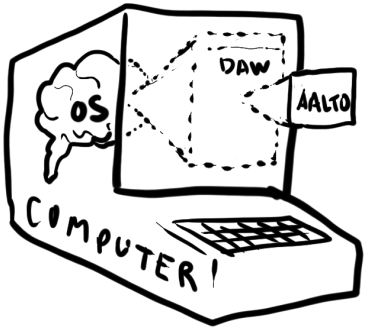
This manual is arranged in three sections:

- Taking Control of Aalto—A map of Aaltoland and a crash course in speaking fluent Aaltoian; how to interact with and interpret the various dials, displays, and doodads you'll find along the way.

- A Menagerie Of Modules—A hands-on tour through the different parts of Aalto, module by module.
- Synthesizing in Style—How to put all of Aalto's modules together to get musical results. Tricks and tips. Mind-opening asides. Ample departure opportunities for points unknown.

1 Taking control of Aalto

This section will take you through Aalto's place in the software ecosystem on your personal computer, and the various types of controls you'll find throughout Aalto. Some controls act just like you would expect, and some have more personality than that. A working knowledge of a few principles should let you use Aalto effectively in your projects, and make its workflow open to you like a well-worn book.



Very first things first

Aalto is a software synthesizer *plugin* that comes in both VST and AU formats. A plugin does not run on its own. It needs a piece of application software, a *host,* to provide an environment in which to run. Sometimes hosts are also called DAWs, for *D*igital *A*udio *W*orkstation. Some good hosts include Logic and Numerology (Mac), FL studio (Windows), and Ableton Live (both Mac and Windows). All of these hosts provide easy ways to use synth plugins like Aalto.

In Ableton Live, for example, you start by clicking the plug icon to show the "Plug-In Devices" menu. Within this menu, you can find the

Aalto AU plugin in Audio Units / Madrona Labs, and the VST plugin in VST / Local, if you used the default locations when installing Aalto. Now, you can click on the Aalto AU or VST, hold the mouse button down, and drag it to the area marked "Drop Files and Devices Here" in the Arrangement or Session views. This makes a new track with Aalto on it, ready to play.

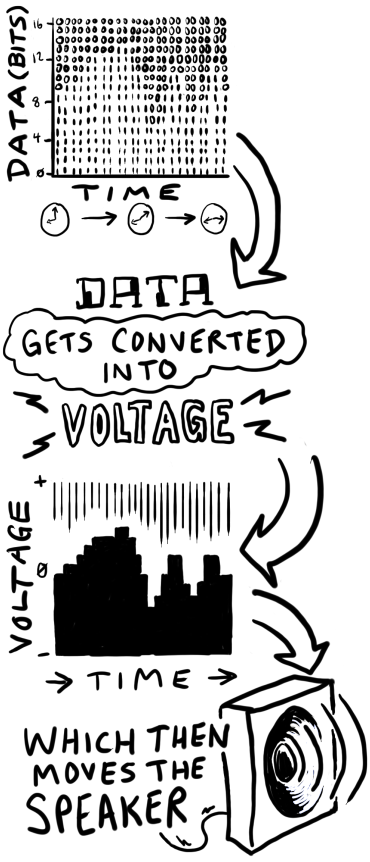
Unfortunately the methods for adding plugins in other hosts are all slightly different, and there are a lot of different hosts out there. So if this is your very first time running a plugin synth, we're sorry but we have no choice but to refer you to the manual for your particular host for guidance.

Signals

Aalto is a digital instrument that generates and processes *signals* to make sound. What is a signal? Well, for starters, telephone transmissions are signals. Radio waves are signals. Wild hand gestures are signals. A signal is something changing, over time.

Most things in life that change, change in different ways at once. Imagine trying to write down a description of a hand gesture so that someone else can recreate it perfectly, later. You'd need to come up with some way of measuring the different changes going on: the position of the hands in space, the distance between them, maybe the bending of each joint of each finger, and so on—and record numbers to describe each change. Real-world signals are often like this: complex and multi-dimensional.

Signals we make with machines, on the other hand, are often conveniently one-dimensional. When we connect a computer to a sound



card to an amplifier to a speaker, we can make sounds by changing a single number over time. This number describes a voltage to be made at the sound card's output, which eventually travels through the amp, changing the position of the speaker cone, and the resulting compression of the air. Our ears are very very sensitive to air pressure, and any tiny change, provided it is rapid enough, can be heard as a momentary sound. If the changes repeat the same shape over and over, we hear this shape as a consistent tone, provided that the shape is repeated between 20 and 20,000 times per second, the low- and high-frequency limits of our hearing, roughly speaking.

If the changes are too slow to hear, the signal may be best considered as a *modulation* signal, meant for affecting other signals that you do hear. Modulating a signal means simply: changing it somehow over time according to another signal. Audio and modulation signals are exactly the same thing: a single changing value, but with different rates of change. Aalto is programmed so that you can manipulate them easily in exactly the same way.

Every repeating tone has a *timbre,* or sound quality. High A on a flute sounds different from the same note played on a violin, and different from the same note made by singing "aah", and so on. So, we say that these sounds have different timbres. Physically, what's going on here? Well, they all share the same lowest repeating frequency in common, which is why we say they are the same note. Since they are all repeating tones, they are made up only of integer multiples of that lowest frequency, called *harmonics*. Different timbres have different mixtures of harmonics.

If you've only read about this stuff, you might think timbre is pronounced like "timber," but actually people say it like "amber" with a t in front.

In this manual we will use two kinds of pictures to show signals. One is the *time-domain* kind, which you may be familiar with. Like the clas-

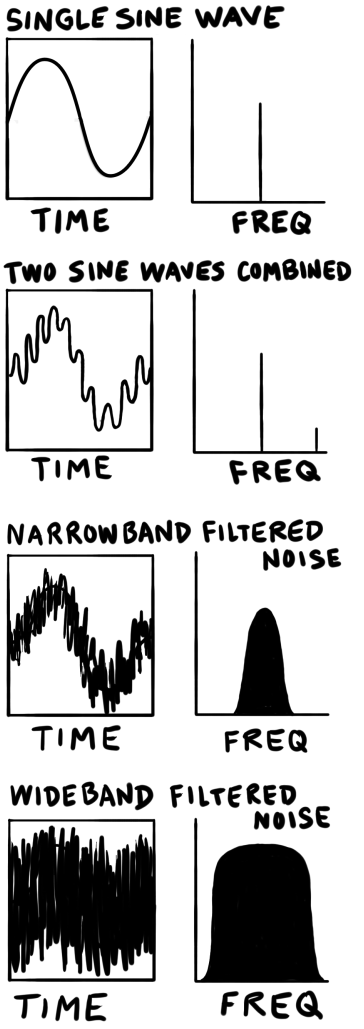
sic oscilloscope display, a time-domain image of a signal is simply the signal's value as a vertical position over time, from left to right. Zero is halfway up the graph, because most audible signals oscillate more or less equally above and below zero; however, we will run into signals used for modulation, such as envelopes, that swing in only one direction. This kind of picture is said to be in the time domain because the x axis of the image, called the *domain* in mathematics, represents time.

Another kind of picture is in the *frequency domain*. As you can guess, the x axis in this kind of an image is frequency. A frequency-domain image shows the frequencies that make up a sound at a particular instant in time. An unchanging timbre, in other words. If you like, you can think of time in a frequency-domain image as running through the z axis, out of the page, to make a 3D image of the changing sound.

A frequency-domain picture of a sound shows all of its *partials*. Saying that a sound has a partial at a given frequency, just means that part of the sound is made up of a sine wave at that frequency. Every sound can be described as a collection of changing partials over time. Each partial is usually shown as a single vertical line at the given frequency. The line's height is that partial's volume.

By comparing time-domain and frequency-domain pictures of the same sounds, one can develop an intuition about what frequency components are in a given time-domain picture, or vice versa. The two ways of looking at sound are complementary—some qualities of sound are much easier to see in one way than another.

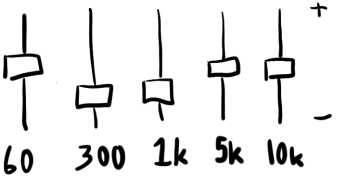
A *frequency response* is another kind of picture we'll see in this manual. There's seemingly one of these diagrams in the corner of every high-end stereo ad. And, the graphical EQ curve on your car stereo (if you have a car, and it has a stereo, and the stereo has a graphical EQ) is



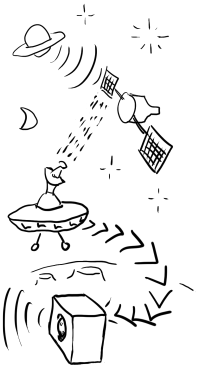
another example. Frequency responses are like frequency-domain images of signals, except they show the changes that will result to any signal from passing through a filtering process. They are shown as continuous lines over the frequency domain.

With practice, you can use these different kinds of shapes—time domain, frequency domain, and frequency response—to help you think about synthesis. Aalto was designed to be a flexible, inspiring synthesizer for making changing timbres, noises, and all kinds of signals—for turning sounds you can only imagine into real-world signals that you can hear, and for exploring new sound worlds.

Only *linear* filtering can be shown in a frequency response graph. Linear filters cannot create any new partials that are not present in the input, they can only change the volume of existing partials.



An annotated map of Aaltoland



Everything that follows assumes you have your copy of Aalto running in your host of choice. If you had trouble getting this to happen, please search our forums at http://madronalabs.com or, if that fails, email us at support@madronalabs.com for guidance. We, and the growing community of Aalto users, are here to help you.

Let's look around the Aalto plugin window, as seen in Figure 1.1.

1. Header

   We call the area at the top the *header,* maybe because it sounds better than "the top." This area contains a nice big preset selection button / display with back and forward buttons for flipping through presets, and settings that affect the overall plugin. There's also a title at left, so you remember what plugin you're using and who made it. At the very right, the header shows your license information as well as the type of plugin running. This last information can be handy to see at

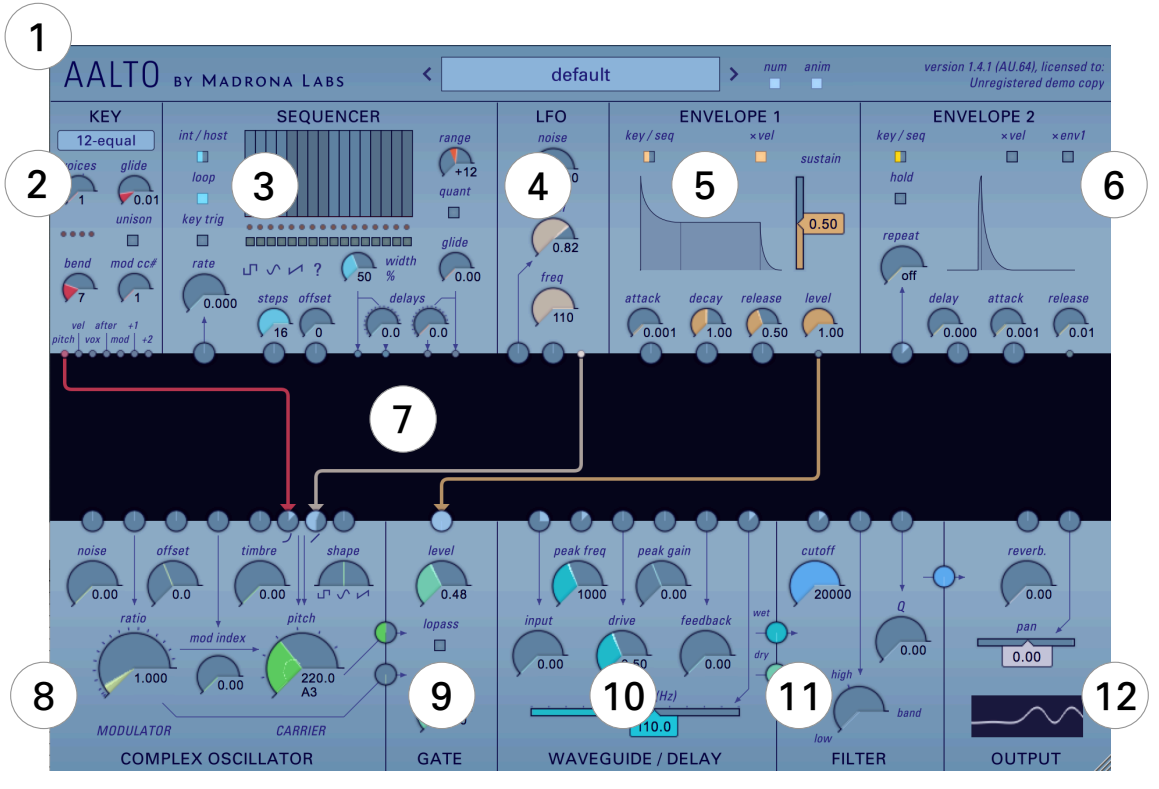


Figure 1.1: Map of Aalto and environs.

a glance, since Aalto comes in multiple formats including VST, AU, 32-bit and 64-bit.

2. KEY

   The Key module receives the note, pitchbend and mod wheel signals you give Aalto over MIDI, and makes them available to the rest of Aalto'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 signals.

3. SEQUENCER

   The Sequencer module creates both arbitrary linear functions and patterns of rhythmic pulses over time. You can use it to create repeating melodies, control changes, rhythms, or any combination thereof.

4. LFO

   LFO stands for Low Frequency Oscillator. Just about every synthesizer has one. Aalto's LFO creates a musical blend from sine waves to tuned noise that can extend from very slow wobbles up to audio frequencies.

5. ENVELOPE 1

6. ENVELOPE 2

   The two envelope modules create time-based control vectors, commonly used to control volume or timbre as notes are struck and/or held. Envelope 1 is a fairly standard ADSR (Attack-Decay-Sustain-Release) type, but with fully mod-able ADR. Envelope 2 looks like a "simpler" AR (Attack-Release) model at first blush, but its Hold, Repeat and Delay functions hint at far greater possibilities.

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. COMPLEX OSCILLATOR

This module is the heart of Aalto. It contains two internal oscillators: *MODULATOR* and *CARRIER*. The modulator is connected directly inside the module to control the frequency of the carrier. This arrangement is known as *FM*, or Frequency Modulation synthesis. In combination with the timbre control, a nonlinear wave folder, and the shape control, which can turn a sine wave into a saw or pulse wave, FM synthesis in the complex oscillator module is a gateway to a huge, multi-dimensional space of timbres.

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 1 and LFO to change the level (VCA mode) or the low-frequency cutoff (LPG mode) of the synthesizer's signal.

Synth heads will note that Aalto's complex oscillator looks a lot like the mighty Buchla 258. While our work was inspired by Buchla's excellent hardware designs, Aalto is not an emulation. If you want Buchla sounds, you have to get a Buchla!

10. WAVEGUIDE/DELAY

This module is a delay with a waveshaper and a peaking EQ built into the feedback loop. Because it has such short and controllable delay

times, unlike a typical analog delay, it can be used as an additional oscillator or waveguide filter.

State-variable: a classic kind of analog synthesizer filter, well-suited to digital implementation.

11. FILTER

    Modeled after the Oberheim SEM filter. A state-variable filter with mixable simultaneous outputs.

12. OUTPUT

    This is the final line of defense/manipulation the signal will pass through before you can hear it. It includes a single-control spring reverb effect, inspired by the reverb tank in the Arp 2600. Also found here are the Pan control, a modulation-friendly stereo balance control, and the Oscilloscope, which gives you visual feedback about the waves you're making.

Presets

We tried to make Aalto so easy to use, so compellingly tweakable, that you may find yourself wanting to dive in and start making your own sounds right away. But flipping through the preset sounds first is obviously a good way to hear what Aalto can do. Aalto 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 Aalto 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. The categories of factory presets are:

- Aalto keys

  A collection of simple and musical sounds, most traditionally played

with keyboards, resides in here. If you're looking for an Aalto-style version of a classic synth sound, try Aalto keys.

- Aalto pads

  Here you'll find sounds that have more complex variations, but can still be used to play a recognizable chord or melody. In other words, lush and weird tonal sounds. Hold down some keys and listen.

- Aalto percussion

  This category is where you'll find drums—both synthy and acoustic-sounding—metallic sounds, atonal plonks recalling our favorite modular synthesizer patches, and some weird and wonderful presets that are maybe the most uniquely Aalto-like of all.

- Aalto solos

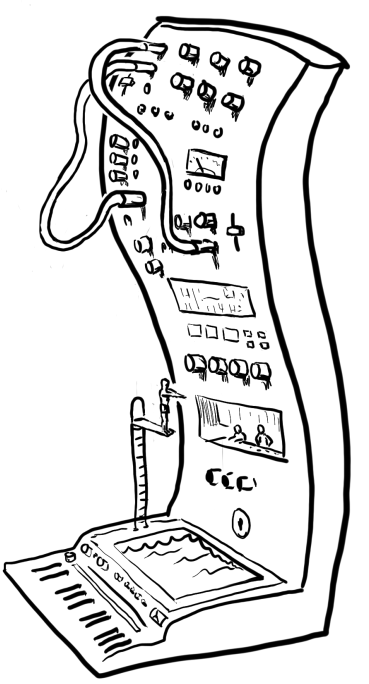
  Melodic sounds that you'll probably use only one voice of at a time. These include basses and complex lead sounds.

- Aalto techniques

  This category contains patches that are meant to demonstrate a single technique for using Aalto. Not great sounds, in other words, but instructive patches. We tried to keep these patches to three cords or fewer. Some less obvious ways of using Aalto, like using an envelope as an oscillator, or modulating the sequencer with its own output, can be found here.

- Aalto textures

  In this category, the gloves are off. Hold down a single key—or don't, in some cases—and listen to evolving panning swooshing, a repeat-



ing melody, or organized noises that might be useful as evocative ambience.

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, Aalto'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 Aalto 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 a dial's position, 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

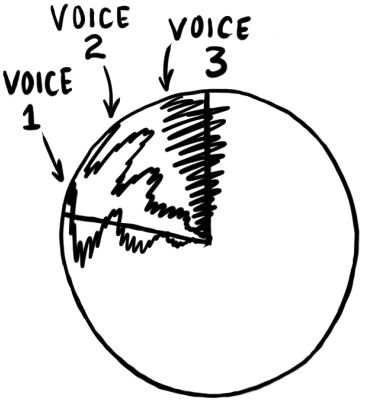
the smallest currently visible increment of the dial. Scrolling faster accelerates the change.

- 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 Aalto 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 the oscillator pitch, you will see four straight lines in the pitch dial. And if you send more complex modulations to the pitch dial, you will see multiple scribbly lines, all animated. If all the visual fireworks get to be too much, remember, you can turn the animations off using the *anim* button in the header. But we think you'll come to find that the signal displays are useful indicators of what's going on throughout Aalto.

Detents

Some dials, such as the oscillator pitch, have *detents*. Detents are useful default positions. For example, the oscillator 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 Aalto'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 editable.

The ctrlnum toggle in the header lets you turn off all the numerical displays, if you're more of a visual than numeric kind of person.

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

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, 20, 200, 2000 and 20,000 Hertz are all equally spaced apart on the filter cutoff dial.

Those horizontal ones

Two controls, the waveguide repeat time and the pan position, made much more sense as horizontal bars rather than round dials. So while calling them "dials" seems a little weird, everything about these controls, both viewing and setting them, is the same as what we've just gone over for the round dials, except for the geometry.

Using buttons and switches

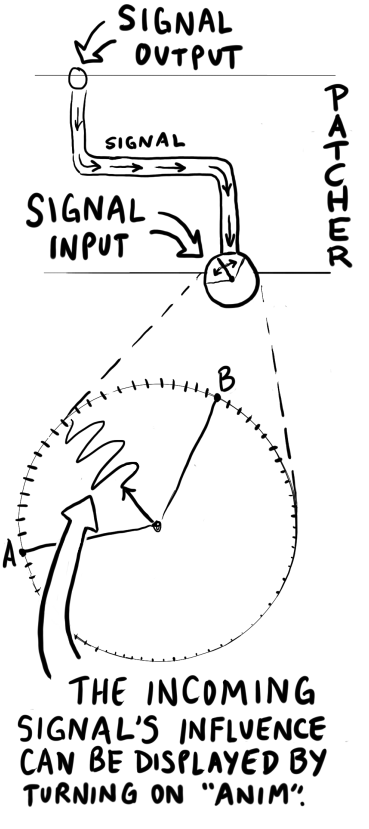
There's not a lot to say here, only that switches need only be clicked to be toggled (you'll see the little dark switch move back and forth) and the same goes for the buttons. Dark is off, bright is on. A switch always has the meanings of the two positions labeled clearly, as in for example the *int / host* switch.

Using the patcher

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

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 1 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.

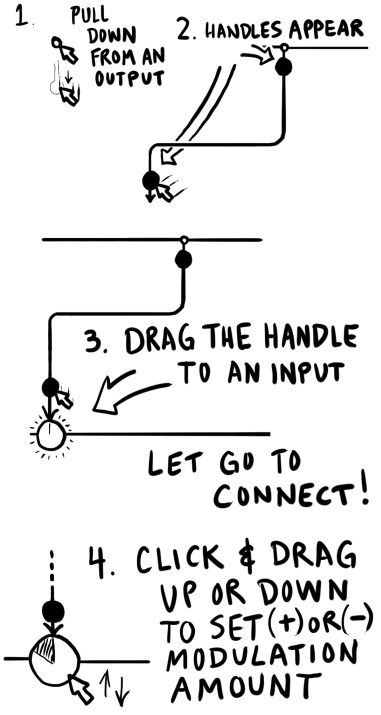
You can use the LFO to see this, even without using any patch cables, as follows: turn the *freq* dial on LFO to 1.0. Double-clicking the dial

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 light 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 pitch input to the COMPLEX OSCILLATOR has 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.



Patching

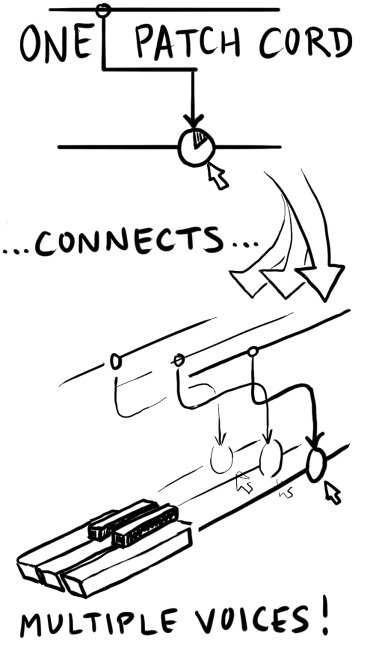
To make a patch cord, drag from an output to an input. As you drag, you'll see a glowing line with an arrow at the end stretch from one to the other. This shows the new connection you are making. This is a pretty nifty thing, since such routing does not involve deciphering menus or matrixes of things 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 Aalto 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*.

Almost every module panel in Aalto'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. If you connect the output of the LFO to the oscillator pitch, for example, 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 modulation signals that flow through each voice's patch can be very different. Thus, each voice is separately controllable, in timbre or any other parameter you can think of.

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

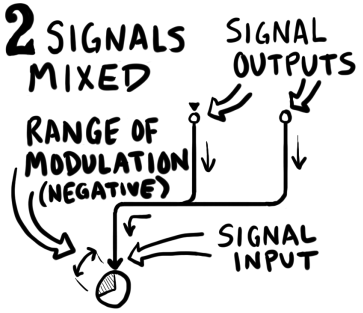


that point. You can also select a group of cords in the patcher by clicking on an empty part of the patcher, then dragging over multiple cords. When a cord is selected, its *handles* are visible. Handles appear as circles at either end of the cord—you can drag them to move the ends. If multiple cords are selected starting or ending at the same place, clicking the handle there will move all of the selected cords.

Typing the delete key should delete all the selected cords. You can also delete a patch cord 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.

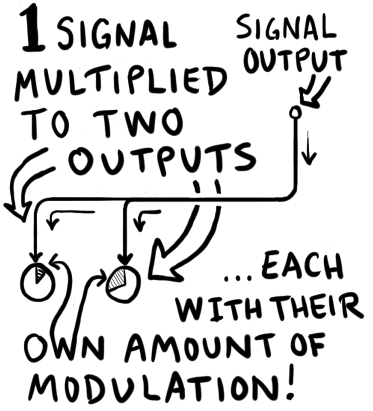
If you create a new copy of Aalto in your DAW, it will appear with the preset "default" selected: two patch cords that make a simple sine wave patch. The pitch output of the KEY module is connected to the pitch input of the COMPLEX OSCILLATOR, and ENVELOPE 1 is connected to the GATE. Each connection is clearly visible as an arrow connecting a signal output to a signal input. When you play a key, the KEY module translates the pitch of the MIDI note to a signal representing that pitch. The patch cord sends this signal to the COMPLEX OSCILLATOR, which is calibrated to play the appropriate pitch in response. While the key is held down, ENVELOPE 1 goes through its attack, decay and into its sustain mode, holding the value set by the sustain slider. This value is sent as a signal through the other patch cord to the GATE module, which lets through the signal from the oscillator while the level signal is held high.

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.

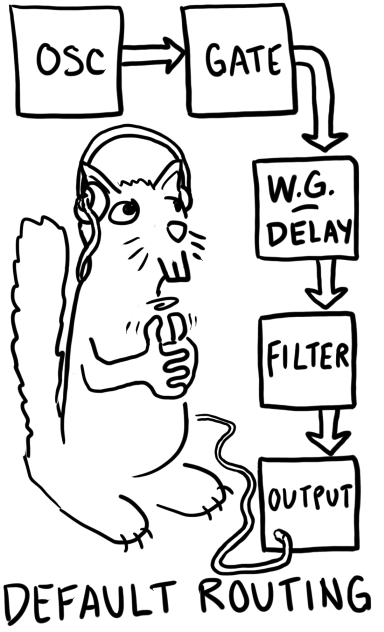
Default signal routing

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

The COMPLEX OSCILLATOR → GATE → WAVEGUIDE / DELAY → FILTER → OUTPUT signal path, consisting of all the modules below the patcher, is pre-routed. Small dials between some of these modules (that suspiciously resemble the signal inputs we saw in the patcher, if you ask me) act as signal level controls between the modules.

The pre-patched modules below the patcher can be called the sound modules. While the module above the patcher generate primarily inaudible signals, the ones below are where the sounds you hear at the output are made. So there's our voice, pre-patched (to a degree!) You might visualize the bottom row of modules as the players in the orchestra pit, and the top row of modules as their wildly gesticulating conductor. Or wait, a row of conductors. All waving their arms wildly, the motions of which are... sent to the players on little cables. OK, this analogy doesn't work at all. Or maybe that's what orchestras will be like in the future.

The COMPLEX OSCILLATOR has two signal outputs: the carrier oscillator and the modulator oscillator. The small dials between it and the GATE module let you choose how much of each oscillator to apply to the gate module's input. The dials between the WAVEGUIDE/DELAY and FILTER modules let you set the wet/dry balance of the WAVEGUIDE/DELAY output. Finally, the one small dial between the FILTER and OUTPUT modules acts as a post-filter volume control, before pan and reverb are applied in the OUTPUT module.

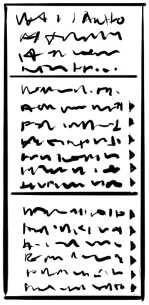


2 A menagerie of modules

This chapter will take you on a more detailed tour of the various modules that compose Aalto, one by one. A lot of features are covered here in detail, and mainly in isolation. For more information on how to bring them all together, see Chapter 3: Synthesizing in Style.

The header

Apart from reminding you which amazing synthesizer you're currently using, Aalto'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 Aalto 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 Aalto 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 Aalto. This can be useful when recording multiple takes of Aalto 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

The menu choice "Convert presets..." will appear only in the Macintosh version of Aalto, where you might find the need to convert from .mlpreset to .aupreset and vice versa. When selected, Aalto will look for presets that aren't readable by the current type of plugin, and try to convert them so they can be used. This function can be useful if you're converting from a VST to an AU workflow, or vice versa.

program change 128 to Aalto, 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 Aalto's patches in two more sections: the so-called "factory presets," in directories all starting with "Aalto," followed by storage space for your own personal patches. Some user presets, contributions from fellow Aalto users, are installed here by default.

Thanks, wonderful sound-crazed Aalto 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 Aalto, 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)* / User-Data. 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 Aalto, 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: /Music/Madrona Labs/Virta
- Windows: (Your home directory)/AppData/Roaming/Madrona Labs/Virta

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 Aalto 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

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

Aalto that corresponds to your chosen patch, and Aalto will dutifully switch to that patch.

Display options

The *num* button lets you toggle the numerical displays on and off for all controls. Sometimes, you want to set adrift on waveguide bliss and numbers will just mess up your headspace. Other times, you crave precision, and want them back. It's up to you.

The *anim* button turns the signal dial animations on or off. While always fun and often useful, these animations can be somewhat CPU intensive, so you may want them off.

Aalto's display is fully vector-based, and thus can be resized freely, to suit your needs or mood. Drag the handle on the lower-right corner of the window to resize it.

These options are saved globally, so they won't change until you decide to change them again.

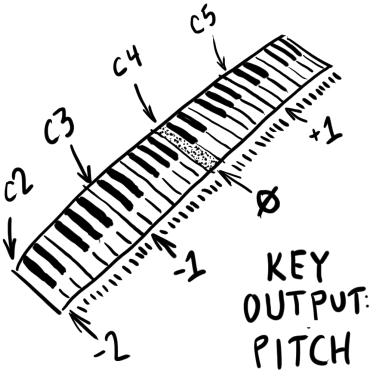
Version and registration

The right top corner shows the version of Aalto you are running, as well as your registration info. When you purchase a copy of Aalto 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 Aalto if any problems arise, and to maintain and improve it.

KEY

The KEY module receives all the MIDI data you send Aalto'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 Aalto 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 Aalto. 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 semi-

tones, 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 Aalto'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 Aalto to take when sliding between notes.

- *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 Aalto 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 COMPLEX OSCILLATOR *pitch,* and FILTER *cutoff,* 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 sequencer 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 Aalto by sending messages from a MIDI knob or fader controller.

SEQUENCER

The sequencer is a powerful control and trigger generator, capable of creating sequences of control signals for varying pitch, timbre and other over time, and trigger pulses for doing things like firing envelopes. Unlike most other modules, the sequencer's outputs are color-coded with two different colors, blue for triggers and orange for values, to remind you of their functions. Both the triggers and the value outputs are still just signals, however, compatible with all the signal inputs in Aalto.

Each of Aalto's voices has its own copy of the sequencer under the hood. In this section we'll look at all of the controls for the sequencer, including ways to make each copy do something slightly different from the others using the patcher. By varying the rate or offset of each sequencer copy, Aalto can create some very complex textures without breaking a sweat.

The big multi-slider / trigger button area is is where you create your sequences. Click anywhere in the multi-slider to set the control signal value output at each step. Drag across it to set multiple steps. Click any of the 16 buttons to fire a trigger pulse at that step. The row of round lights shows you which step the sequencer is currently on. When multiple voices are playing, and they are on different steps, you'll see multiple lights making their way independently around the cycle.

Like analog hardware sequencers of the past, and unlike many digital instruments, Aalto's sequencer operates completely within the signal domain. So timing is rock-solid, and you can do fun things like varying the speed up smoothly from normal tempos to audio rates.

Controls

- *int / host:* This switch lets you the choose the clock source that runs the sequencer: either Aalto's internal, freely controllable clock, or the main tempo in your host app. When this switch is set to host, the sequences don't move unless you press play in your host application.
- *loop:* This button toggles the sequencer's looping on or off. When on, the sequence will loop indefinitely when played. When off, it will play through its steps once, then stop on the last step until triggered again.
- *key trig:* This button lets you select whether or not to restart the sequencer each time a MIDI note is received.
- *steps:* This dial controls the number of steps in the sequencer's cycle. Sending signals to its input dynamically changes the number of steps in the cycle, with just the sort of mysterious results you might expect.
- *rate/host ratio:* This dial controls the rate of the sequencer's internal clock. When *int/host* is set to "int", this dial lets you set the sequencer's rate in Hertz. When *int/host* is set to "host", the dial lets you set the sequencer's rate as a ratio of the host sequencer's tempo. By default (1/1) the sequencer advances in a 16th-note rhythm, locked to host tempo. Settings above or below 1/1 let you advance the sequencer at larger or smaller note values. You didn't hear it from us, but this control can be particularly fun to send modulation signals to.

  It may seem like the top end of the *rate* control, 13.75 Hz, is not enough to really create audio-frequency signals at the output. But note that this is the rate at which a 16-step cycle repeats; the duration

With *loop* turned off and *key trig* on, you can use the sequencer as a multi-segment envelope generator. The envelope will restart every time you play a note, play all the way through at the speed selected by the *rate* dial, and then stop.

In "host" mode, sequencer rate slews smoothly from ratio to ratio as you manipulate (or modulate) the *host ratio* setting. This introduces many intriguing rhythmic possibilities that would be quite difficult to sequence by hand.

of each step stays the same when you change the number of steps in the sequence. If you set the number of steps to 8, the sequence can repeat twice as fast: 27.5 Hz. And if you set ***steps*** to 2, you get a two-step sequence repeating at 110 Hertz.

Hey, that's a low A note!

- ***offset:*** This dial lets you move the sequence position forward a specified number of steps. If the total position is greater than 16, the position will be wrapped to the beginning.
- ***width:*** This dial lets you set the pulse width of the triggers generated by the selected step buttons in the Sequencer.

Internally, the sequencer generates a phase signal that increases continuously from 0 to the number of steps, then wraps back to 0. Within each integer step, if the fractional part is less than the pulse width, a 1 is sent to the trigger output. Otherwise, a 0 is sent.

- ***delays:*** The controls for the two delays are calibrated in steps. So, they do not set fixed times between events, but fixed offsets within the sequence that maintain their musical relationships even as the sequencer's rate changes. They vary from 0 to 8 in increments of 0.5 steps. If you want finer precision, hold the Shift key while dragging to set delay in increments of 0.1 steps.
- ***range:*** This dial multiplies the sequencer's control output by a positive or negative value, calibrated in semitones.
- ***quant:*** The signal then travels to the quantizer, toggled on and off by the ***quant*** button. When on, it constrains the control output to semitones, which is helpful when you're using the sequencer to play a melody, for example.
- ***glide:*** This control lets you set the amount of glide (also called slew, or portamento) applied to the quantized value signal, much like the ***glide*** dial from the KEY module. The signal is then passed down to the two value outputs. How glide works is actually a bit tricky. It's made using a direct calculation as follows: ***glide*** sets the percentage of each step over which the value signal is linearly mixed from the

previous step value to the current step value. The mix occurs while the fractional part of the step is less than this percentage. When the fractional part is greater, the current step value is output. This enables the output signals to be consistent as *rate* is changed and the sequence is moved forwards and backwards.

- Preset wave buttons: The square, sine, and saw buttons are momentary buttons that change all of the value outputs into one of 3 preset shapes, especially useful when using the sequencer as a glorified LFO. The ? button creates a randomized sequence. What will it be? Only chance can say!

Outputs

The two blue outputs are trigger outputs. When the sequence step is within the first part of a step selected by the *width* control, the trigger value is 1, otherwise it is 0. The two orange outputs are value outputs. They send out the values set by the multi-sliders for each step, as processed by the *range, quant,* and *glide* controls.

Each type of output has a direct and a delayed version. Each delay can be set independently—the controls are under *delays,* of course. This arrangement lets you get four different streams of information out of the sequencer, at constant distances apart that you can easily set.

LFO

Now we go from the most complex module to the simplest. The LFO module is a Low-Frequency Oscillator—one of the canonical modulation sources in synthesis. Practically every synth has an LFO. LFOs are what make the mod wheel on your DX7 wiggle the pitch around rhythmically as you play in your Zapp tribute band.

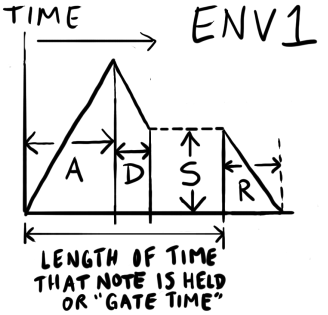
Controls

- *noise:* This dial crossfades the output signal from a sine wave to pure noise. This can introduce a pleasing randomness, or complete chaos, to the normally serene sine waves at the output.

- *level:* This dial controls the amount of signal sent to the patcher. This dial has a control signal input, letting you do all kinds of neat things, such as fading the LFO's influence in and out or introducing frequency modulation via another control source, such as the Sequencer or Envelopes.

- *freq:* This dial adjusts both the frequency of the sine wave and the cutoff of a lowpass filter on the Noise generator. The filter frequency is equal to the sine frequency, multiplied by 50. This value was chosen by ear to create a good balance between sine and noise as modulation sources. This dial can accept control signals, even from LFO's own output.

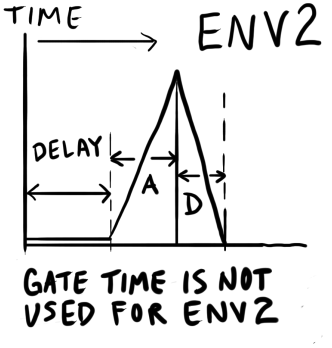
ENVELOPE 1 and ENVELOPE 2



Aalto has two envelope generators (lucky us!). Each is designed to complement the other by doing some special things that the other one can't do. Each has only one output: a time-varying signal that is triggered by an input signal. Typically an envelope is used to control the overall loudness of a note or some aspect of timbre as the note evolves.

ENVELOPE 1 is an ADSR generator, so your old pals Attack, Decay, Sustain and Release each have their own control. A final Level control sets the envelope's overall signal level, for more options when patching. The ***attack***, ***decay*** and ***release*** controls have inputs for control signals.

ENVELOPE 2 is a less-common envelope type—we could call it DAR, for Delay, Attack and Release. Envelope 2 also has a repeat time setting, which can turn it into a repeating LFO with a variable shape, or even an audio oscillator. The ***delay***, ***attack*** and ***repeat*** controls have inputs for control signals.



Each envelope has a ***key/seq*** switch that lets you choose whether the envelope is triggered by a midi notes, or by the sequencer's trigger outputs. When seq is selected on ENVELOPE 1, pulses from the direct trigger output on the sequencer trigger it. On ENVELOPE 2, triggers from the sequencer's delayed output are used.

The ***x vel*** control on each envelope multiplies that envelope's output by the velocity of incoming MIDI notes, when ***key/seq*** is set to key. Typically this is used to make notes louder when keys are hit harder, but with two velocity-sensitive envelopes, there are many other qualities of sound that the velocity can be applied to.

The graphs in each Envelope module represent the actual shape of the envelope over time. They are scaled to match the total duration of

the envelope sequence. Envelopes have logarithmic attack and decay curves, and those time settings are calibrated to correspond with the time at which the output value has traveled approximately 60% of the way to its destination.

The *hold* toggle on ENVELOPE 2 acts just like ENVELOPE 2's *sustain* control, except that it only has the values 1(on) and 0(off).

The *repeat* control on ENVELOPE 2 re-triggers the envelope at the rate (times per second) you set. The envelope is always retriggered when a trigger is received, and the repeat clock restarts from that instant. In the graph, the section that repeats is shown as a dark bracket below the envelope. If the *hold* toggle is turned on, *repeat* has no effect.

The *delay* control on ENVELOPE 2 lets you set the pause between the incoming trigger and the start of the envelope's attack. Sometimes you want an envelope to wait to pounce, yes? This is useful for creating complex envelope shapes such as flams, in combination with ENVELOPE 1.

Finally, ENVELOPE 2's *x env1* button lets you multiply the output of ENVELOPE 2 by the current value of ENVELOPE 1. This is useful for making those bouncing ping-pong ball sounds that are so important in Cornish dance music.

COMPLEX OSCILLATOR

Ahoy! Here be the source of waves aplenty. The COMPLEX OSCILLATOR module is the heart of Aalto. It contains two internal oscillators: *MODULATOR* and *CARRIER*. The modulator oscillator is connected directly inside the module to change the frequency of the carrier. This arrangement is called two-operator FM, for Frequency Modulation. The frequency of the modulator is set as a multiple of the carrier frequency.

This allows a timbre you make by finding a nice modulation setting to remain the same over the whole range of possible pitches. Every control on the COMPLEX OSCILLATOR features at least one signal input, so let's get modulating!

Controls

- *noise:* This dial adds noise to the modulator signal. Like the LFO noise, it doesn't just fade in white noise, but gradually tunes a filter matching the sine wave frequency. In this case, the filter is a resonant bandpass that smoothly fades the sound from a single sine wave to wide-spectrum noise.

The section on Signals in Chapter 1 has some illustrations that show what this mixture looks like in both the frequency and time domains.

- *ratio:* This dial lets you set the modulator oscillator frequency as a ratio of the carrier oscillator. Note that when dragging, it stops at detents in the *harmonic series:* integer multiples of the carrier frequency. As always with dials that have detents, shift-drag to get values in between.

- *offset:* This dial adds a constant offset to the modulator frequency, in Hz. Set it to just a few Hz to add a little sort of flanging to tones, or higher to get clangorous FM sounds.

To set the modulator to a constant frequency, set *ratio* to 0.000, then dial in a constant Hz offset.

- *mod index:* This dial lets you set the amount that the modulation oscillator modulates the frequency of the carrier oscillator. Turning this up increases the amount of higher harmonics in the output. Signals from the patcher are sent to this control through a special lowpass filter that mimics the response of a *vactrol.* A vactrol is a combination of a light source and a light-sensitive resistor, used to control voltages. The shape of its response to a pulse is very interesting: it slews up much faster than down, and has a complex nonlinear

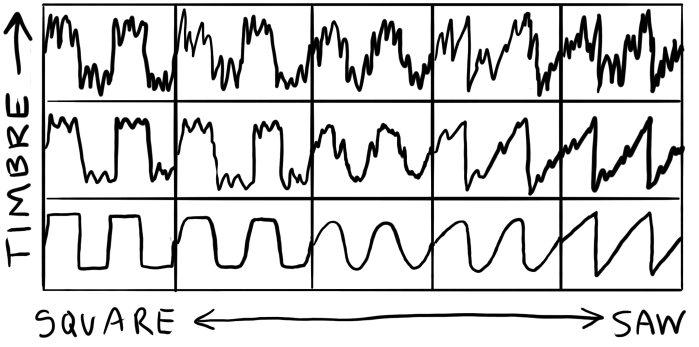
decay. In the right circuit, it can help make signals come alive with a special organic quality.

- *timbre:* This dial controls a complex wave folder network. It is applied to the carrier oscillator after frequency modulation. The timbre network generates a number of higher harmonics from an input signal simultaneously, and the number increases as the control is turned up. In order to avoid any distortion due to digital aliasing, the effect of the *timbre* control decreases at very high pitches. Modulation signals to the *timbre* dial flow through another vactrol emulation.
- *pitch:* This dial controls the pitch of the carrier oscillator. As with other dials, the numerical display shows the value, pitch in this case, when no modulator signal is connected. This is also the pitch of the oscillator when the *pitch* signal from KEY is connected, and an A4 note (MIDI note 69) is played.

  Unlike other dials, the pitch dial has two signal inputs: one exponential and one linear. The exponential input is calibrated to match the 1.0 / octave standard of Aalto's patcher signals. The linear input has a much wider range and can be modulated for fast "thwips," kick drums, laser beams and so on.
- *shape:* This dial controls an antialiased waveshaper that can transform a sine wave into either a square or a sawtooth wave. The waveshaper is applied to the carrier oscillator after the FM, so the two can be combined to make very interesting sounds full of moving harmonics. Patching LFO or SEQUENCER signals to this control can create very satisfying wave sequence effects.

  The wavefolder and waveshaper circuits are both applied directly to the carrier oscillator post-FM, then mixed to form the final carrier output that is sent to the GATE module. As *shape* is moved, the mix

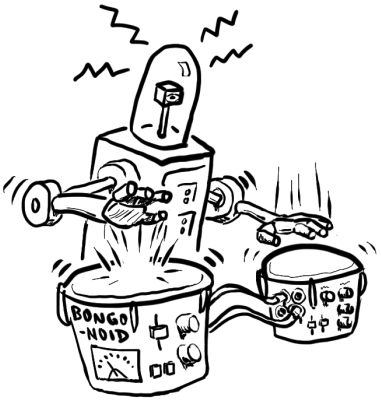
proportion changes: at the center, it is all wavefolder, and at either extreme, about 90% waveshaper.



As ***shape*** is moved to either end, the mix of the timbre circuit in the output becomes less and less. At either extreme, about 1% of the timbre circuit remains in the mix, just to remind you that nothing is broken when you change the ***timbre*** dial.

The combination of the wavefolder and waveshaper circuits forms a two-dimensional sea of timbral possibilities in itself. Some scattered points on it to guide you are shown here. Once you start navigating with the ***timbre*** and ***shape*** controls, you'll start to get a feel for the many kinds of wave shapes you can sail to. Trying them out with ***mod index*** set to 0, so that the input is a simple sine wave, is a good way to understand their behavior by themselves. To make a drawing of all the timbral possibilities with frequency modulation in the mix, we would need three more

dimensions for modulator frequency, modulator index, and noise! If you have any five-dimensional paper, please send us a piece.



GATE

The GATE module is a dynamic volume control, akin to the VCAs (Voltage Controlled Amplifiers) found in modular synths. Its input is a mix of the two signals from the COMPLEX OSCILLATOR according to the small knobs in between the two modules. Its output is sent to the WAVEGUIDE/DELAY. You send it 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 another vactrol emulation, this time with a settable decay, which opens up a world of cool, percussive envelopes.

The GATE is the main tool used to sculpt the dynamic profile of Aalto'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.

WAVEGUIDE / DELAY

This module is an allpass-interpolated delay, with a waveshaper and a peaking filter built into the feedback loop. Say that five times fast. Since it offers such short and controllable delay times, unlike a typical analog delay, it can be used as an additional oscillator or a waveguide filter.

An *allpass* filter leaves the levels of a signal's component frequencies intact, but delays some of them more than others.
Interpolation is the process of finding the values in between the samples that make up a signal. This is needed if you want to have a delay length that is not an exact number of samples, or to change the delay length smoothly.

Controls

- *input:* This dial lets you set the level of the combined signals from the COMPLEX OSCILLATOR sent to the delay.

- *peak freq:* This dial lets you set the frequency of the peaking filter inside the delay loop. Here again the default position of the input dial makes this frequency track the keyboard pitch, if you so desire.

- *peak gain:* This dial lets you set the gain of the peaking filter. When the frequency of the delay is low, the filter acts like a gentle boost or cut EQ. At higher delay frequencies, the filter acts on the feedback signal many times per second, radically changing the harmonics that are created by the system.

- *drive:* This dial lets you set the amount of distortion in the delay loop. Distortion is introduced using a waveshaping algorithm that adds both odd and even harmonics.

- *feedback:* This dial sets the amplitude of the delayed signal fed back to the input. Obviously, this is the main way to control how the repeating signal decays, but don't forget that the *drive* control is very important too. A tiny bit of noise is added to the input of the delay, so that either lots of drive or feedback or both will put the delay into self-oscillation. At low frequencies, less than 20 Hz, this oscillation will take a long time to build up, but at higher frequencies it will make noise right away. You may notice that the oscillation never runs away to infinity. That's because there is a soft saturation algorithm built into the feedback loop, which both limits the signal and creates complex nonlinearities in the oscillation. As with any chaotic system, the most interesting sounds happen right at the verge of oscillation where the system is almost, but not quite, stable.

- *frequency:* This long linear control sets the repeat frequency of the delay, along a logarithmic scale. Patching the KEY module's Pitch output to this slider controls the delay's frequency in musical intervals that match the oscillator's pitch.

- *wet* and *dry:* These two small dials on the right side of the module set the mix of delayed and direct signal output to the FILTER.

FILTER

Ah, here we are, at the filter; every synthesist's best friend. Modeled after the crisp and excellent sounding Oberheim SEM filter, Aalto's filter is a

state-variable design with mixable simultaneous outputs. This provides a wide tonal palette and a good counterpart to the GATE module's mellow lowpass. All of the filter's controls have corresponding signal inputs on the patcher.

Controls

- *cutoff:* This log-curve dial lets you control, you guessed it, the cutoff frequency of the filter. Patching the Pitch input to this control will give you consistent pitch intervals, the better to match up in cool ways with incoming notes.

- *Q:* Short for "quality factor," Q is a way of measuring the sharpness of a filter. It's the same measurement as what we call the filter's *resonance,* or tendency to self-oscillate, but in different units. At 0, the filter has a very gentle rolloff, and at 1, the filter has the sharpest possible rolloff and will begin to self-oscillate, or ring. A value of 0.7 is approximately the steepest rolloff the filter can achieve without any ringing.

Though the maximum dial value of 1.0 does not quite make the filter oscillate, you'll find you can push it a little farther by adding some constant positive signal, from an envelope for example.

- *low / high / band:* This dial lets you crossfade between the low-pass, high-pass, and band-pass outputs of the filter. Settings which mix two filters let two different parts of spectrum through, and can create phaser-like sounds when *cutoff* is swept.

  When this control is between the low-pass and high-pass settings, the output is a kind of peaking filter that allows you to adjust the proportion of lows to highs, while adding a resonant bump at the cutoff. This is very useful for sounds that mimic the resonance of instrument bodies.

OUTPUT

The Output model lets you put the finishing touches on your lovely new signal before it gets sent out into the cruel, cruel world. Here, you'll find Aalto's lovable one-knob reverberator, the signal-controlled panner circuit, and a nifty little oscilloscope that shows you the waveforms you're making. Both controls have signal inputs on the patcher.

Controls

- *reverb:* This dial sets the amount of each voice's signal that is sent to the reverberator. If you're use to mixer terminology, think of this as a reverb send for each voice.

- *pan:* This dial lets you pan your signal across the stereo field. Panning each voice to a different place, using envelopes or the *vox* signal for example, can lead to some truly spacious results. Modulating this dial with audio-rate signals can create scintillating spatial shuffling effects!

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 Aalto? This is an important thing to consider. Aalto has been designed to give you the purest possible sound from its oscillator, if you choose, and an extremely wide dynamic range. So there is no compressor or limiter built into Aalto's signal chain.

Aalto is designed to be a useful tool for working producers and engineers, who all have their own favorite solutions for compressing and

limiting digital signals within the computer. So, Aalto outputs a signal exceeding 0dB, potentially causing clipping, if that's what you ask it to do. To avoid digital clipping at your audio output, you need to add some sort of limiting in the digital domain to keep the signal within the range -1. to 1. Your DAW almost certainly has a limiter plugin of some kind included, and it probably sounds OK. If you haven't thought about this issue before, you can take this opportunity to experiment with different solutions for limiting the signal, and listen for the differences in sound quality.

Finally, a high-quality audio interface is really a must to get the best sounds out of Aalto. 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, RME, and MOTU. As always, the only enduring advice is: use your ears and your own good judgement.

3 Synthesizing in style

If you've read Chapter 2, you know the modules we've got to work with, and what their controls do. But how can we best use them as an ensemble to organize sounds to our liking? That is the topic of this chapter.

> "I happen to think that computers are the most important thing to happen to musicians since the invention of cat-gut which was a long time ago."
> *–Robert Moog*

Limitations

Computers offer sound-making capabilities nearly without boundaries. Any sound we can perceive can potentially be created with computer software, so when makers of patchable software tools say "the only limit is your imagination," that's not mere marketing hype. There are plenty of software environments that provide the components needed to build any conceivable signal generator.

Aalto, on the other hand, is designed to be an instrument with a fixed set of controls that you can learn thoroughly. An instrument that remains the same lets you become intimately familiar with its workings,

so you can make changes in a sound almost as easily as thinking of them. And as a bonus, you can learn from (and share patches with) other people using the same instrument. Throughout Aalto's design process, the question often came up: how can we make as wide a range of possible sounds with as few controls as possible? This concern drove the whole semi-modular approach, what modules to provide, and the selection of controls.

We feel that limitations are key to becoming fluent with an electronic instrument. Look at the best-regarded classic synthesizer designs like the Minimoog, the Arp 2600 and the Buchla Music Easel. An experienced player can get a huge range of sounds out of them, and learn all the controls well enough so that the interface drops away, the individual decisions of what knob to turn are made more by muscle memory than conscious thought, and only the experience of playing is left. As much as is possible in software, we want to achieve this goal of making an instrument that you can connect with intimately.

Oscillators everywhere

Aalto has one oscillator module. It's a very capable one, but there's still only one. How can we make multi-oscillator sounds? Let's look at a couple of ways.

The COMPLEX OSCILLATOR has two outputs, one for the carrier and one for the modulator. You can turn up the direct modulator output (the lower one) to hear the modulator's sine + noise blend. You can detune it just a couple of Hertz for beating effects (using the offset control if you like), or to a musical interval—check out the factory patch "Aalto keys / sneaky fifth" for an example.

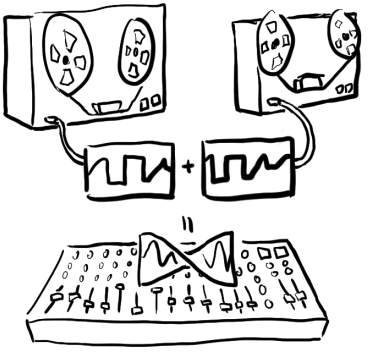
The WAVEGUIDE/DELAY can also act like an oscillator. The patch

"Aalto techniques / waveguide as oscillator" shows how to connect it to the KEY module so it will stay in tune. By adjusting the filter controls and the drive control, you can make it play a variety of timbres. Using the *wet* and *dry* controls, you can mix it with the COMPLEX OSCILLATOR totally independently.

Note that the FILTER is after the WAVEGUIDE/DELAY in the default signal routing. This allows the filter to control the overall envelope when the WAVEGUIDE/DELAY is oscillating.

Another way that the WAVEGUIDE/DELAY can be used to make the sound of multiple oscillators is by adding a slightly detuned copy of the single oscillator. By varying the delay time slowly, a sort of flanging effect is produced, and one oscillator sounds like two slightly detuned ones.

In the "Aalto techniques" section in the factory patches, you'll find additional ways to make oscillations from other modules. You can use these as building blocks for patches that use Aalto in ways that break away from the default COMPLEX OSCILLATOR → GATE → FILTER routing.



Flanging is an audio effect that was first accomplished by recording the same sound to two reel-to-reel tape machines simultaneously. The two copies were mixed together and the speed of one was slowly varied by putting a finger on the flange, or edge, of one of the reels. This adds some frequencies in the sound and cancels others, turning noisy signals into sweeping, jet-plane-like sounds.

More and more modulation

The COMPLEX OSCILLATOR is capable of modulating the frequency of one oscillator with another, or two-operator modulation. If you are familiar with some of the Yamaha FM synths from the 1980s and '90s, you may recall that 6-operator FM was pretty great, 4-op OK, and 2-op only fit for the most basic bleeps and bloops. How, then, does Aalto go about making a wide variety of interesting sounds with only two-operator FM?

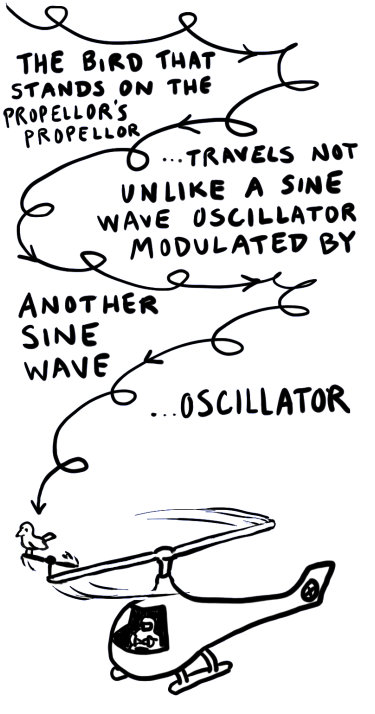
One obvious answer lies in the FILTER, and in the *timbre* and *shape*

controls we have already covered. These provide a combination of additive and subtractive synthesis that leads to a big variety indeed. But what if we didn't have these? What other timbre-changing tricks and resources can we bring to the party?

We've seen how the generators above the patcher can function as additional oscillators. By patching these to the COMPLEX OSCILLATOR controls, we can create three or even four-operator FM effects. The factory patch "deep bongo" gives one nice-sounding example of this, in which the LFO is patched to the *pitch* input of the oscillator, and controlled by the *pitch* output of the KEY module. This simple 3-op setup has the effect of modulating the carrier oscillator by the sum of the modulator oscillator and the LFO. The resulting timbres are much more complex than could be obtained with the COMPLEX OSCILLATOR alone.

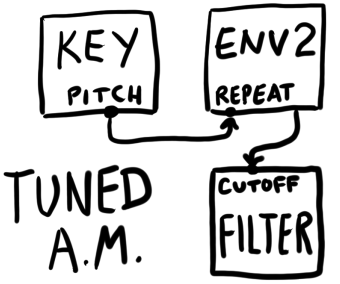
Another less obvious answer lies in modulating the filter. By patching the LFO output to the filter *cutoff,* and turning the signal input dial to maximum, we can change the filter's cutoff frequency from around 0–10,000 Hz. When this is done slowly, it sounds like a pulsing or vibrato, but at audio rates it becomes a kind of *amplitude modulation* (AM), which varies the timbral quality of the output in often surprising ways.

True AM would mean that we were varying only the volume of the output, not filtering it. However, at low resonance settings the effect of the filter modulation is indeed very much like AM. At high resonance settings, since the FILTER creates its own tone at the resonance frequency, modulating its frequency is like modulating another oscillator, and a doorway to a nether world in-between AM and FM which has not been well codified. We invite you to step in, take notes (or just save



copious amounts of patches), and as always, let your ears be your guide!

You can experiment with the LFO in the capacity of modulation source, and compare the different timbral changes that come from modulating with a fixed frequency, and varying ones. SEQUENCER and ENVELOPE 2 in repeat mode are also useful modulation sources here, and can produce yet more different kinds of timbres. Recall that by using the default setting of the signal input dial for ENVELOPE 2, you can use the *pitch* output of the KEY module to vary the frequency in tune with MIDI notes.



Note that a different kind of patch is also possible, in which the amplitude modulation is done using the GATE module and FILTER is used for an overall envelope. Here, the fast modulation source such as LFO would be patched to LPG *level,* and ENVELOPE 1 would be patched to *cutoff.* Pondering this patch, and experimenting with it on your own, is an exercise in what can be done with Aalto's intentionally limited resources. How is controlling the overall volume with FILTER like controlling it with the LPG? How is it different? What effects do the LPG *lopass* and *decay* have on the modulation?

Time, timbre, and shape

It could be said that synthesis, indeed music, is all about making audible shapes over time. Shapes define music on many time scales, from the oscillations of a single waveform, to the attack and decay of a single note, to the overall density of note events, and at as many levels of musical structure as you care to invent. In this light Aalto can be seen as a tool for making shapes into sounds. When you have a sound in your head that you would like to realize, you can start to describe it by focusing on

the shapes of its changes over time. Envelope generators, sequencers, and other modulators are all cartographical tools we can use to map out these changes. Maybe your sound starts low and ends high, or starts out like the sound of popping corn, and ends up evoking the lazily tromping feet of lunar security guards. One of the core beauties of synthesis is that, with practice, you can learn to pluck sounds from your mind's ear with a high level of precision. Alternatively, you can make some crafty decisions ahead of time, set the machine rolling, and end up with results you never would have predicted.

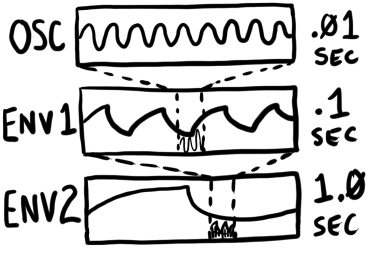
> "Musical ideas are prisoners, more than one might believe, of musical devices."
> *–Pierre Schaeffer*

Aalto gives you fluid control over shapes at different levels of musical structure. It also gives you a powerful tool, the signal dial, that lets you use your eyes as well as your ears to perceive the shapes throughout your patch.

In the patch "Aalto percussion / bongo party," there are at least three levels of musical shapes involved. First, whether you have played a key or not, the COMPLEX OSCILLATOR is always running, generating sine waves at whatever pitch it's set to. You can't see the sine waves directly when they're not sounding, because they flow through the default signal routing across the bottom modules where there are no signal dials in the path. When you do play a key, you can see them in the oscilloscope.

When the patch is quiet, the pitch is constant, but when a note is ringing out, you can see a periodic variation in the *pitch* dial. This variation is made by ENVELOPE 2, which repeats from about three to ten times a second, depending on the note being played.

Finally, the filter cutoff and the overall volume of the patch are con-



trolled by ENVELOPE 1, which makes one shape that rises and falls over about ten seconds. You can see this overall shape most clearly in the FILTER *cutoff* dial. Because ENVELOPE 2's *x env1* toggle is on, its output is multiplied by the overall volume, creating the "bouncing down" effect.

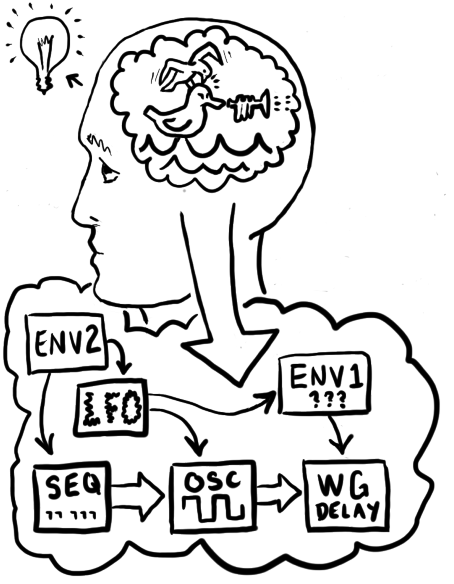
So, with nine patch cords, we've inter-organized musical shapes on multiple time scales from 1/1000 second up to 10 seconds, and added a few bells and whistles. And we have done all this without even getting the SEQUENCER involved! So there's still room to grow. By triggering the envelopes from the sequencer, and using the *key trig* toggle on the sequencer, we could compose a sequence that gets triggered when you play a note— another whole level of structure lasting up to hundreds or even thousands of seconds.

Putting it all together

Hopefully, this manual has given a good picture of the resources available in Aalto and some of the promising methods for combining them. But where you're headed with these tools is your business. We could devote many more pages to detailed recipes for soothing ambient patches and harsh death metal patches alike, but frankly this endeavor would seem counter to the spirit of an instrument designed primarily to help you get the sounds out of your head that are uniquely yours. So we won't.

We hope we leave you with enough information to begin either a methodical dialing-in of sounds you want, or an informed play that leads to unexpected results. Remember that while not all connections make sense in any given patch, none of them are harmful or "wrong," either. *A signal is a signal.* There is no difference between audio and modulation signals, except in how they are used; some of the most interesting sounds

may come from blurring the distinctions between the two categories. So feel free to let your experiments run in advance of your understanding. When you hit on something amazing, you now have the tools to save it, sort out what's happening, identify a technique, and develop on it. Happy patching!



A Frequently asked questions

Why "Aalto?"

It's the Finnish word for "wave."

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

Aalto 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 Aalto supposed to sound like this?

Probably, unless you hear an abrasive series of glitches. Here's a good way to check that Aalto 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 Aalto 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 Aalto'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.

For reference, as of version 1.2, each voice of Aalto takes around 5% of the CPU time on an 2.4 Ghz Intel i5 processor. Since all of the audio processing is constantly running, Aalto will take this time whether or not a MIDI note is playing.

If you are running the 64-bit version of Aalto 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 ***anim*** control or hiding the Aalto interface altogether will increase performance, for those times you are trying to squeeze out that last few percent and get your mixdown to happen. 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 Aalto users on this topic, visit the ongoing discussions at madronalabs.com.

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

Yes. Aalto'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 Aalto 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?

Aalto 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 Aalto securely with user data, consisting of your name and a unique ID. This is your own copy of Aalto, 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 Aalto 1.0 in version 1.2 or 1.3?

Yes. Aalto 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 Aalto, you will get errors.

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

Aalto 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 Aalto eating my CPU time?

We designed Aalto as a very general-purpose sound-making machine that behaves very much like an analog modular synthesizer. So Aalto 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.

Aalto 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 Aalto'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.

Aalto 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 Aalto. 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.