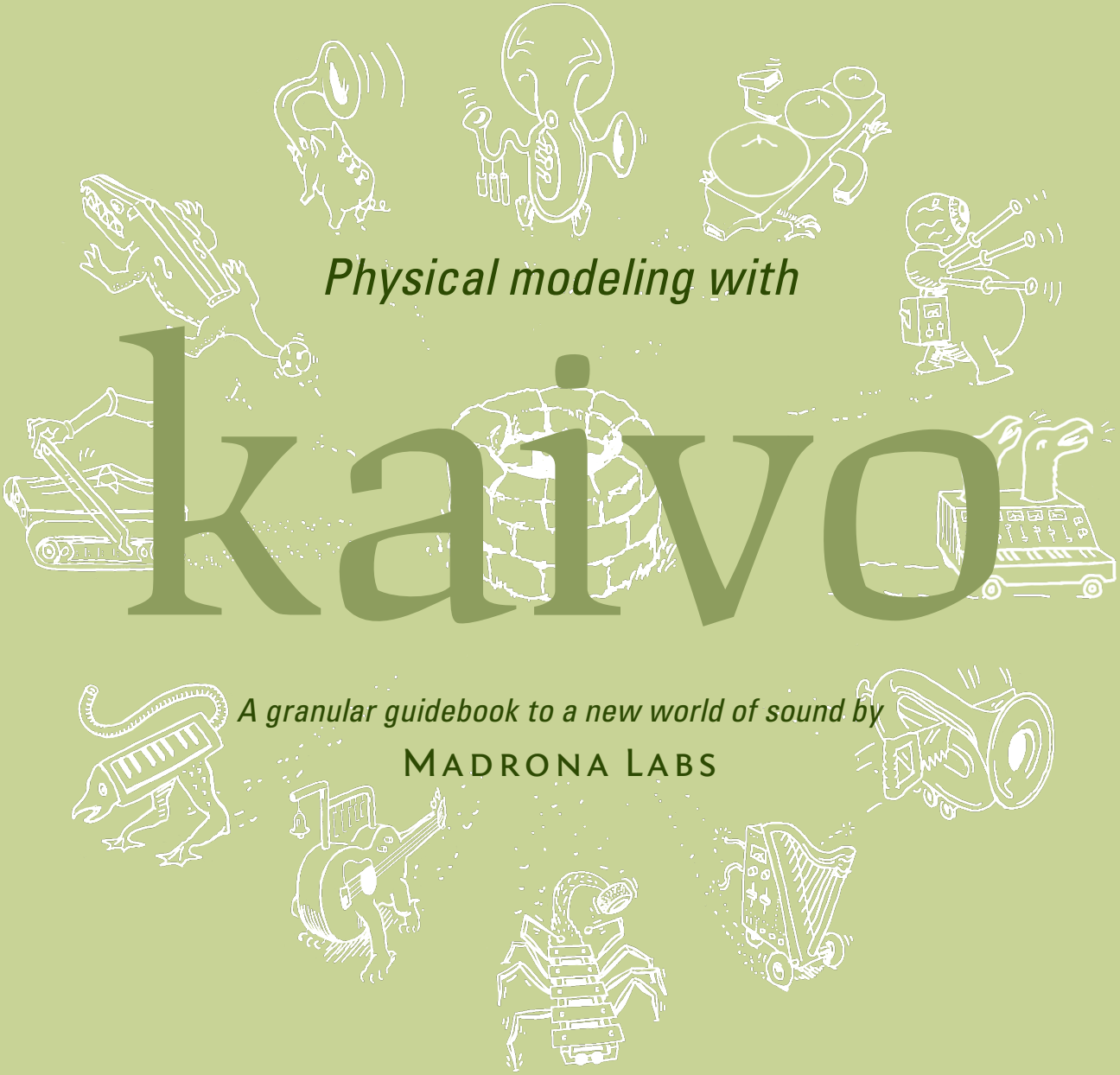


Physical modeling with

kaivo

A granular guidebook to a new world of sound by

MADRONA LABS

Version 1.0, March 2014. Written by George Cochrane and Randy Jones.

Illustrated by David Chandler.

Typeset in Adobe Minion using the TEX document processing system.

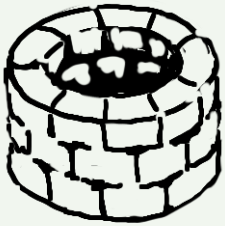
Introduction

"The future evolution of virtual devices is less constrained than that of real devices."
–Julius O. Smith

Kaivo is a software instrument combining two powerful synthesis techniques (physical modeling and granular synthesis) in an easy-to-use semi-modular package. It's laid out a bit like an acoustic instrument; the GRANULATOR module acts like the player's touch, exciting one or more tuned objects (here, the RESONATOR module, based on physical models of resonant objects) that come together in a central resonating body (the BODY module, also physics-based).

This allows for a natural (or uncanny) sense of space, pleasing interactions between voices, and a ton of expressive potential—all traits in short supply in digital synthesis. The acoustic comparison begins to pale when you realize that your "touch" is really a granular sample player with scads of options, and that the physical properties of the resonating modules are widely variable, and in real time.

For more information on the theory behind Kaivo, see Chapter 1, *"Physi-who? Granu-what?"*

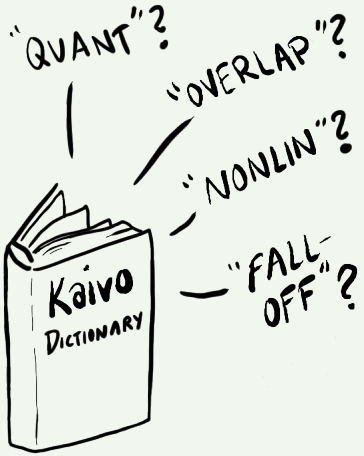


All together, this arrangement gives you room to create and manipulate sound with considerable freedom, while maintaining a pleasing sense of physical plausibility. Throw in some fresh new modulators and a dash of patching magic, and well... We're excited. We sincerely hope that Kaivo excites you, too.

A Quick Overview

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

- The *header* section, where the title of the plug-in sits, next to the patch menu and some UI behavior settings.
- The *shapes* section, where control data (and more) flows from the KEY, SEQUENCER, LFO 2D, NOISE, and ENVELOPE modules.
- The *patcher* section, where connections between modules are made.
- The *audio* section, where sound is created and processed.



For a full rundown on these, turn to Chapter 3, *"Kaivo: Module by Module."*

Read The Fabulous Manual

If you're new to all this, this manual is a great place to start. If you are familiar with modular synthesis, or have used Kaivo's elder sibling Aalto, feel free to dive in and have fun. However, Kaivo has new some kinds of modules we can safely say you've never seen before.
This manual is split into four sections:

- Physi-who? Granu-what? – A bit of background on the tech that makes Kaivo tick.

- Getting to Know Kaivo – Info on how Kaivo fits into your system, and how to use the various dials and doodads that you'll find.
- Kaivo: Module by Module – A hands-on romp through the different parts of Kaivo, module by module.
- Patching for Fun and Profit – A starter course in putting Kaivo's modules together to get musical results. Tips and tricks.

1 Physi-who? Granu-what?

This chapter is a brief primer on physical modeling, granular synthesis, and the basics of signals and sound.

For a more detailed look at physical modeling and granular synthesis with Kaivo, see the RESONATOR, BODY, and GRANULATOR sections of Chapter 3, *Kaivo: Module by Module*.

What is Physical Modeling?

It's a digital signal processing (DSP) technique that lets you process and generate signals according to mathematical models of real-world physics. Pioneered in 1960s academia, the high processing and technical demands of physical modeling kept the technique out of wider musical use until the 1990s, when it made waves in instruments such as Yamaha's VL series and Korg's Prophecy.

The technique found early use in the realistic simulation of acoustic instruments—a tough task for the more static synthesis and sampling techniques of the time. That same earthbound tendency makes physical modeling uniquely able to create wild, dynamic, unprecedented sounds, but in a manner that remains familiar to our ear. As our computers grow faster and our models more refined, more and more of the language of the physical world is laid open for us to explore.

And exploit.

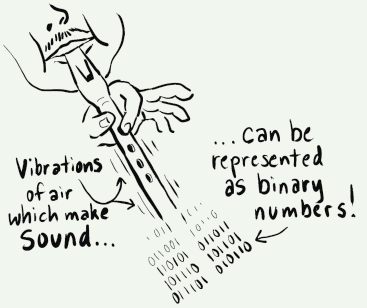
A Rose is a Rose is an Equation

The goings-on of the physical world can seem impossibly detailed and organic; more the realm of poetry than math. However, the efforts of physicists and mathematicians have shown us that much of nature *can* be described quite realistically in equations—most often *differential equations*. In our case, creating such an equation lets us use the things we know about the physical properties of an object as rules to predict the outcome of exciting that object with a given sound (as we do in Kaivo's RESONATOR and BODY modules).

Getting the results we want is just half the fun. Once we have a physical model that satisfies our initial goals, it's open season. We can adapt the model to change its properties, its behavior, and that of our interactions with it. In Kaivo, we can modulate these properties in real time, opening a whole vault of fresh new timbral and expressive abilities.

Nobody really knows why the world is quantifiable in this fashion, but it sure comes in handy when we want to design bridges that don't collapse, bouncy castles with the perfect rebound, or drones that can deliver a pizza.

"Why yes, I'd like my violin's body to change size and shape as I play it. You can do that? Wow. Can the strings randomly oscillate between gut and steel, too? Great!"



I Want My FDTD

Perfectly modeling real-world happenings with math is quite the trick in itself, and utilizing such hyper-detailed models in real time has long been impractical, given the limitations of DSP chips and computer power in general. Given that, most physical modeling instruments so far have relied on a simplified technique known as *Digital Waveguide Synthesis*.

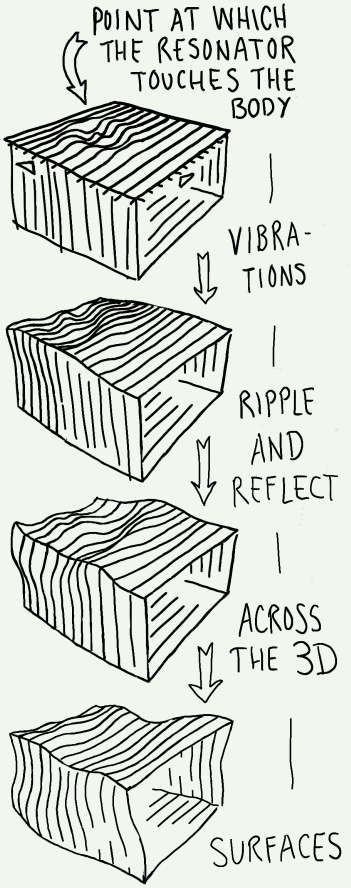
Digital Waveguide Synthesis relies on the fact that some aspects of the behavior of acoustic waves can be satisfactorily simulated with simple delays and filters, rather than more complex math.

This cuts processing expense considerably, albeit with compromises in accuracy and flexibility. In goes your signal, out comes the result, and it may not be just what you're looking for.

Today's computers have plenty of power, so it seems natural that we should move toward techniques that give us more accurate, more flexible models. That is why we chose Finite-Difference Time-Domain (or FDTD) as our modeling method for Kaivo. Without getting too techy, FDTD lets us make more natural-sounding, accurate models to play with. It also gives us the ability to feed signals in and out of the model at any point, and make localized behavioral changes, to boot. In this way, tonal variations abound, and the degree of control available is a beautiful thing.

The most basic FDTD equation used in Kaivo is the 1D wave equation. The motion of a string model, or the air pressure gradient in a modeled tube can be expressed as a single value distributed across space; so for those models, we use a one-dimensional equation.

When we move on to 2D wave equations, we're able to model the behavior of objects that require two dimensions to describe (such as drum heads or cymbals). We can finely control resonant properties across the model to simulate phenomena such as the localized damping effect on a cymbal's vibration when it's mounted on a stand. When we want to model shapes that move in three dimensions (such as a box), Kaivo uses some FDTD magic to fold a 2D model into the desired shape.



Both RESONATOR and BODY feature *nonlin* (short for nonlinear) dials. So what does nonlinear mean? Any sound can be described as a sum of sine waves at different frequencies. We call these sine waves the sound's *partials*. A linear system can modify a sound by changing the loudness of each of its partials over time. But what it can't do is change the frequencies of any of those partials.

Some examples of linear systems are volume controls, EQs and even simple reverberators. An EQ can lower some partials but not others; a reverb can extend many partials into long tails. But neither system will introduce partials at new frequencies. The only frequencies that can be output from a linear system are the ones that go into it.

In the case of Kaivo, nonlinearities refer to the tough-to-model idiosyncrasies inherent to real-world objects—aspects of their behavior that do not correspond neatly to simple, sleek algorithms. These "imperfect" properties are important to the natural *je ne sais quoi* of the objects we seek to emulate, but are really only beginning to be understood (and modeled for our benefit).

The *nonlin* dial on each of our physics-based modules is there to inject a little sonic spice, essentially emphasizing the nonlinear aspects of each model. At low settings, it can add just a little something extra, enriching signals in a euphonic way. Crank it, and things can get pretty out-of-hand. How much spice to use is a personal matter, so give *nonlin* a spin.

For more details on physical modeling and using it in Kaivo, see the RESONATOR and BODY sections of Chapter 3, *"Kaivo: Module by Module."*

De gustibus non est disputandum.

What is Granular Synthesis?

It's a crafty way to make sound using recorded audio as raw material, in a way both akin to and distinct from traditional sample playback. Granular Synthesis started as a tape-based composition technique in the pre-digital era, borne out of experiments that sought to treat tiny (< 50ms) slices of recorded audio as building blocks for new sounds, melodies, and textures.

This technique banked on the fact that if you chop a piece of audio small enough and repeat it quickly enough, you cease to hear the individual pieces. The output looks more like that of an oscillator; fast-repeating shapes that create a continuous tone. Because these shapes are sourced from audio (rather than an analog or digital tone generator) anything the composer could get on tape could become grist for the mill. This was something truly new.

In more recent times, granular synthesis has become a real-time process; less about painstakingly editing bits together in just the right order, and more about playing and manipulating samples on a micro-scale to enable new timbres and interactions. This is just what Kaivo's GRANULATOR module is all about.

In traditional sampling, when you play a note, playback starts at a place in the audio that you've specified, at a speed (and thus, pitch) that you've specified. A "playhead" of sorts travels through the audio file from the start point in a generally linear fashion, maybe does a little looping, and Robert is your aunt's husband. Even if we include modern samplers' modulation trickery, beat-chopping, and time-stretching features, at the end of the day, we're still talking about a pretty static, predictable relationship to the sample.

This section gets into some detail about what granular synthesis is all about. Some of the info is purely functional, and some is more contextual. For more functional instructions, see Chapter 3, *Kaivo: Module by Module*.

A bit like the techniques used in Musique Concrète, done on a much smaller scale.

The "playhead" metaphor here comes from the magnetic playback head on a tape recorder. As tape moves past it, the magnetic signal is continuously read at the point at which the playhead contacts the tape. Though digital sample playback is not a physical process, there is a playhead equivalent that "travels" through the audio as it plays—or in Kaivo's case, as many as *sixteen playheads* per voice.

Granular synthesis takes that model and breaks it into tiny pieces—quite literally. Instead of a single playhead per voice, bound to a static start point, pitch, and so on, Kaivo's playback relationship to a sample is much more abstract and multifarious. Don't get us wrong—GRANULATOR is happy to act like a regular sampler when needed, but there's a whole lot more it can do.

Of Grains and Windows

The tiny sections we dice audio into during granular synthesis are known as *Grains*. For all the talk of "chopping" and "dicing," what we're doing when we create a grain in digital granular synthesis is simply deciding, "*This is the portion of my audio that this playhead will play, and this is the manner in which it will play.*"

Kaivo gives you a lot of manual control over these individual decisions (such as grain pitch, panning, looping, and so on) if you want it. If you want lots of nice variance but would rather automate things or leave them to chance, Kaivo's variety of modulators (such as the SEQUENCER, LFO 2D, and NOISE modules) can help make or vary those decisions for you, over time.

That portion of the audio that a grain is set to play is known as its *window*, which makes a lot of sense, because a grain is not a discrete file, separate from the rest of the audio—it's simply the region of that audio that we can "see" through that grain's window.

Playing sections of audio willy-nilly would normally land us with all sorts of pops and clicks and harsh transitions between overlapping grains. To avoid this, the edges of each grain's window slope downward smoothly in volume. You may not hear this when playing with smaller grains, but when they get larger, you'll hear each grain fade in and out.

A Grain's Life

In much of synthesis, we're used to controls operating on what's happening now—you twist the cutoff knob on a filter, and the filter cutoff

moves. In the case of GRANULATOR, many of its controls operate not on the grains that already exist, but rather on those *yet to be born*.

Grains are independent little things, so in most cases they don't need to be told what to do once they've been created; they just keep on playing in the way they've been told to, until they expire. So when you move GRANULATOR's dials around, you're telling it, *"The next grain(s) you create should play this portion of the audio, at this pitch, with this behavior. Keep making them like that until I send you further instructions."*

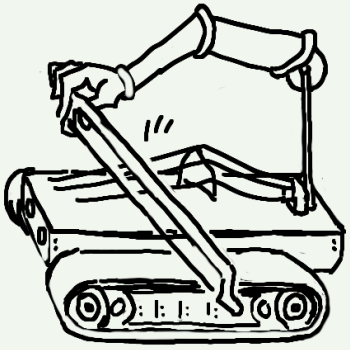
The exception to this rule is when the *follow* option is enabled, which makes existing grains change their pitch upon receipt of new *pitch* data.

There are a finite number of grains that can be active per voice at any given time. You set this amount with the *overlap* dial. For example, if you set *overlap* to 8, there are 8 grains active per voice. When a new grain is created, the oldest one expires.

It's the circle of li-ife!

Pitch vs. Rate

The *pitch* dial sets playback speed for each new grain created by GRANULATOR. This governs how quickly each playhead moves through the audio file. When you're playing back longer grains (using GRANULATOR more like a traditional sampler), the *pitch* dial and its inputs are the main way to influence the central pitch of the patch.



The *rate* dial sets the interval at which new grains are created (and old grains destroyed). When it's set to zero, grains are only created when a trigger pulse hits GRANULATOR's trigger input. When *rate* is set to 1, one grain is created every second. Set it to 2, and two grains are created per second. When *rate* is set to higher settings (each grain forming a cycle in the resultant continuous waveform), the *rate* dial and its inputs become the primary method to affect the pitch you hear.

When creating pitched parts with Kaivo, you may find it helpful to patch your pitch control signal (such as note data) to both *pitch* and *rate*. As you play notes or send control signals into a patch set this way, audio playback speed and grain creation frequency change smoothly in tandem, keeping pitch and timbre well aligned over a wide range of notes.

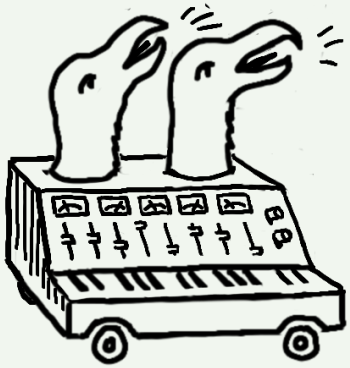
When you send a control signal to *pitch* alone, the speed of playback changes, but the rate at which grains are created does not. Controlling *pitch* with *rate* set low, you hear an evenly spaced sequence of grains, each grain's pitch set by the value of *pitch*. When *rate* is set high, its setting determines the pitch that we hear, resulting in a continuous tone that changes timbrally (yet not in pitch) in response to incoming *pitch* data. Some of these sounds have a similar character to those created with FM synthesis.

The converse is true for sending signals to *rate* alone. This changes the rate at which grains are created, but not the playback speed of each grain. With *pitch* left static, moving *rate* into the low ranges results in a sequence of individual grains, triggered at a rate set by control input. As you move *rate* higher, the grains begin to get small enough to form continuous tones, producing discernible pitches.

However, the speed of audio playback doesn't change. This means that as you move *rate* up and down, the spectral relationship between grain creation frequency and source audio frequency changes, with dramatic timbral results.

Controlling *pitch* and *rate* individually can help you create sounds with a similar character to FM synthesis. The relationship between pitch and cycle frequency is fertile ground for sonic experimentation.

Both *pitch* and *rate* come with exponential control inputs, preset at a perfect input value for reacting to note data in a sensible way.



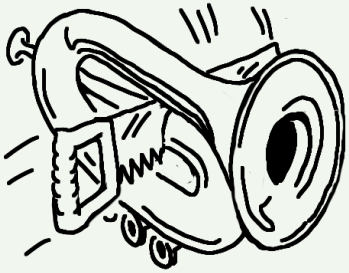
Some Grains are Bigger than Others

In Kaivo, the length of a grain depends primarily on the status of three dials at the time the grain is created: ***rate***, ***overlap***, and ***pitch***. ***rate*** sets the number of grains created per second. ***overlap*** sets the number of active grains per voice. At settings above 1, multiple grains are created, each at a length equal to the grain creation interval, multiplied by the current ***overlap*** setting. So if ***rate*** is set to 1 (one grain per second) and ***overlap*** is set to 4, each grain will be four seconds long. Low ***rate*** and high ***overlap*** makes for longer grains, and the reverse is also true.

Finally, ***pitch*** sets the playback speed for each grain, so the higher you set it, the faster the "playhead" moves through the audio file, and the more ground the grain can cover in its allotted life. The four-second grains in the example above may play through what was originally a much longer or shorter period in the source audio, depending on the setting of ***pitch***.

A grain can encompass an entire sample, or as little as a single-cycle waveform. It's up to you.

Here, length refers to the amount of a source audio file that the grain will play.



Overlapping Grains

When ***overlap*** is set above 1, each voice has multiple grains available. They are created in a row, centered around the start point you choose in the source audio. Each voice is spaced apart by the time value set by ***rate***. True to the title, as you add more grains, they increasingly overlap as they play. At a ***rate*** of 1 and an ***overlap*** of 2, you have two grains, each two seconds long, starting one second apart on the source audio—so these two grains overlap by one second.

If we increase ***overlap***, now we have four grains, each four seconds long, overlapping one another by three seconds. The layered sounds created by large numbers of overlapping grains are hot stuff for swoony pads, chattering soundscapes, and anything else that benefits from strength in numbers.

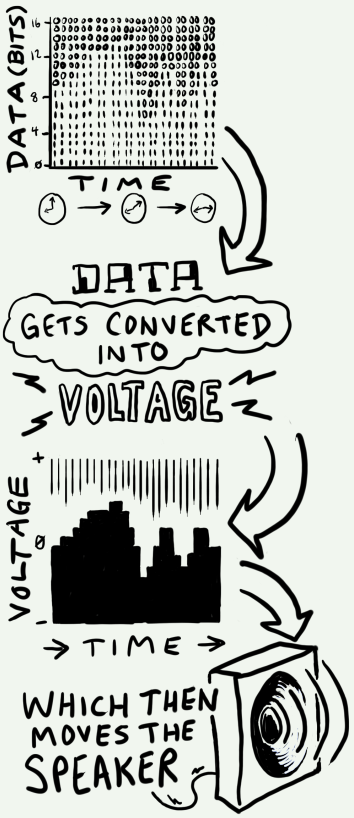
Fun with Signals

Kaivo is a digital instrument that plays and processes *signals* to make new sounds. So, what's a signal? For starters, telephone transmissions are signals. Radio waves are signals. Those huffy notes that neighbors leave on your windshield are signals. In its purest definition, a signal is the measurable embodiment of something changing, over time.

Most variable things in life change in multiple ways at once. Imagine trying to notate a dance step so that someone else could recreate it exactly. You'd need to find a way to measure and log the changes going on (such as the position of each body part over time). Real-world signals are often like this: complex and multi-dimensional.

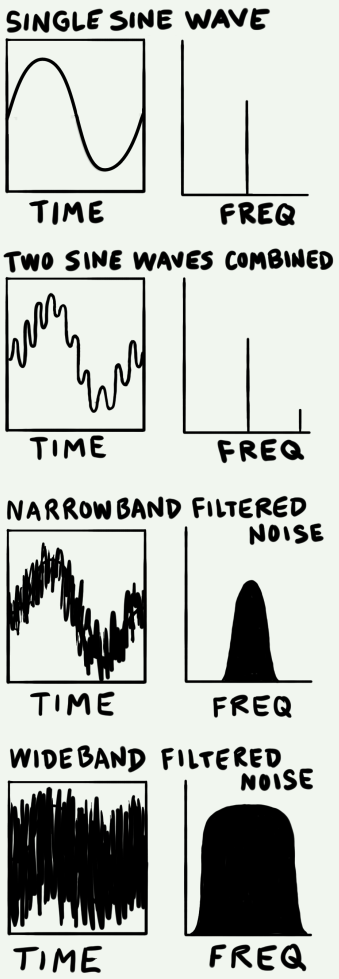
Conversely, the signals we make with machines are often conveniently one-dimensional. When we connect a computer to an audio interface to an amplifier and speaker, we can make sound by changing a single number over time. This number tells the audio interface to output a specific voltage, which it passes to the amp, changing the position of the speaker cone, which moves the air.

Our ears are incredibly sensitive to air pressure. Any tiny change can be heard, provided it is rapid enough. If the changes repeat the same shape over and over at a rate within the range of our hearing, we hear this shape as a consistent tone.



If the repeating changes are too slow to hear (unless you're a whale), you're likely looking at a *modulation* signal, which is most useful for affecting signals that you *can* hear. For example, if you modulate the pitch of an oscillator with the output of an LFO (a modulation signal if there ever was one), the oscillator's pitch moves up and down in proportion to the shape and amplitude of the LFO signal.

Every audible repeating tone has a *timbre,* or sound quality. High A on a flute sounds different from the same note played on a violin, or sung by a human, and so on. These sounds may share the same root note (and thus, the same lowest repeating frequency) but there's a whole world of correlated frequencies (known as *harmonics*) above that core frequency. The balance between these harmonics differs in every sound, and that's what determines its timbre.



Let's Get Visual

We use two kinds of graphs to show signals in this book. Some are *time-domain* graphs, akin to what you see on an oscilloscope. This type of graph describes a signal's value as a vertical movement 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. Time-domain graphs get their name from the fact that 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.

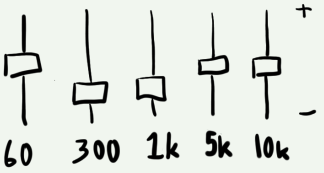
A frequency-domain picture of a sound shows all of its *partials.* Saying that a sound has a partial at a given frequency, just means that part of the sound is made up of a sine wave at that frequency. Every sound can be described as a collection of changing partials over time. Each partial is usually 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 signal graph you'll see in this manual. These pop up as diagrams in every high-end stereo ad. 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.

Kaivo was designed to be a flexible, inspiring tool for making uncommon timbres and artfully (or willfully artlessly) shaping them over time. With practice, you can use these different kinds of sound-shapes (such as time domain, frequency domain, and frequency response) to help you think about synthesis, and that can only help you wring the most out of this little green instrument.



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.

2 Getting to Know Kaivo

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

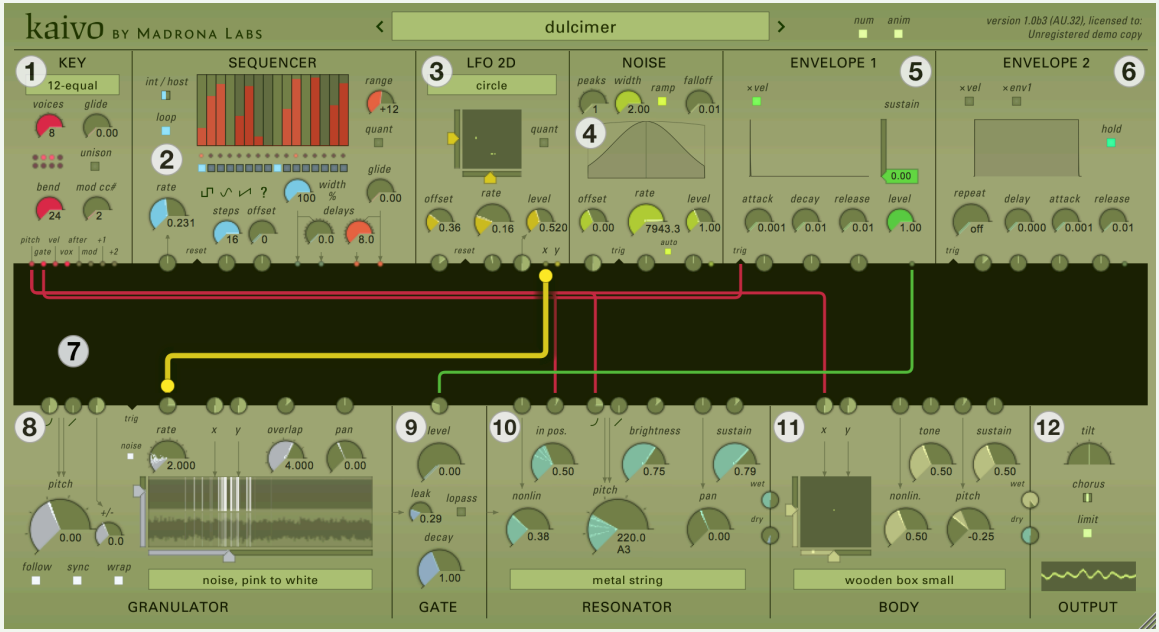
First Things First

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

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

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

An Annotated Map of Kaivo



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

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

1. KEY

   The Key module receives the note and control signals you send Kaivo over MIDI, and makes them available to the rest of Kaivo'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.

2. SEQUENCER

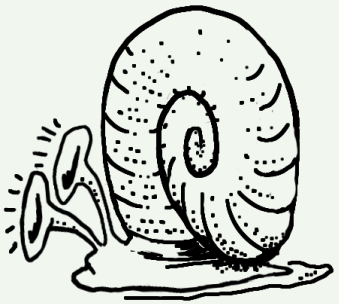
   The Sequencer module creates both arbitrary linear functions and patterns of rhythmic pulses over time. You can use it to create repeating melodies, control changes, rhythms, or any combination thereof.

3. LFO 2D

   "LFO" stands for Low Frequency Oscillator, and just about every synthesizer has one. "2D" stands for "two-dimensional," which is just what LFO 2D is. Whereas a "normal" LFO outputs a single changing value, LFO 2D traces shapes in two dimensions and outputs two signals—one for the y axis, and one for the x axis. Shapes from simple sine waves to circles, triangles, and even rain (!) are available at glacial-to-audio rate ranges.

4. NOISE

This module produces *uncertain signals*—some call it noise. Far from your average white-pink-whatever, NOISE provides an armload of ways to shape the probability distribution of the noise it produces, delivering a boatload of creative possibilities. All the way from drunken ambling to 1000–grit sheen, you'll have fun with this one.



5. ENVELOPE 1 and

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. New in Kaivo are the dedicated TRIG inputs on several modules, allowing for discrete handling of trigger signals.

8. GRANULATOR

This module is where much of Kaivo's sound begins. It's a sample-based granular sound source with a variety of modulation inputs and some crafty sonic tricks up its sleeve. Kaivo comes with a set of useful audio bits to load into the module, but things *really* start to get going when you import your own sounds. This module enables granular synthesis, traditional sample playback, and much more.

You can create and import multi-channel audio files into Kaivo, giving you the ability to move GRANULATOR's "playheads" across them

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 2D to change the level (VCA mode) or the low-frequency cutoff (LPG mode) of the synthesizer's signal.

10. RESONATOR

One of the tasks we had while developing Kaivo was creating some good-sounding physics equations that describe the effects of passing vibrations through resonant objects (such as strings, pipes, plates, and chambers). The RESONATOR module lets you apply some of those equations to the sounds you make with GRANULATOR, as well as make changes to the behavior, shape, tone, and pitch of these resonators.

This not only opens a world of cool new tone colors, but actually introduces a natural sense of variety as you play—a string played while at rest sounds different than a string played while vibrating, and the same is true here.

There is a separate resonator for each voice of polyphony you use. They act somewhat independently, though they are connected to one another through the BODY module, as you'll see below.

11. BODY

While a stringed instrument often has multiple strings (played here by the RESONATOR module) it has but one Body, and this is that—a two-dimensional (notice a theme here?) physics-based resonator module, specializing in chambers and plates. You can move signals around within the model in an X/Y manner, shift things all around timbrally, and modulate the snot out of it.

The pleasant physicality of the RESONATOR can be found here as well. "Vibrations" coming from one of the voices vibrate the body resonator, which in turn subtly stimulates the other voices' resonators—just like you get when you really whack the 'E'-string on an acoustic guitar and keep your left hand off the neck.

Some of the BODY models are three-dimensional (such as the boxes). These are created by "folding" 2D models into 3D configurations.

12. OUTPUT

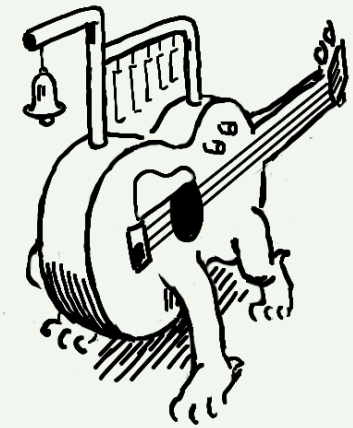
This is the final line of defense/manipulation the signal will pass through before you can hear it. It includes a classic one-knob "Tilt" EQ—turning the knob one way cuts bass and boosts treble, and the other way, vice-versa. Also found here is a basic-but-useful Limiter, to keep your unruly signals at bay, and the Oscilloscope, which gives you visual feedback about the waves you're making.

Presets

We labored heavily to make Kaivo simple and inviting to use, but flipping through the preset sounds first is obviously a good way to hear what Kaivo can do. Kaivo 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 Kaivo 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:

- Kaivo keys: Tuned sounds that respond well to the keyboard. Some verge toward acoustic sounds, while others are more synthetic.
- Kaivo pads: Tuned atmospherics and spacey sounds, with long attacks and decays. Good for floating, flowing parts.
- Kaivo machines: Patches with ideas of their own. Some ignore keyboard input entirely, and some require host clock to run.
- Kaivo percussion: Patches that dole out rhythmic sounds with dispatch.
- Kaivo techniques: Simple patches that can help you learn how to use Kaivo.
- Kaivo textures: Complex, cinematic environments that evolve.



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, Kaivo'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 Kaivo 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.

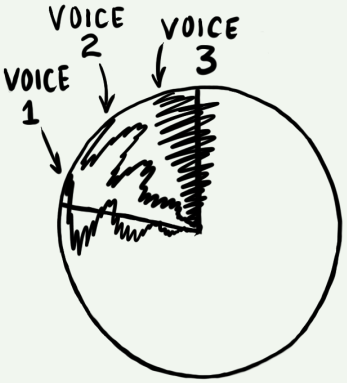
Each dial controls a signal that can be modulated, and shows the most recent sample values of that signal every 1/30th of a second. That means that static or slow-moving modulation will cause the pointer to stay still or move slowly back and forth.

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 Kaivo 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 GRANULATOR 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 Kaivo.



Detents

Some dials, such as the GRANULATOR pitch, have *detents.* Detents are useful default positions. For example, the pitch knob has a detent at an A note in each octave (110Hz, 220 Hz, 440 Hz...) to keep the GRANULATOR 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 Kaivo'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, so just get that crazy idea out of your head.

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

The *num* toggle in the header lets you turn off all the numerical displays, if you'd rather not see them.

Dial Scales

While many dials are linear (the change per degree from high to low is constant), some dials have logarithmic scales where the change is much larger as the value gets higher. This was done in cases in which a logarithmic scale matches the changes you perceive better than a linear one, as in oscillator pitch, for example. In a logarithmic scale, equal movements of the mouse in different positions on the dial will produce differently-sized changes. For example, 55, 110, 220 and 440 Hertz are all equally spaced apart on the *rate* dial in the GRANULATOR module.

Linear "Dials"

Some controls, such as those pertaining to X/Y position and Env 1 sustain, made much more sense as linear 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.

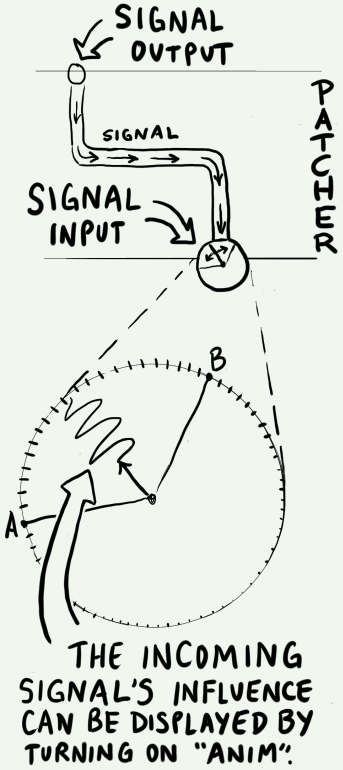
Using Buttons and Switches

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.

Using the Patcher

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

The Patcher is the large dark central area surrounded by all the modules. It lets you patch signals from the outputs of modules to the inputs of modules by drawing patch cords. Each patch cord has an arrow on it that shows which way the signal is flowing. Note that though signals tend to flow down, from ENVELOPE 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.

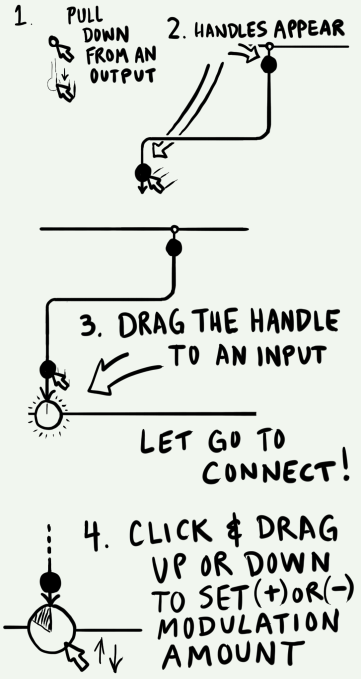
You can use the LFO 2D to see this, even without using any patch cables, as follows: turn the *rate* dial on LFO 2D to 1.0. Double-clicking the dial will do the same thing, because 1.0 is the dial's default value. Now, turn the *level* dial up towards 1.0. You can see the LFO 2D signal output lights pulsing more and more brightly.

Signal Inputs and Modulation

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

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

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



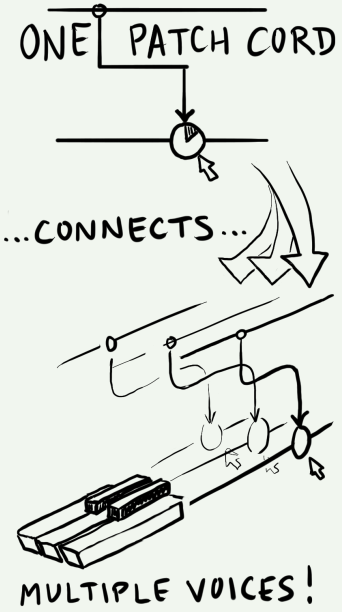
Trigger Inputs

Kaivo's sequencer, LFO, noise source, envelopes, and GRANULATOR can all accept *trigger* signals, and they do so through triangle-shaped inputs labeled "reset" or "trig." You patch signals into these inputs just as described above. The big difference is that there is no dial to set the level of trigger inputs—a signal either triggers the intended response, or it doesn't. For this reason, you'll want to make sure that the signals you patch into trigger inputs are appropriately "triggery," whether they're purpose-built trigger signals like the KEY module's *gate* output or SEQUENCER's trigger outputs, or a particularly spiky audio or control signal.

Patching

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

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



Multitudes Within Multitudes

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

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

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

Removing or Repatching a Cable

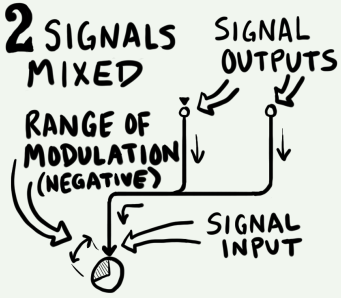
When a cord is selected, its *handles* are visible. Handles appear as circles at either end of the cord—you can drag them to move the ends. If multiple cords are selected starting or ending at the same place, clicking the handle there will move all of the selected cords.

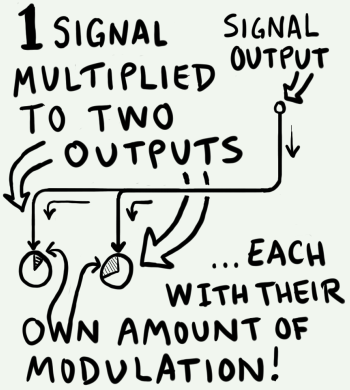
Typing the delete key should delete all the selected cords. You can also delete a patch cord by dragging either end to a place with no input or output. An X will appear instead of the handle at the end, and POOF. It's gone. Again, the changes happen in real time for any currently held notes.

If you create a new copy of Kaivo in your DAW, it will appear with the preset "default" selected: two patch cords that make a simple, playable patch. The KEY module's pitch output is connected to the pitch input of the GRANULATOR, and its gate output is connected to

the GATE's level input 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 GRANULATOR, which is calibrated to play the appropriate pitch in response.

When a key is played and held down, the KEY module's gate output goes high and remains there until the key is released. This value is sent as a signal through the other patch cord to the GATE module, which lets through the signal from the GRANULATOR 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 2D, 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.

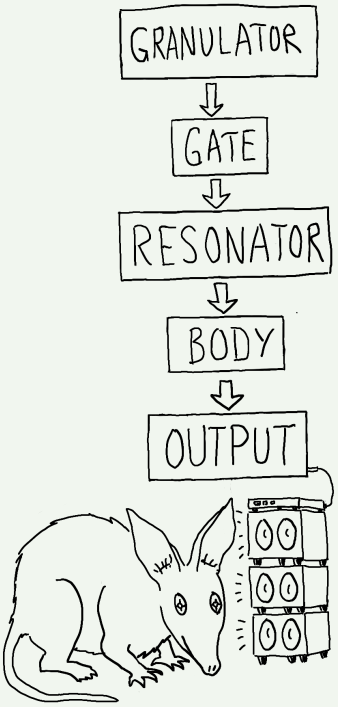
Default Signal Routing

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

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

The pre-patched modules below the patcher can be called the sound modules. While the modules above the patcher generate primarily inaudible control signals, the ones below are where the sounds you hear are made. Of course, this being Kaivo, there are exceptions to this rule, and you can find ways to use an envelope as an oscillator, or the GRANULATOR as an LFO.

You might visualize the bottom row of modules as the players in an orchestra, and the top row of modules as their conductor. Or wait, a row of conductors, each conducting different aspects of each section. Yeah, that works.



3 Kaivo: Module by Module

This chapter will take you on a more detailed tour of the various modules that compose Kaivo, 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 4, *"Patching for Fun and Profit."*

The Header

The header mainly deals with patch 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 Kaivo 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 Kaivo 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 Kaivo. This can be useful when recording multiple takes of Kaivo dial-twiddling as audio in Ableton Live. In the Clip View for the MIDI clip you’re working with, set the “Pgm” parameter to 128. Each time that MIDI clip is launched (with its launch button or a stop-and-start of the transport), Live sends program change 128 to Kaivo, reverting the patch to

its saved value. This gets you back to a consistent starting point for the next recording pass.

"Copy to Clipboard" and "Paste from Clipboard" let you copy the current patch's data to the clipboard, then paste it, either to the same instance of Kaivo (to reset all controls to their copied value), another instance within the same set/session/whatever, or to a instance in a different set/session/whatever.

If you're copying a patch you care about to the clipboard, be sure it's saved traditionally, as well. The clipboard is a busy place on most computers, and it'd be a shame to lose a vital patch state because you took a break to share a kitten photo online.

The second section of the menu displays your personal presets. Some user presets, contributions from fellow Kaivo users, are installed here by default. If you're trying to copy your preset files from Windows Explorer, be aware that even though it's the recommended path for user data, Windows makes this directory invisible by default.

The third section contains the factory patches installed by the Kaivo installer.

On Mac OS, user and factory patches are stored in (Your home directory)/Music/Madrona Labs/Kaivo. On Windows 7, it's in C:/AppData/Roaming.

Selecting Patches with the Magic of MIDI

If you'd like to be able to load patches by sending MIDI program changes to Kaivo, 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: (Your home directory)/Music/Madrona Labs/Kaivo
- Windows: (Your home directory)/UserData/Madrona Labs/Kaivo/Samples

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 Kaivo on startup, and the presets in it are assigned numbers, in alphabetical order.

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

To rearrange the programs, give them new names so they are in a different alphabetical order. Send a program change to Kaivo that corresponds to your chosen patch, and Kaivo 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.

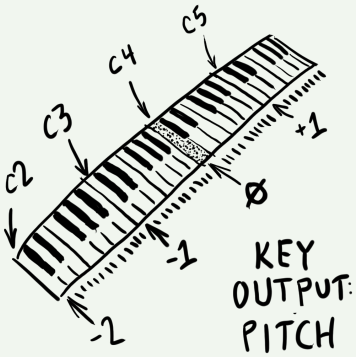
Kaivo's display is fully vector-based, and thus can be resized freely, to suit your needs or mood. Drag the handle on the lower-right corner of the window to resize it.

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 Kaivo you are running, as well as your registration info. When you purchase a copy of Kaivo 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 Kaivo if any problems arise, and to maintain and improve it.

KEY



The KEY module receives all the MIDI data you send Kaivo's way, and turns it into useful control signals that you can route to other modules with the Patcher.

Tuning Menu

The menu on the top selects the tuning table that Kaivo 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 Kaivo. 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

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

- *bend:* This control lets you set the amount that the *pitch* output varies when MIDI pitch bend messages are received. It is calibrated

in semitones from zero to 24. Yes, 24-count-em-twenty-four semitones, which really means a range of 48 (+- 24), or four octaves. Can you handle such power? No? OK, then set it to 7, or something.

- *unison:* This button lets you toggle Unison mode on or off. Unison mode combines all four voices into one monophonic voice, which can make for some very big sounds. If multiple oscillators at exactly the same pitch are added together, the result can sound quieter than a single oscillator because the waveforms cancel each other out. This is hardly ever what anyone wants, so Kaivo'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 Kaivo 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 pitch signals Kaivo can use. When MIDI note C4 is played, the pitch signal output is 0. C5, an octave higher, outputs the value 1, and C3 outputs -1. Another 1 is added or subtracted for each octave up or down. We chose this scaling so that keyboard input maps naturally to all 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.

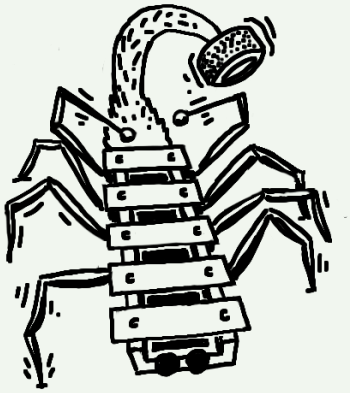
In the patcher, any input dials that control pitches, such as GRANULATOR and RESONATOR *pitch*, are all calibrated so that when you connect a pitch input and set the default scaling (double-click), they will track the same frequencies or intervals according to the 1.0 per octave standard.

- *gate:* This output sends a full-scale signal when a note is played, and when the note ends, it stops. This output is akin to a gate output on a CV/gate-enabled keyboard or MIDI converter. This signal is perfect for driving Trigger inputs (such as the *trig* input on Envelope 1 and 2), or for any instance where you'd like to control something with the on/off state of incoming notes.
- *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!
- *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.

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 Kaivo by sending messages from a MIDI knob or fader controller.

- *+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.

SEQUENCER



The sequencer is a powerful control and trigger generator, capable of creating sequences of control signals for pitch, timbre, and other parameters, 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 Kaivo.

Each of Kaivo'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, Kaivo 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, Kaivo'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 Kaivo'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.
- *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.

With *loop* turned off and a trigger signal patched into the *reset* input, 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.

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 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!

- *reset:* This isn't a "control" per se, rather a trigger input that resets the sequencer back to the first step when it receives a valid trigger signal. This is quite useful for, say, rhythmically triggering fast sequences, when using the SEQUENCER as a complex envelope generator, for restarting the envelope upon note input, as described in the *loop:* blurb above.

- *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.

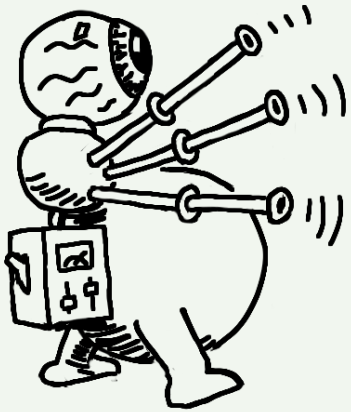
Internally, the sequencer generates a phase signal that increases continuously from 0 to the number of steps, then wraps back to 0. Within each integer step, if the fractional part is less than the pulse width, a 1 is sent to the trigger output. Otherwise, a 0 is sent.

- *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 the currently selected scale in the KEY module, 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

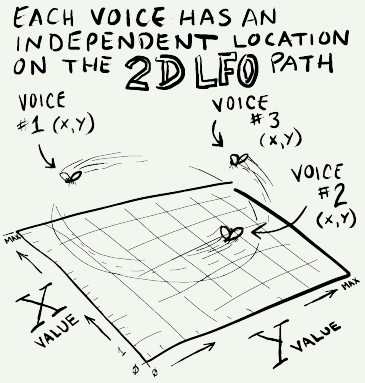
The two blue outputs are trigger outputs. When the sequence step is within the first part of a step selected by the *width* control, the trigger value is 1, otherwise it is 0. The two orange outputs are value outputs. They send out the values set by the multi-sliders for each step, as processed by the *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 2D

This module is a Low-Frequency Oscillator—one of the canonical modulation sources in synthesis. Practically every synth has an LFO, but they *don't* have a two-dimensional LFO! So what makes this LFO so 2D? Whereas your average LFO outputs a single oscillating control signal, LFO 2D traces 2D shapes over time, and outputs two changing values—one for the Y axis, and one for the X axis.

If you're into math, you can think of LFO 2D as a 2D function (x, y) with independent parameter *t*.



At slow rates, the position of each voice's LFO output can be seen moving as small "sparks." At high rates, the sparks blend together to show a clear picture of the shape being output.

Controls

- Menu: Select the LFO pattern from a variety of 2D shapes. These range from traditional LFO shapes translated into 2D space, to some pretty crazy-pops choices.

- X/Y display and X/Y offset sliders: This display shows the pattern of movement currently specified by the chosen shape, offsets, rate, and level values. Sliders to the left and bottom of the X/Y display let you shift the LFO's output in X/Y space.

- *quant:* When enabled, the X and Y outputs are quantized to match the currently selected scale in the KEY module.

- *offset:* Adds an temporal offset between voices as they travel 'round the LFO's path. If you imagine the voices as cars on the same winding track (the pattern), each voice can drive a fixed distance behind or in front of the other ones.

This offset can be different for each voice, when driven by a multi-voice source such as the KEY module's *pitch*, *vel*, or *after* outputs.

- *rate:* This dial adjusts the frequency of the LFO cycle. This dial can accept control signals, even from LFO 2D's own 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.
- *reset:* This isn't a "control" per se, rather a trigger input that resets the LFO to its initial value when it receives a valid trigger signal.

NOISE

This module produces *uncertain* signals—wild, pseudo-random thrashings that some call noise. The controls provide a myriad of ways to shape the probability distribution of that noise, which provides a myriad of modulation possibilities.

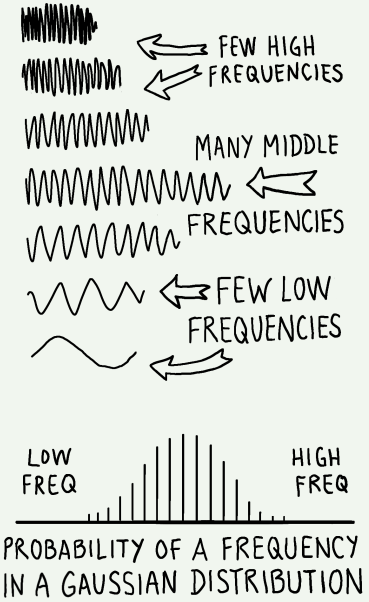
While we're used to hearing "noise" as a high-frequency wall of sound (really just a fast succession of random values), this module lets you set the rate at which it updates, and even slew between values, which opens up a lot of neat possibilities at slower rates. Suddenly we get beautifully fluid, unpredictable control shapes, swooping from hither to thither. Just the move for those tape-like pitch fluctuations and subtle timbral monkeying about.

The noise output is based on a combination of Gaussian distributions, which are a fundamental kind of distribution in math and nature; essentially the average of any kind of decision made multiple times—for example, flipping a coin.

The graph in the NOISE module shows the probability distribution of the output signal. If only one sharp peak is shown, that means only one value will ever come out: the X (left-to-right) location of that peak. As you make the peak wider, the Y (top-to-bottom) value of the graph describes the probability of the output being equal to the X location of the graph.

Well, that was a bit of a mouthful. Suffice to say that experimentation, especially with *rate* set low to show off the subtleties of the movement, will tell you everything you need to know.

Fun Fact: The shape of a Gaussian distribution is commonly called the bell curve.



Controls

- *peaks:* Lets you set the number of Gaussian peaks in the distribution.
- *width:* Lets you control the width of the peaks.
- *ramp:* Enable this option to smooth the transition between values when a new value is computed. When this option is disabled, the output jumps straight to the new value at each re-trigger.
- *falloff:* More probability stuff! This lets you set the maximum probability of the peaks as they fall away from 0 on either side.
- *offset:* Lets you set the the value of the central peak.
- *rate:* Lets you set the rate at which the module generates values, and the rate of the ramp to the next output value, if *ramp* is enabled.

- *level:* Lets you set the final noise output level.

- *trig:* Send trigger signals to this input to compute a new value whenever you like.

- *auto:* Enable this to compute new values at the rate set by the *rate* dial. Disable it to trigger new values only when the *trig* input is stimulated.

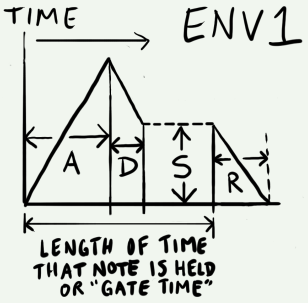
ENVELOPE 1 and ENVELOPE 2

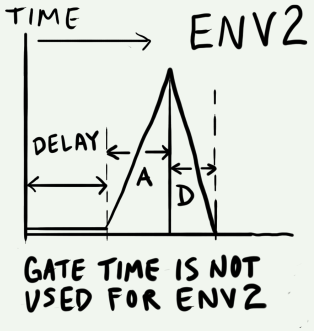
Kaivo has two envelope generators (lucky us!). Each is designed to complement the other by doing some special things that the other one can't do. Each has only one output: a time-varying signal that is triggered by an input signal. Typically an envelope is used to control the overall loudness of a note 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* dials 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 *repeat*, *delay*, *attack*, and *release* dials have inputs for control signals.

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





The *x vel* control on each envelope multiplies that envelope's output by the velocity of incoming MIDI notes. 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 1'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.

GRANULATOR

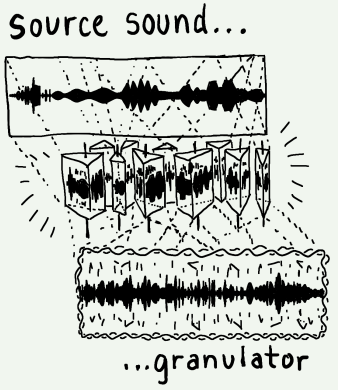
This module is the nerve center for sample playback and manipulation, and the closest thing in function that Kaivo has to a traditional synth oscillator. GRANULATOR takes the audio you load into it, and creates *grains* (smaller subsections of that audio) to play. If the grains are very small, they can be repeated quickly to create sustaining tones. Larger grains play larger portions of the audio.

The width of each grain is determined by the settings of the ***overlap*** dial (which governs how many grains can be active at a time, per voice), the ***rate*** dial (which sets the number of grains created per second, per voice), and the ***pitch*** dial (which sets the playback speed and basic pitch of each grain).

Each individual grain (of which there can be dozens active at a time) can be set to play at a specific pitch, starting position, looping state, and so on, even when they're being created hundreds of times per second. In this way, GRANULATOR acts a bit like a stack of samplers working in tandem, all operating from the same source audio, each dealing with that audio in its own distinct way. This power to determine the character of each grain as they are created is in your hands, and at the mercy of all modulators available in Kaivo.

In the grand scheme of this instrument, GRANULATOR is here both to make sound all its own, and to strum and pluck the RESONATOR and BODY models into action. Your choice of sounds and the way you use them have *MASSIVE* effects on the feel and sound of the whole biscuit, and there are no wrong moves (besides, perhaps, entirely silent patches).

This is a functional run-down of the GRANULATOR module. For more in-depth information on granular synthesis and how GRANULATOR relates to it, see Chapter 1, *Physi-who? Granu-what?*



Waveform Display

The main display shows the waveform(s) of the currently loaded sound. Multi-channel audio files are shown with a row for each channel. The X-axis slider (horizontal) lets you set the center point in time from which grains are created. Any incoming control data will move the grain creation point fore or aft of that setting.

Much like the "start" parameter in a traditional sampler.

The Y-axis slider (vertical) lets you set the center point in the vertical stack of channels at which new grains will be placed. Incoming control data moves the grain creation point up or down from that setting. If your audio file is mono, the Y-axis slider won't do a thing. The position of each currently active grain is shown as a small vertical line.

If you set the Y-axis between channels with the slider or control input, Kaivo will create new grains that play a mix of those two channels.

Controls

- X/Y offset sliders: As discussed above, these set the X/Y starting points for the creation of new grains. X affects their place in time, and Y affects their position in the stack of audio channels. If you modulate these parameters, you will see the results of that modulation as small white lines moving around near each slider. That way, you always have a sense for what position the next grain will start from.

- Source menu: This menu displays the name of the current sound. Click the menu to choose from the list of installed sounds, or to import your own with the *Import..* command. You can import any sound you like, up to 8 seconds in length. Longer sounds will be truncated to 8 seconds, ruthlessly.

Sounds are stored on the hard disk in this directory:

- Mac OS: (Your home directory)/Music/Madrona Labs/Kaivo/Samples
- Windows: (Your home directory)/UserData

Keep in mind that any grains that are currently active when you choose a new sound (or patch) will continue to play their portion of the previously-loaded sound until they expire. This lets you seamlessly transition between sounds and patches as things play, or stack up loads of different textures when making a granular cloud.

For for more information on cloud-making, see Chapter 4, *Patching for Fun and Profit*.

- *pitch:* Lets you set the playback speed of grains as they're created. When working with long grains, this dial and its inputs are the way to set pitch. 0 is the natural pitch of the source audio. Labeled in octaves above and below, from -3 to +3. The *pitch* dial has two inputs: Exponential, for pitch control from the sequencer or keyboard, and Linear, for linear FM.

Yes, you can do FM synthesis with this Granulator!

- *+/-* Lets you dial in a positive or negative-going amount of pitch envelope—a mini-env that controls the pitch trajectory of the GRANULATOR over time, as a linear ramp from 0 to the dial setting. This control has a range of -2 to +2.

- *trig:* Grains are created automatically at the rate set by the *rate* dial. If you set that dial to 0, grain production stops. thus halting the production of new grains) you can trigger the creation of a new grain by patching a trigger signal to this input. A brand-spanking new grain is created whenever it receives a trigger. Each special, hand-triggered grain also has a bit of attack added to it, so that note starts remain crisp.

- *rate:* Sets the rate at which grains are created, from 0 - 880 times per second. When this rate is 0, new grains are only created when a trigger is received. When working with shorter grains (the kind that you might want when using GRANULATOR to create tones, rather than play long samples), the *rate* dial is the main way to set note pitch.

Toggle the nearby *noise* button on to modulate *rate* with its built-in noise source. We find this comes in handy for making organic sounds, and lets you reserve the NOISE module for more advanced uses.

- *x and y* input dials: These provide offsets and precision scaling for control inputs patched to X (time) and Y (channel mix) position.

- *overlap:* Lets you set the number of grains that can be active simultaneously. When overlapping is called for, multiple grain windows are created, centered around the position of the X axis offset slider. The length of each grain is expanded, multiplied by the *overlap* value you set. All these now-overlapping grains sit next to one another, spaced apart by the time value set with the *rate* dial.

  With an *overlap* of 1 and a *rate* of 1, grains are created one at a time, one second per grain. If we boost *overlap* up to 4, we now have four grains at a time, each four seconds long, each positioned one second from the next. The center of this group of grains is still set with the X-axis offset slider. When *overlap* is set below 1, each grain only plays for a fraction of its normal cycle according to the dial setting. At high rates, this makes for cool choppy, ripping synth sounds, and at low rates, it inserts rhythmic pauses into playback.

- *pan:* This lets you control the stereo position of new grains as they are created. Each grain stays where it's put in stereo once it exists, so by varying *pan* over time, you can create some wonderful spatial effects.

- *follow:* Enable this to make all of each voice's grains shift their pitches when that voice is sent *pitch* data. Toggle it off, and grains stick to their starting pitches until they expire.

Enabling *follow* makes GRANULATOR act a bit more like a traditional oscillator. When playing longer grains with *follow* disabled, sounds tend to evolve, cloud up, and shift subtly between pitches as successive notes are played.

- *sync:* Enable this to snap new grains' starting positions to the nearest smooth-transition spot in the waveform. This helps to keep grain playback in sync, especially when working with single-cycle oscillator sounds. Disable it when grain synchrony is not important, or you just don't feel like it.
- *wrap:* Enable this to allow grains to loop through the source audio continuously. When *wrap* is disabled, any new grains that are created will expire if they reach the end of the audio file. Grains created while *wrap* is enabled will play indefinitely, looping through the entire audio file, until they are replaced by new grains.

Creating grains with *wrap* enabled can lead to a "stuck note" sort of condition. While they'll eventually expire when replaced by new grains, you can also turn *overlap* down momentarily, to deactivate stray grains.

GATE

The GATE module is a dynamic volume control, akin to the VCAs (Voltage Controlled Amplifiers) found in modular synths. Its takes its input from the GRANULATOR. Its output is sent to the RESONATOR. 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 an emulated vactrol with a variable decay, which opens up a world of cool, percussive envelopes.

The GATE is the main tool used to sculpt the dynamic profile of Kaivo's sound, but it can have spectral effects as well. This happens when you flip the GATE into *lopass* mode, or play with the *leak* feature...

A vactrol is an electronic device, usually a pairing of an LED (or little bulb) and a photosensor. Its light-driven attenuation lends many Low Pass Gates (such as our GATE) their delicious dynamic response.

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.

- *leak:* This sets the amount of simulated DC leakage that you want. The more you turn it up, the more leakage occurs and the more low end disappears. At high levels, obvious noise is added, for that *capacitors set to vent* panache. Higher settings can juice up your signal nicely and help give them greater texture with which to stimulate the RESONATOR and BODY models.

- *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, thank goodness.

RESONATOR

This module acts a bit like the strings on our proverbial violin. Every voice Kaivo plays gets its own RESONATOR, which applies physics equations that describe the real-time reaction of resonant materials to incoming signal. With it, you can harness the character and behavior of modeled strings, springs, pipes, and other things.

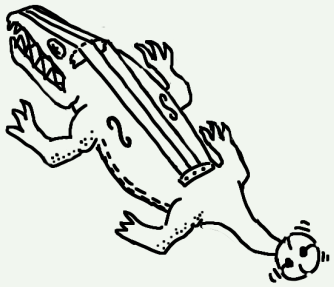
This is a functional run-down of the RESONATOR module. For more in-depth information on physical modeling and how RESONATOR relates to it, see Chapter 1, *Physi-who? Granu-what?*

RESONATOR acts a bit like the filter in an analog synth, but can also act like an oscillator, both adding and subtracting frequencies from the input. We call this kind of behavior *nonlinear*, and we like it.

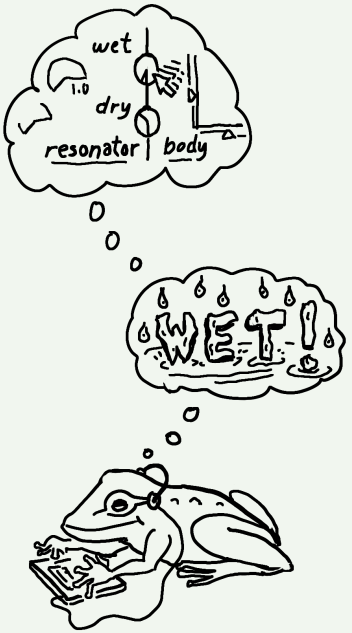
Controls

- Mode menu: Lets you select the resonator mode of your choice. Choosing a mode loads the physical modeling goodies needed to create the effect, and makes a few behind-the-scenes tweaks to the *brightness*, *sustain*, and *nonlin* dials, tailoring their response to the chosen mode.
- *nonlin:* Lets you smoothly vary a combination of nonlinear effects that contribute to the "flavor" of each mode.
- *in pos:* Lets you set the position within the resonator model at which it is *excited* by incoming audio. Changing this dial will produce a shifting spectrum of resonant frequencies in the model.
- *pitch:* The pitch of each resonator can not only track the keyboard or other control input, but can be modulated continuously and even radically to produce effects reminiscent of FM synthesis. Applying FM to the pitch of a physics equation? We're really off in space, here.
- *brightness:* Lets you control how bright the model sounds by causing high frequencies to be damped more aggressively than lower ones as you increase its value.
- *pan:* Lets you precisely position the RESONATOR's output in the stereo field.

Each mode starts as a fairly literal model of a physical sound. Then the controls give you ways to warp its response into other natural or unnatural places. It's all just math, but some of that math produces real-world responses, and some do not.



- *sustain:* Lets you set the time for which the model will "ring out" after an input signal is applied. Higher settings exaggerate resonant effects in sometimes-wonderful ways.
- *wet* and *dry* dials: These let you specify the amount of dry signal (from the GATE module) and wet signal (from the RESONATOR effect) you wish to pass on to the BODY module.



RESONATOR Modes

- metal string: A simple bright string, ideal for guitar-type sounds
- nylon string: Another string, but with a mellower sound.
- gut string: A mellow string with some nonlinear raunchiness.
- small chime: A small, bright, tuned piece of metal, with many inharmonic partials.
- medium chime: A simple hanging bar, with harmonic partials. Tube-like character.
- large chime: A bigger, mellower metal chime, with lots of complex harmonics.
- small spring: A small, tuned sproingy thing.
- large spring: A big sproingy thing. Tuned. Sort of.

BODY

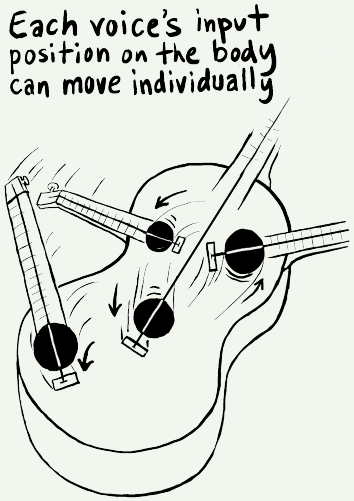
At first blush, the BODY module may remind you of the RESONATOR. They're both resonant effects based on physical models of vibrating objects. However, if RESONATOR was a little like the strings on our violin, BODY is the body. There's just one of them (not one for each voice like RESONATOR) and it specializes in chamber and plate resonances. It's here to bring all of the voices together in space.

Though the BODY models have names like "small wooden box" and "metal plate" these descriptors only really describe each model when played at default settings. Think of the names as a starting point for the sounds each model can make—then twiddle on.

You can freely move the point at which each voice's RESONATOR model contacts the BODY model, in two dimensions (X and Y). Each part of the BODY model responds and resonates differently, much like striking a drumhead at the edge sounds different than at the center.

This is a functional run-down of the BODY module. For more in-depth information on physical modeling and how BODY relates to it, see Chapter 1, *Physi-who? Granu-what?*

A few folks reading this just looked at the X/Y outputs of LFO 2D and the X/Y inputs of BODY, and said, "Oooooh..."



Controls

- *x* and *y*: Each RESONATOR is connected to the BODY at a set location, where its energy feeds into the resonant model. Using the *x* and *y* controls and display you can view and set each resonator's contact point, and modulate those positions endlessly.

- *nonlin:* Lets you smoothly introduce a combination of nonlinear effects that add a bit of sonic spice to the response of the current model.

- *tone:* Lets you shape the tonal response of the BODY model. Ranges from dark and filtered, to full-range.

- *pitch:* Lets you shift the relative pitch of the body resonance, up or down one octave. In the boring physical world, the pitch of the body of an instrument doesn't change based on the note being played, but this one sure can. Subtle modulations of this dial can land you in a very special spatial place.
- *sustain:* Lets you set the time for which the model will "ring out" after an input signal is applied.
- *wet* and *dry* dials: Lets you specify the amount of "dry" signal (from the RESONATOR module) and wet signal (from the BODY effect) you wish to pass on to the BODY module.

Not so dry, is it?

BODY Modes

- large wooden box: A guitar-sized wooden enclosure, filled with air.
- small wooden box: A smaller violin-sized wooden enclosure, also filled with air.
- metal plate: A metal plate, supported at the center, with its edges free to vibrate. A primitive cymbal.
- frame drum: A membrane, supported by a thin frame.

OUTPUT

The Output module 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 Kaivo's lovable one-knob Tilt EQ, a limiter (to keep wild signals in check), and a nifty little oscilloscope that shows you the waveforms you're making.

Controls

- *tilt:* This dial lets you change the overall tonal balance of the output signal. Move it clockwise, and you'll get increased treble and a cut in bass. Move it counterclockwise, and you'll do the opposite. This simple EQ is often all you need to get that final balance just right.

- *chorus:* Lets you enable a quite delightful chorus effect, in two variations (or none at all).

- *limit:* Toggles the inbuilt limiter on and off, capping the output just below clipping. Kaivo 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. If you've got a limiter you love the sound of, feel free to use it in lieu of the stock limiter—or avoid limiting altogether by judiciously setting the output level to avoid clipping.

4 Patching for Fun and Profit

So far, we've covered what this instrument is about, how to operate its controls and settings, and the functions of its modules. This chapter seeks to bring all that knowledge together, in the form of a few patching exercises that show you a few novel ways to make sound with Kaivo. Is this the final word on patching? Not in the least, but by the end we hope you feel well-prepared to continue exploring on your own.

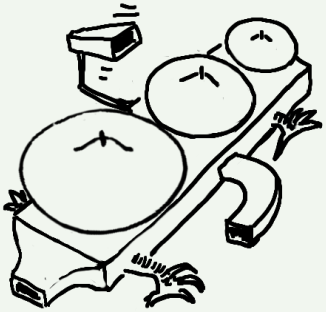
This chapter assumes you have a basic grasp of how to patch modules to one another and manipulate Kaivo's controls. For foundational information such as this, please see Chapter 2, *Getting to Know Kaivo*.

For Whom the Dog Barks

Before we get into heavy granular territory, it'll be good to know how to use GRANULATOR to play samples in a traditional sampler-like way. Let's string together a very simple patch:

1. Load the patch "blank" (from the "Kaivo techniques" category).

2. Load the sound "arf!" (from the "Vocal" category) into GRANULATOR. Move the X-offset slider all the way to the left, at the start of the sample, so that playback begins there when a note is received.



The following patch also comes as a preset called "arf!" which can be found in the "Kaivo Keys" category.

3. Patch the KEY module's *pitch* output to GRANULATOR's exponential *pitch* input (the leftmost of the pair). This ensures that as note data is received, the sample is played at the proper pitch.
4. Patch the KEY module's *gate* output to GRANULATOR's *trig* input. This triggers the creation of a new grain (and thus, playback of the loaded sample) when a note is received.
5. Finally, patch KEY's *gate* output to the GATE module's *level* input. This simply opens the GATE while a key is pressed, allowing sound to pass through.
6. Set the *rate* dial to 0. This ensures that sample playback starts only when a note is received, rather than being re-triggered continuously as you may want when working with smaller grains.

This is the simplest way to open and close the GATE with note input. In many cases, you'll get better control by using the *gate* output to trigger one of the Envelope modules, and using the Env's output to control the GATE module.

Now, send Kaivo some MIDI notes, and you've got one fine barking machine.

Pitch and Rate in Tandem

Now, what if you wanted to use this bark as the source for a continuous tone to play with? If you turn *rate* up to 55 or higher, suddenly you've got an oscillator-like pitched tone to play with, but you'll notice that as you play notes, the timbre shifts, but the pitch of the tone stays steady.

This is because at higher *rate* settings, *rate* governs the pitch that you hear. So far, you've got pitch data patched into the *pitch* dial, but not into *rate*, so the apparent pitch of the sound remains unchanged. Patch the *pitch* output into the exponential *rate* input as well, and you've got a granular sound that remains aligned in pitch and timbre across a broad range of notes.

Waves of Uncertainty

Kaivo seems to love sounds from nature—there's just an endless array of timbres and dynamics to work with. Playing through such samples at low *rate* settings lets you take full advantage of their ever-shifting nature, but sometimes a little something extra is required to keep things sounding "natural." Let's open the patch "mechanical waves" (from the "Kaivo techniques" category) and take a look at what's going on.

Looking at the patch, we see that it comes with the "creek" sample loaded into GRANULATOR. We're getting a nice multi-layered sound, because *overlap* is set to 8, meaning there are 8 simultaneous grains flowing through the sample at a time.

For example, suddenly it's no longer just a recording of wind through some trees—it's multiple winds and multiple stands of trees, each behaving differently.

You can tell there's been a little effort applied to creating changes over time—LFO 2D has been called in to modulate the X/Y offsets for GRANULATOR, and to create wave-like volume swells with GATE. NOISE is patched to GRANULATOR's *pan* and *pitch* dials, scattering new grains across the stereo field, each at a slightly differing pitch.

Still, regardless of the organic source material and all this modulation, there's still a mechanical, chugging aspect to the sound you hear. That's because grains are being created at a regular interval, set by *rate*. If only there were an easy way to make that interval less predictable...

You could route NOISE or another modulator to the *rate* input, but we've included a noise source right inside of GRANULATOR for just this sort of occasion. Flip on the *noise* button (right next to the *rate* dial) to hear the beautiful uncertainty take hold.

This leaves the modulator modules free to be configured for other tasks.

With *noise* engaged, the overall rate of grain creation stays similar to that set by *rate*, but the interval between grains varies, just enough to keep things natural.

Three Envelopes

In a traditional subtractive synthesizer, it's often easy to think of the tone-generating elements (such as oscillators and sample players) and the signal processing elements (VCAs, filters, effects) as two distinct camps. One side creates the core of the sound, and the other carves it into the desired shape.

Kaivo strays from that model a bit. While the RESONATOR and BODY modules are technically signal processors, they often act like additional tone generators, spurred on by GRANULATOR's signals. To that end, each of them has a *sustain* dial, offering a range of decay rates for the model, from short blips to near-drones.

Perhaps it goes without saying that each *sustain* dial has a corresponding signal input.

So if GATE is applying volume shapes to GRANULATOR's output, and both RESONATOR and BODY have their own personal amplitude-shapers, we're dealing with three separate-yet-related volume envelopes. If that sounds intriguing, load up the preset "so many envelopes" from the "Kaivo techniques" category.

Whoa nelly! There's quite a few patch cables strewn around in there. Let's take a look at what's going on:

The SEQUENCER is providing a melody (which is patched into *pitch* and *rate* for GRANULATOR, as well is the exponential *pitch* input for RESONATOR) and a trigger pattern, which is triggering ENVELOPE 1. The envelope is controlling GATE, shaping the dynamics of the signal from GRANULATOR.

This patch derives its tempo from that of your host. If the sequence isn't playing at first, press Play on your DAW.

If you look at ENVELOPE 1, you'll see that the Y output from LFO 2D is slowly sweeping the envelope's *release* back and forth, subtly shaping the lengths of each note over time. Snappy, aren't they?

Calling in Reinforcements

Now, kindly turn RESONATOR's *wet* dial up until you hear the resonant model working. You'll notice that the pitch of the model stays fairly constant, besides a little texture-boosting frequency modulation coming into its linear pitch input from NOISE. Now, try double-clicking on RESONATOR's exponential pitch input dial. This will set the input to its default setting, and you'll soon hear RESONATOR following along in pitch with GRANULATOR.

The linear pitch input responds smoothly to incoming signals, so yes, you can do linear FM with RESONATOR!

RESONATOR's *sustain* dial is being swept back and forth by the X signal from LFO 2D, changing the decay of the model over time. The X and Y signals are 90 degrees out-of-phase but otherwise identical in shape, so we wind up with two modulators on the same "path," yet always at different points upon that path. This makes the sounds from GRANULATOR and RESONATOR lengthen and shorten at identical rates, simply out-of-step with one another. So, though the notes coming from GRANULATOR are very short, the decay of the RESONATOR model has ideas of its own.

Although technically, I suppose those "ideas" are really those of LFO 2D.

Finally, try double-clicking on BODY's *wet* dial to bring in the body model. The sustain for the body model is also being modulated by the X output from LFO 2D, yet if you look closely at both *sustain* dials, you'll see that as one is pushed upward, the other is pushed downward. This is because we've set BODY's *sustain* input dial at a negative value, and RESONATOR's at a positive one. As the X control signal enters both inputs, they move in opposite directions in relation to the incoming signal.

So, this just goes to show that while there is one main audio signal path in Kaivo, there are many sources for sound (additive or shaping others through FM/AM) , and each can be controlled separately.

Cloud of Crows

This final example is a brief one; a demonstration of one method for creating granular "clouds" with Kaivo. Open patch "cloud of crows" from the "Kaivo techniques" category, and prepare for a little Hitchcock-ian thrill. The source audio is a 4-channel sample of crows cawing. If you were to listen to that sample as a loop, it might seem chaotic at first, but a pattern would soon emerge—it's just eight seconds of audio, after all.

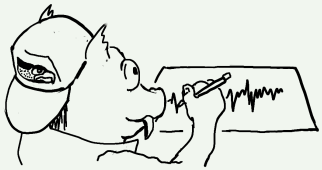
Not able to leave well enough alone, we've gone and patched an unsteady signal from NOISE *all over the place*—it's setting the *pitch*, *rate*, X/Y offset, and *pan* of each new grain, and even adding a bit of queasiness to the BODY module, just for fun. This means that our eight second sound is now endlessly varied, with no discernible pattern.

We're getting continuous sound with no notes held down because we've left the GATE fully open. Having *wrap* enabled doesn't hurt, either. In the case of long, multi-channel source audio like this, adding some X/Y offset motion lets us listen longer without sensing too much repetition. There are many valid views of this scene, and we're going to hear them all.

LFO 2D is set to languorously shift the center value (*offset*) and output amplitude (*level*) of NOISE around, subtly modulating everything that NOISE touches over time, for another layer of change.

There's not a lot to this patch, but we recommend you try disconnecting each of the cables coming from NOISE, one after the other, to hear what changes. Disconnect the cable from GRANULATOR *pitch*, and the crows suddenly seem more consistent in size. Disconnect it from GRANULATOR *pan* and away goes the sense of space.

Also, try adjusting the input level dial for each parameter NOISE is affecting. Negative, positive, lots or a little, it's a good thing to get in touch with how modules react to differing levels of signal input. You never know what you might find.



Putting it All Together

We hope this book has given you a good picture of the resources available in Kaivo and some good inspiration to get going with them. We could devote many more pages to detailed recipes for patch after patch, 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 "Kaivo?"

It's the Finnish word for "well."

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

Kaivo is a plug-in, in VST and Audio Units formats. 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 Kaivo supposed to sound like this?

Probably, unless you hear an abrasive series of glitches. Here's a good way to check that Kaivo is functioning well: Select the "default" patch from the factory 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 bit. You should hear a mellow, slowly shifting drone. If there are 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 Kaivo 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 Kaivo's voices. You can try turning the ***voices*** control down to 1, and turning up the audio again on the default patch. If this helps, then it's almost certainly the case that CPU power is the issue. You can try adding voices one by one to hear where the problems come up.

If you are running the 64-bit version of Kaivo 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 Kaivo 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 Kaivo users on this topic, visit the ongoing discussions at madronalabs.com.

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

Yes. Kaivo'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 Kaivo 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?

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

Yes. Kaivo 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 Kaivo, you will get errors.

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

Kaivo 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 Kaivo eating my CPU time?

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

Kaivo 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.

How do I make Kaivo'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.

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

B Index