GrainWave 3.0.0 Manual

Contents

GrainWave 3 Manual

Introduction

GrainWave 3 is a real-time, software synthesizer for Power Macintosh (PPC) computers. GrainWave will run on the standard equipment that comes with any PPC. It does not require any additional hardware.

GrainWave has an open synthesis architecture, based upon units called **operators**. An operator can generate a signal or transform one. Operators are collected into **regions** which represent individual instruments. One or more regions are contained in a **patch**. Only one patch at a time is available to be played. GrainWave may be played by using the mouse, the keyboard, and/or MIDI.

GrainWave was designed to be flexible, with high-quality audio output, while still maintaining real-time performance. GrainWave 3 has an output latency of 11.6 ms on any Mac running OS 8.1 or higher. GrainWave can also manipulate an audio stream coming into your computer, with a throughput latency of 23 ms. GrainWave has sophisticated MIDI timing system to ensure a less than 1 ms. of jitter with incoming MIDI events.

GrainWave is completely backgroundable. It can run below other applications while continuing to output an uninterrupted stream of sound. Furthermore, it can accept MIDI information from other, OMS-capable applications running on the same computer. A sequencer or control application can use GrainWave as a virtual synthesizer without a significant impact on its own performance. Because of this capability, GrainWave does not have a sophisticated interface to control its playback from the screen. You can easily design your own interface using Max, HMSL, or a sequencing program.

GrainWave contains a wide variety of operators with which you can create synthesis or processing algorithms. Each operator completely encapsulates its own output. Any operator can be used as the input to another operator.

GrainWave uses a graphical design interface. Each operator is positioned in a region window and connected to other operators. A network of operators makes up a region, which is similar to an instrument or voice in a hardware synthesizer.

GrainWave makes extensive use of function tables and soundfiles. A graphical and mathematical editor is provided for use in editing table and soundfiles.

Differences from GrainWave 2.x

GrainWave 3 has a completely redesigned synthesis engine, which is considerably more flexible than the GW2 engine. The user interface for patch design has been overhauled. The engine now allows region design while the patch is playing. Operators can be linked into feedback paths. For a more complete description of the changes, please read “Upgrading from GrainWave 2.x.”

Registration

GrainWave 3 is shareware. If you use it, you MUST register it. The registration fee is $40 US. Once you register, you will receive a code which will unlock the record-to-disk features. If you are a registered user of GrainWave 2.x, your current registration code is still valid. If your situation precludes the payment of $40, I am happy to accept musical works or other items in place of the registration fee. To register, send your name, email, and $40 US to:

Michael Berry
12401 Princess Jeanne
Albuquerque, NM, 87112 USA

or follow the instructions in the "Register GrainWave" app in the main GrainWave download. "Register GrainWave" will allow you to pay by credit card or foreign check through Kagi (www.kagi.com).

You can also reach me by email at mikeb@nmol.com. There is a GrainWave email list for discussions between users. To subscribe, send email to listproc@eartha.mills.edu with the following message in the body:

subscribe grainwave <Your Name>

Patch Reference

General

The patch is the basic document of GrainWave. Only one patch can be open at a time. A patch contains one or more regions which contain sound-generating algorithms. A patch also contains all of the FTables and soundfiles which are needed by the regions.

A patch can either be playing or not. When you first open a patch, it is not playing. You can use the menu option **Play** (⌘Y) to toggle between playing and not playing. When the patch is playing, the “Playing” LED in the **Control** palette will light green. There are several actions which cannot be performed while a patch is playing. They include: editing a Function Table or Sound File, and changing the length of a Delay Line.

GrainWave is a completely backgroundable application while it is in play mode. The generation of sound will automatically take a higher priority than most other operations on your Mac. Most significantly, this allows you to generate control data in another application (like Max or Vision) running on the same computer and use the data to control GrainWave. (see **MIDI Reference**)

Patches may be saved as files and opened at a later time. These files contain all of the relevant information for the patch except for any soundfiles used in the patch. An alias to these files is saved instead. If you move or rename a soundfile, GrainWave will prompt you to find it when you open a patch using that soundfile.

Patches may also be saved as "Combined Packages". A combined package includes the patch as well as all of the needed soundfiles in a single file. This is the best method for transferring patches which contain sound files from one computer to another.

Master Volume

Each patch has two master volume controls. The knob in the **Controls** palette is the overall master volume. Simply move the knob up or down to adjust the output levels.

The second master volume is only available via MIDI. The master volume control defaults to MIDI controller 7 on channel 1. You may set this to any controller on any channel. The master volume control can be accessed by pressing the volume knob button in the patch window. The MIDI master volume will affect the volume downwards **from the knob setting**. So if you are getting no output, check both the knob and the MIDI controller.

Combined Packages

A combined package is a special file format which can be used to easily move and exchange patches which use soundfiles. When you save a standard patch, the soundfiles are separate from the patch itself. If you move the patch to a different machine or disk, the patch will ask you to find each of the necessary soundfiles, which can be time-consuming and confusing.

The combined package avoids this problem. A combined package file includes a copy of each of the necessary soundfiles in a single file with the patch. When a combined package is opened, it unbundles the soundfiles in the new location. This is similar to Unstuffing a stuffed file with StuffIt. After you open a combined package, a new copy of each soundfile and the standard patch will appear in the folder where the combined package was.

Combined packages should only be used for moving GrainWave packages. Each time a combined patch is opened, it will create a new copy of the patch and the soundfiles. If a patch does

not include any soundfiles, there is no advantage to saving it as a combined package.

Region Reference

General

The region is the unit of GrainWave that roughly corresponds to a voice on a synthesizer. Each region contains a set of operators which generate a signal of some kind. A region may have polyphony–multiple copies of itself that allow you to play "notes." Regions have a triggering and de-triggering mechanism. And regions can be enabled and disabled using MIDI program change messages.

Each region has a box in the patch window, which displays the region name and allows access to the region. You can open a region by double-clicking on its window. Once opened, a region has a tab panel with different panes for different parts of the region. Several of the panes may be split out into separate windows for ease of use.

The position of the region in the patch window is important, even if you are not using the mouse to control the region. GrainWave will process the regions in order from the top to the bottom, and then the left to the right. If one region depends on the output of another, it should be placed below the one it is dependent upon.

Region Window

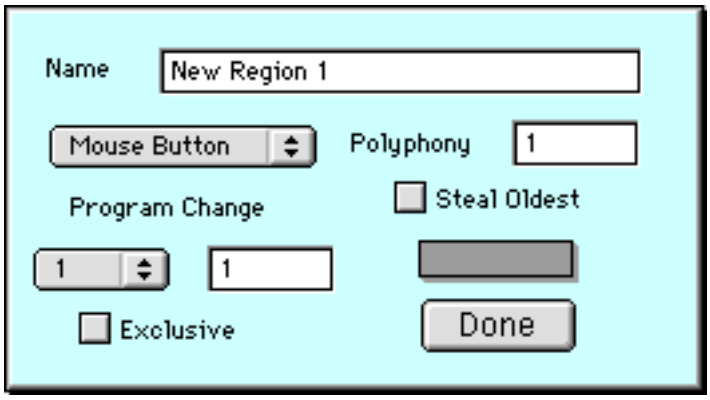
The region window is where instrument design occurs. The main work area contains operators and connections. To add an operator, simply drag the desired type from the **Operator** palette to the region window.

When you click on an operator, it will become selected and appear in yellow. Additionally, depending on the operator, setting controls will appear in the bottom of the region window. These are

operator-specific and will be discussed in more detail in **Operator Reference**.

Region Info

If you choose **Region Info** (⌘I) while you have a region window open, a dialog box will appear which allows you to set the overall characteristics of the region.



Name is the name of the region. Below the name is the **Trigger Type** menu. You can choose from the following options.

Mouse Button: The region will be triggered by a mouse click in the region box in the patch window.
MIDI Trigger: The region will be triggered by a MIDI note-on that is recognized by an operator in the region.
Always On: This is a special kind of region. As soon as you enter play mode, a note will be triggered that will not stop until you exit play mode. Furthermore, the polyphony in an **Always On** region is limited to one note. This is primarily designed for global effects which are not triggered by notes.
Key Trigger: The region will be triggered by computer keyboard keys that are recognized by an operator in the region.

Each region can only be triggered by one method at a time. However, certain operators will allow a region that has been triggered by one method to respond to other input methods as control data.

Polyphony allows you to set the maximum number of notes available simultaneously from a region. GrainWave needs to pre-allocate memory for all of the notes at the beginning. The maximum effective polyphony will depend on the amount of RAM in your computer and the speed of your processor. If **Steal Oldest** is checked, a new note will stop the oldest note playing and use its memory space if there are no free notes available. Otherwise, no new notes will start until one of the existing notes ends.

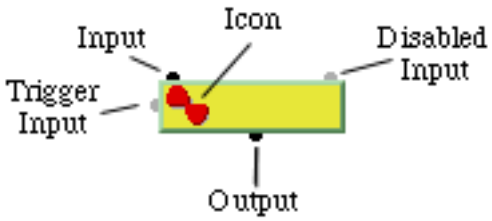
Program Change sets how the region responds to MIDI program changes. The popup menu sets the MIDI channel and the edit field sets the program number. When **Exclusive** is checked, the region responds like a synthesizer. It will turn off all of the other regions and then turn itself on. This process will terminate any existing notes in the region. If the region is non-exclusive, a program change will only effect the one region. If the region is presently enabled, it will be disabled, and vice versa. In this mode, the program change acts more like an on/off switch for the region.

Operator Reference

General

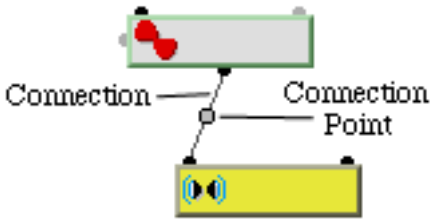
An operator is a unit which performs a particular kind of calculation. Each operator is represented by an **element** in the region window. To add an operator, simply drag the desired icon from the **Operator Palette** to the region window. An element will appear where you drop the icon.

This is a sample operator element:



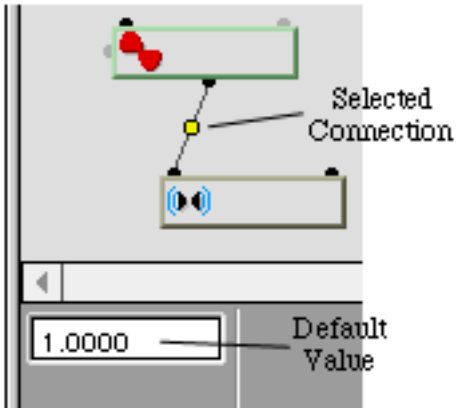
The icon shows what kind of operator this is (in this case, a **Wave** operator). A Wave operator has two Inputs, it is triggerable, and has one output. Each of the small buttons is called a point. Some points may be disabled at some times. For instance, the second input for a wave operator is the number of the function table. This input will only be active when “Use Source List” is chosen in the source menu (see **Switchable Sources** in this chapter for further information). Disabled inputs and triggers will be gray instead of black. You can still make a connection to a disabled input, but as long as it is disabled, the connection will have no effect.

Any output point can be connected to any input or trigger point. To do this, click and hold on the output point that you want to connect. Then drag to the input or trigger point you want to connect to. When you are over a valid point, it will get larger. Simply drop the drag on the enlarged point to make a connection. Here are two connected operator elements:



A signal will flow from the output of the upper operator into the first input of the lower operator. Each connection has a connection point. You can control the volume of the connection using the connection point by hooking an operator that makes a volume output to the point.

When you move the mouse over any point, a short description of the point's function will appear at the bottom of the region window. For all input and connection points, you can also set a default value. When you click on an input or connection point, it will turn yellow and a number field will appear in the lower portion of the region window. This number is the default value. You can change the number to whatever you like. The default value will be used by the operator or connection whenever there is no connection hooked to that point. Once one or more connections are made to a point, the default value is ignored. Here is an example of setting the default value for a connection:



If you were to change the default value here to 2, the output of the Wave operator would be multiplied by 2 before it was delivered to the next operator.

Audio signals are considered to vary from 1.0 to -1.0. If all of the Mono and Stereo Outs add up to more than 1.0, clipping will occur on the output (unless you decrease the Master Volume). You may use any values you wish inside of a region (except with certain operators like Waveshaper and Distortion), as long as they are appropriately scaled before the output. All of the calculations use floating point values and can be scaled without signal degradation. GrainWave does not take into account polyphony automatically. A region with a single note may not clip, but five simultaneous notes might.

Resolution

Each operator has a resolution. This sets how many output values are created. A resolution of audio rate (1) creates 1 output value every audio sample. A resolution of 4 samples creates 1 output value every 4 audio samples. A resolution of buffer rate creates 1 output value per audio buffer, or 1 every 256 audio samples. By making the resolution larger, you can shorten the time it takes to calculate an operator. Smaller resolutions, however, are necessary for audio quality signals. Some operators do not have settable resolutions. In most cases, this means that only one sample per buffer could possibly be created (the input readers function this way). The Mono and Stereo Out operators always runs at audio rate.

Some operators also have a special resolution "Note Rate" available. Note Rate operators are evaluated only once at the beginning of any note. After this, they are bypassed, though their output value is available for any operator. An example is the Random operator. When set to Note Rate, the Random operator will generate a single random number at the beginning of the note. This number is fixed for the length of the note. When a new note is triggered, a new random number will be generated.

Triggering Operators

Some operators are triggerable operators. These may or may not be active in a note, depending upon circumstances. Triggerable operators can be identified by the trigger point on the left side of the operator element.

There are four different trigger modes available for triggerable operators:

Always On: Any time the region is triggered, this operator will be triggered too. This is the default value.

MIDI Trigger: An operator can be triggered by any combination of MIDI note-ons on any channels. Simply choose the channel you want and highlight the appropriate keys on the piano keyboard display. Any note that is lit red will trigger this operator.
You can also set the trigger notes using a MIDI keyboard. Press "Set by MIDI In." While it is depressed, any MIDI notes played will be added to the piano keyboard display.

Trig Input: This activates the trigger input. Now this operator will only be triggered when it receives a non-zero value at its trigger point.

Keyboard: The operator will be triggered by pressing the computer keyboard. Simply enter (without spaces) the keys which you wish to use to trigger the operator.

Naming Operators

You can name each operator if you wish, though it is not required and does not affect processing in any way. To change the name, hold the control key and click on the white box in the operator element. Then type in the name.

Switchable Sources

All of the operators which use function tables or sound files can have their sources switched in real-time. In each source menu, if you

choose the item "Use Source List", a operator menu will replace the open table button. This input accepts a number corresponding to the position of the desired function table in the list. So a 3 would set the third function table or sound file for that buffer. This can only be switch at buffer rate, no matter what the resolution of the operator or input.

This is most useful for multi-sample setups, where different MIDI notes trigger different samples. In the past, a separate Sample operator was required for each sample, even if only one would ever be triggered at a time. Now, one Sample operator can make all of the sound files available. If the table is switched by the MIDI note number, then an efficient and simple multi-sample region can be designed.

Operator Definitions

Oscillators

Oscillators generate a periodic waveform from a function table. All oscillators are triggerable.

Wave

The Wave operator creates an output waveform that matches the chosen FTable.

Input 1: The frequency of the output wave in Hz.
Input 2: The number of the Ftable if “Use Source List” is selected.

Options: The interpolation type for the wave is selectable. “Truncating” is the fastest and lowest quality mode. “Linear” is slower but better suited for audio rate signals.

Oscillator

Oscillator creates a specific waveform chosen in the options. Oscillator automatically prevents aliasing and is therefore better suited for creating sharp-edged audio waveforms, like square, sawtooth, and triangle waves.

Input 1: The frequency of the output waveform in Hz.

Options: The output waveform shape; sine, square, sawtooth, or triangle.

Pulse

Pulse is identical to Wave except that the duty cycle of the wave can be varied. The duty cycle is the ratio of positive cycle to negative cycle. A duty cycle of 0.75 would make a waveform that spent 75% of its time in the positive range and 25% in the negative range. Pulse can be used to create “Pulse Width Modulation” effects.

Pulse does not take a complete cycle of a waveform like Wave. Instead, it takes a Ftable of only the positive portion of the desired waveform and duplicates it for the negative portion. There is no automatic aliasing protection with pulse, so care should be taken when designing the waveform to be used.

Input 1: The frequency of the output waveform in Hz.
Input 2: The duty cycle of the waveform, from 0 to 1.
Input 3: The number of the FTable if “Use Source List” is selected.

Feedback FM

Feedback FM generates a frequency-modulated signal where a fraction of the output is fed back into the frequency input.

Input 1: The frequency of the output waveform in Hz.
Input 2: The multiplier of the output that is fed back into Input 1.
Input 3: The number of the FTable if “Use Source List” is selected.

Options: The feedback can be calculated in one of two ways. “Raw Feedback” simply multiplies the output by the feedback value and adds it to the frequency input. “Scaled Feedback” adjusts the feedback to the input frequency value. If the input frequency is 200, the output will be multiplied by 200 before being multiplied by the feedback value.

Samplers

Samplers are sample playback operators which use externally generated audio files. All audio files for use with GrainWave must be mono AIFF, 16 bit, and 44.1 kHz.

Sample Player

Sample player plays a sound file. The sound file can be played at any speed, including backwards, and can be looped. There are two separate loops; the sustain loop and the release loop. In most

cases, only the sustain loop will be used. The release loop will only be activated if an Envelope operator switches the region into a release (see Envelope below).

Input 1: The frequency of playback. All frequencies are related to a base frequency of 200 Hz. So a frequency of 400 Hz. Will result in double speed playback, 100 Hz. In half speed playback.
Input 2: The starting position in the sound file.
Input 3: The starting loop point for the sustain loop.
Input 4: The ending loop point for the sustain loop.
Input 5: The starting loop point for the release loop.
Input 6: The ending loop point for the release loop.
Input 7: The number of the sound file if “Use Source List” is selected.

Options: Just like Wave, Sample player has adjustable interpolation. Sample positions can be defined in two ways. “Raw index” reads the sample position as an actual sample number. “Zero to One” reads the sample position as a fraction of the whole file.

A sample operator can control triggering of the entire region. “Stop Note” means that when the sample finishes playing, the region will be de-triggered. “Don’t Stop Note” means that the region will continue to play after the end of the sample, at which point the sample player will output zeroes.

Both the sustain and release loops have three modes. “No looping” means that loop points are ignored. “Fixed Loop” reads the loop points from the sound file. You can set these loop points using the **Ftable Editor**. “Free Looping” enables the start and endpoint inputs and reads the loop points dynamically.

Grain

Grain is a granular synthesis generator. It is designed for synchronous granular synthesis, where a new grain is created at an interval set by the stream frequency. However, if the stream frequency is randomized, you can create cloud effects.

Grain uses two sources. The first is the audio data for the grain. This can be either an Ftable or a sound file. The second is the window that is applied to the grain. If no Ftable is chosen as the window, no window will be applied.

Input 1: The frequency of the stream of grains in Hz.
Input 2: The starting position of the grain in the sound file. This is only active for sound file grains.
Input 3: The length of the grain. This is only active for sound file grains. For Ftables, the length is the length of the whole Ftable.
Input 4: The playback frequency of the grain itself, relative to 200 Hz. (see **Sample Player** for more info).
Input 5: The number of the grain Ftable or sound file if "Use Source List" is chosen.
Input 6: The number of the window Ftable if "Use Source List" is chosen.

Options: For sound files, the start position and length of grains can be read either as "Raw Index" or "Zero to One" (see **Sample Player**). Grain normally does not adjust its output volume depending on the number of overlapping grains ("Auto Gain Off"). "Auto Gain On" means that as more grains overlap, Grain will decrease the output volume automatically.

Control I/O

Control I/O operators read control data from sources external to GrainWave. All Control I/O operators always have a resolution of Buffer Rate.

MIDI Reader

MIDI Reader allows for MIDI data to be interpreted by GrainWave. All channellized MIDI data can be read by MIDI Reader. The data can then be scaled using an optional Ftable.

Input 1: The number of the scaling Ftable if "Use Source List" is chosen.

Options: MIDI Reader can read different kinds of MIDI data. “Note Number” and “Velocity” are for the current note. They will only be active if the region is MIDI Triggered. “Pitch Bend” and “Aftertouch” read the values from the designated MIDI channel. “Controller”, “14 bit Controller”, and “Poly Aftertouch” read the value of the designated number on the designated channel.

Mouse Reader

Mouse Reader reads one of the mouse coordinates. 0,0 is the upper left corner of the region box in the patch window. Positive x coordinates move left to right and positive y coordinated move top to bottom.

Input 1: The number of the scaling Ftable if “Use Source List” is chosen.

Options: Mouse Reader can be switched to read the x or y coordinate.

Key Reader

Key Reader reads whether a certain key is pressed or not. If pressed it outputs a 1; otherwise 0. On all Macs before the iMac, only two keys can be pressed at one time and read by the system (other than shift, option, command, and control). On iMacs, this has been changed to 6 keys.

Audio I/O

Audio I/O operators read or write audio to/from the audio jacks on your Mac.

Audio Input

Audio Input reads the stereo audio input from the input jacks on your Mac. It has two outputs–left and right. Audio Input is a triggerable operator.

Mono Output

Mono Output sends the same signal to both the left and right audio output jacks on your Mac.

Input 1: The audio signal.
Input 2: The volume of the audio signal.

Stereo Output

Stereo Output sends different signals to the left and right audio output jacks on your Mac.

Input 1: The audio signal to send to the left output.
Input 2: The audio signal to send to the right output.
Input 3: The volume of both channels.

Mathematical Operations

Mathematical Operations are a variety of basic math functions which are available in GrainWave.

Random

Random outputs random numbers at a variable interval. The numbers are evenly distributed between the low value and low value + range. The generator has a very long period and is suitable for white noise generation. Random is a triggerable operator.

Input 1: The lowest possible number generated.
Input 2: The range between the lowest possible number and the highest possible number.
Input 3: The frequency at which new numbers are generated. The default (22050) creates full bandwidth audio white noise when the resolution is Audio Rate.

Constant

Constant is single floating point constant. It always has a resolution of Buffer Rate.

Mathematical

Mathematical performs a selectable math function on either a single input or two inputs, depending upon the operation.

Input 1: The “a” input.
Input 2: The “b” input.

Options: Mathematical can perform the following operations:

Operation	Output =
+	a + b
-	a - b
*	a * b
/	a / b
round	a to the nearest integer
floor	a rounded down
ceil	a rounded up
upper bound	if a > b, b otherwise a
lower bound	if a < b, b otherwise a
sin	sin(a) (a in radians)
cos	cos(a)
tan	tan(a)
log	ln(a)
exp	e^a
mod	remainder of a / b
pow	a^b
sqrt	a^{-2}

Comparator

Comparator evaluates a relational statement between the two inputs. If the statement is true, it outputs a 1; otherwise 0.

Input 1: The “a” input.
Input 2: The “b” input.

Options: Comparator can perform the following relations:

Relation	Output
==	does a = b
!=	does a not = b
>	is a greater than b
>=	is a greater than or equal to b
<	is a less than b
<=	is a less than or equal to b

Equation

Equation evaluates a arithmetic equation on up to four inputs plus constants. The equation can use +,-, *, and /, as well as parentheses. The equation can only be changed while GrainWave is not playing.

Input 1: Input “a”.
Input 2: Input “b”.
Input 3: Input “c”.
Input 4: Input “d”.

Options: The equation is entered in the edit field. Example equations are: a + b (a + b) * c ((a + b) * (c + d)) + a. Use only lower case letters.

Accumulator

Accumulator performs an operation on an input and keeps a running total. Accumulator also has a special function which will output the number of samples elapsed since the beginning of the note.

Input 1: Input “a”.

Options: The following operations can be performed.

Operation	Output	Start Value Used
Add	total = total + a	yes
Subtract	total = total - a	yes

Multiply	total = total * a	yes
Divide	total = total / a	yes
Average	(total + a) / number	no
Maximum	if a > current, a else current	no
Minimum	if a < current, a else current	no
Elapsed	number of samples	no

Interpolator

Interpolator performs linear interpolation on a higher resolution to create a smooth lower resolution signal. For example, Interpolator set to 16 Sample resolution could smooth the transitions between values for a Mouse or MIDI Reader operator at Buffer Rate resolution.

Input 1: The signal to be interpolated.

Envelopes

Envelopes create a shape that varies over the life of a note.

Envelope

Envelope creates a shape that has an attack, a decay, and a release. After the decay, Envelope enters a steady state sustain which persists until the note is ended, at which time it enters the release. Each segment is read from an Ftable. If any segment does not have an Ftable assigned to it, the segment is skipped. Each segment also has a time input which determines the length of the segment.

Envelope creates its shape by multiplying the input signal by the current value in the segment's Ftable. During the sustain segment, the input is multiplied by 1. For this reason, the end of the decay segment (or attack segment if there is no decay segment) should always be 1, to avoid a click.

Though it is possible to pass an audio signal directly through an Envelope, this is usually not the most efficient methods of using an

envelope. Because it is a computationally intensive operator, it is better to run Envelope at 16 Sample or 64 Sample resolution. Simply do not connect anything to the input and connect the output to the connection points at the outputs of the Audio Rate operators that you wish to control with the Envelope.

Envelope is a triggerable operator.

Input 1: The input signal to be multiplied by the current segment value.
Input 2: The length of the attack segment in milliseconds.
Input 3: The length of the decay segment in milliseconds.
Input 4: The length of the release segment in milliseconds.
Input 5: The number of the attack Ftable if “Use Source List” is chosen.
Input 6: The number of the decay Ftable if “Use Source List” is chosen.
Input 7: The number of the release Ftable if “Use Source List” is chosen.

Options: Envelopes can have a sustain segment (“Sustain”) or not (“No Sustain”). If not, then as soon as the decay segment is done, the envelope enters the release segment..

An Envelope can control the triggering of its region. If “Stop Note” is selected, then the Envelope will end the note at the end of the release segment. Furthermore, if “No Sustain” the Envelope will de-trigger the note at the end of the decay segment and enter the release segment. If “Don’t Stop Note”, the Envelope will have no effect on the region as a whole. It will only enter the release segment if some other Envelope is controlling the region.

Signal Flow

Signal Flow operators control the routing of a signal path.

Switch

Switch outputs one of two inputs, based on the value of a third input.

Input 1: This input passes to the output if Input 3 is non-zero.
Input 2: This input passes to the output if Input 3 is zero.
Input 3: This input switches the output between Input 1 and Input 2.

Blank

Blank does not affect the signal in any way. It can be useful for routing feedback loops.

Input 1: This input is passed through to the output.

Ftable Operators

Ftable Operators perform specific operations involving function tables.

Ftable Reader

Ftable Reader uses its input value as an index to read a value from an Ftable.

Input 1: The index of the value in the Ftable to be outputted.
Input 2: The number of the Ftable if “Use Source List” is chosen.

Ftable Writer

Ftable Writer allows you to change the values in an Ftable during playing. You can use Ftable Writer to store control numbers, or, in conjunction with Ftable Player, as a real-time sampler. Ftable Writer also allows for dynamic alteration of waveforms and scaling tables.

Input 1: The input signal.
Input 2: The index to write the input or the trigger signal.
Input 3: The number of the Ftable if “Use Source List” is chosen.

Options: Ftable Writer has three operational modes. “Write To Index” simply writes the input signal to whatever index is specified by Input 2. In this mode, the Ftable Writer is always active. A second input with the same index will over-write the first.

“Sequential, No Loop” writes the input values sequentially starting at the beginning of the table. It will start writing as soon as it receives a non-zero value at Input 2. It will continue writing as long as Input 2 is non-zero. Once it fills the table, it will stop writing. If it is retriggered with a non-zero value at Input 2, it will begin again at the start of the table.

“Sequential, Looping” operates just like “Sequential, No Loop”, except that it will not stop writing at the end of the table. When it reaches, the end, it will start again at the beginning.

FTable Player

Ftable Player is identical to Sample Player, except that is uses an Ftable as its source instead of a sound file. While it was designed for use as a real-time sampler with Ftable Writer, it can also function alone with Ftables designed to be played once.

Input 1: The frequency of playback. All frequencies are related to a base frequency of 200 Hz. So a frequency of 400 Hz. will result in double speed playback, 100 Hz. in half speed playback.
Input 2: The starting position in the sound file.
Input 3: The starting loop point for the sustain loop.
Input 4: The ending loop point for the sustain loop.
Input 5: The starting loop point for the release loop.
Input 6: The ending loop point for the release loop.
Input 7: The number of the sound file if “Use Source List” is selected.

Options: See **SamplePlayer**.

Inter-Region Operators

Inter-Region Operators allow for the transfer of signals between different regions.

Global Outlet

Global Outlet sends makes its input signal available to Global Inlets in other regions.

Input 1: The signal to send to other regions.

Options: You need to set a unique ID number for each Global Outlet. This ID number is used by Global Inlets to identify which Global Outlet to hook to.

Global Inlet

Global Inlets receive signals from Global Outlets. The Global Inlet needs to be processed after the Global Outlet, so the region containing the Global Inlet must be placed lower in the patch window from the region containing the Global Outlet. Global Inlets will always have the same resolution as the Global Outlet to which they are matched.

Options: Each Global Inlet needs an ID number that matches a Global Outlet. If there is no matching ID, the Global Inlet will output zeroes. If two Global Outlets have the same ID, the Global Inlet will use the first one it finds.

Filters

Filters perform digital filtering on input signals.

Filter

Filter is a multi-mode digital filter that can do lowpass, highpass, bandpass, and bandreject filters, in 2-pole (12 dB per octave) and 4-pole (24 dB per octave) configurations.

Input 1: The input signal to be filtered.
Input 2: The frequency of the filter in Hz. For lowpass and highpass filters this is the cutoff frequency. For bandpass and bandreject filters this is the center frequency.
Input 3: The bandwidth for bandpass and bandreject filters.

Options: The mode of the Filter can be selected in the options. For bandpass and bandreject filters, the bandwidth can be in Hz. ("Width as Hz.") or Q ("Width as Q"). For "Width as Q", the bandwidth in Hz. Is calculated by the following formula:

width(Hz.) = center(Hz) / width(Q)

Resonant Lowpass

Resonant Lowpass is a 4-pole lowpass filter with an adjustable resonance. As the resonance is increased, more and more emphasis is placed on the cutoff frequency over all others.

Input 1: The input signal.
Input 2: The cutoff frequency in Hz.
Input 3: The resonance from 0 to 1.

Parametric EQ

Parametric EQ makes a boost or cut to a specific frequency range of the input signal.

Input 1: The input signal.
Input 2: The center frequency of the EQ in Hz.
Input 3: The bandwidth of the boost or cut in octaves.
Input 4: The amount of boost or cut in dB, from -24 to +24.

Allpass Filter

Allpass filter is a filter which causes phase alterations without frequency alterations. It is often used as a component of a reverb system.

Input 1: The input signal.
Input 2: The amount of delay in the filter.

Input 3: The gain of the filter, from 0 - 2.

Effects

Effects perform a process on an input signal.

Delay Line

Delay Line stores it input signal and releases it after a preset amount of time. The size of the Delay Line can only be changed while GrainWave is not playing.

Input 1: The input to be delayed.

Options: Each Delay Line needs a unique ID number if it is going to be used in conjunction with a Delay Tap (see below). The amount of delay is set in milliseconds.

Delay Tap

Delay Tap reads from a specified point in a Delay Line and optionally feeds back into the Delay Line. In order to function, a Delay Tap must be matched with a Delay Line. More than one tap can use a single Delay Line for multi-tap effects. Delay Taps will always have the same resolution as the Delay Line to which they are matched.

Input 1: The length of the tap in milliseconds.
Input 2: The amount of feedback into the Delay Line.

Options: You must specify an ID number that matches an existing Delay Line ID number in order for a Delay Tap to be active.

Flanger

Flanger is a delay line with a delay tap that is modulated by a sine wave.

Input 1: Audio input.
Input 2: The modulation frequency of the flanger in Hz.
Input 3: The depth of the flanging from 0 - 1.
Input 4: The center of the flanging from 0 - 1.

Phaser

Phaser is a set of 4 allpass filters whose delay length are modulated by a sine wave.

Input 1: Audio input.
Input 2: The modulation frequency of the phasing in Hz.
Input 3: The depth of the phasing from 0 - 1.
Input 4: The center of the phasing from 0 - 1.
Input 5: The amount of feedback across the allpass filters (0 - 1).

Compressor

Compressor performs dynamics compression of an input signal based on a trigger signal. For standard compression, hook the same signal to both the input and trigger. For sidechaining, you can use a different signal to compress the input. The compressor is designed to operate on audio signals varying from -1.0 to 1.0.

Input 1: The audio input to be compressed.
Input 2: The trigger signal to determine the amount of compression.
Input 3: The slope of the compression. This is the ratio of output signal to input signal above the threshold. 2:1 compression would be an input 0.5. 10:1 compression would be an input of 0.1.
Input 4: The threshold at which compression begins. This is a positive signal value in the trigger input which engages the compressor. Equal compression will occur on the negative side.
Input 5: The attack time in milliseconds. This is the time between the onset of compression and full compression.
Input 6: The release time in milliseconds. This is the time between the end of the attack (full compression) and no compression.

Pitch Shifter

Pitch Shifter is a pitch-synchronous overlap-add pitch shifter. It will move an input signal up or down in pitch. It works best for small pitch changes.

Input 1: Audio input.
Input 2: Pitch-shifting ratio. This is the ratio of the pitch of the output signal to the pitch of the input signal.

Waveshaper

Waveshaper uses an Ftable to perform non-linear waveshaping on an input signal.

Input 1: The audio input.
Input 2: The number of the transfer function Ftable if “Use Source List” is chosen.

Options: The transfer function Ftable can represent either the positive half of the function or the whole function. Use “Positive” for distortion systems, where the transfer function is diagonally symmetric across the origin. Use “Positive & Negative” for non-symmetrical functions, like Chebyshev functions.

Distortion

Distortion performs audio degradation and compression similar to a guitar stomp box. Distortion is designed to use an audio signal between -1 and 1.

Input 1: Audio input.
Input 2: The amount of distortion from 0 - 1.

Analysis

Analysis operators perform an analysis on the input signal and output the results.

Envelope Tracker

Envelope Tracker measures the power of the input signal and outputs an average of the power over an amount of time.

Input 1: The signal to be tracked.

Options: Envelope Tracker can be set to average over three time spans. "Fast" averages over 5 ms., "Average" over 10 ms., and "Slow" over 20 ms.

Debugging

Debugging operators display their input values in the **Debugging Palette**.

Debug

Debug displays its input value in the Debugging Palette. Depending on the speed of you Mac and the size of your patch, the display may run slightly behind.

Input 1: The signal to be displayed.

Options: Debug has six display modes:

Mode	Display
First	The first input value in each buffer.
Last	The last input value in each buffer.
Mean	The average value in each buffer.
Maximum	The maximum value in each buffer.
Minimum	The minimum value in each buffer.
Off	No value displayed.

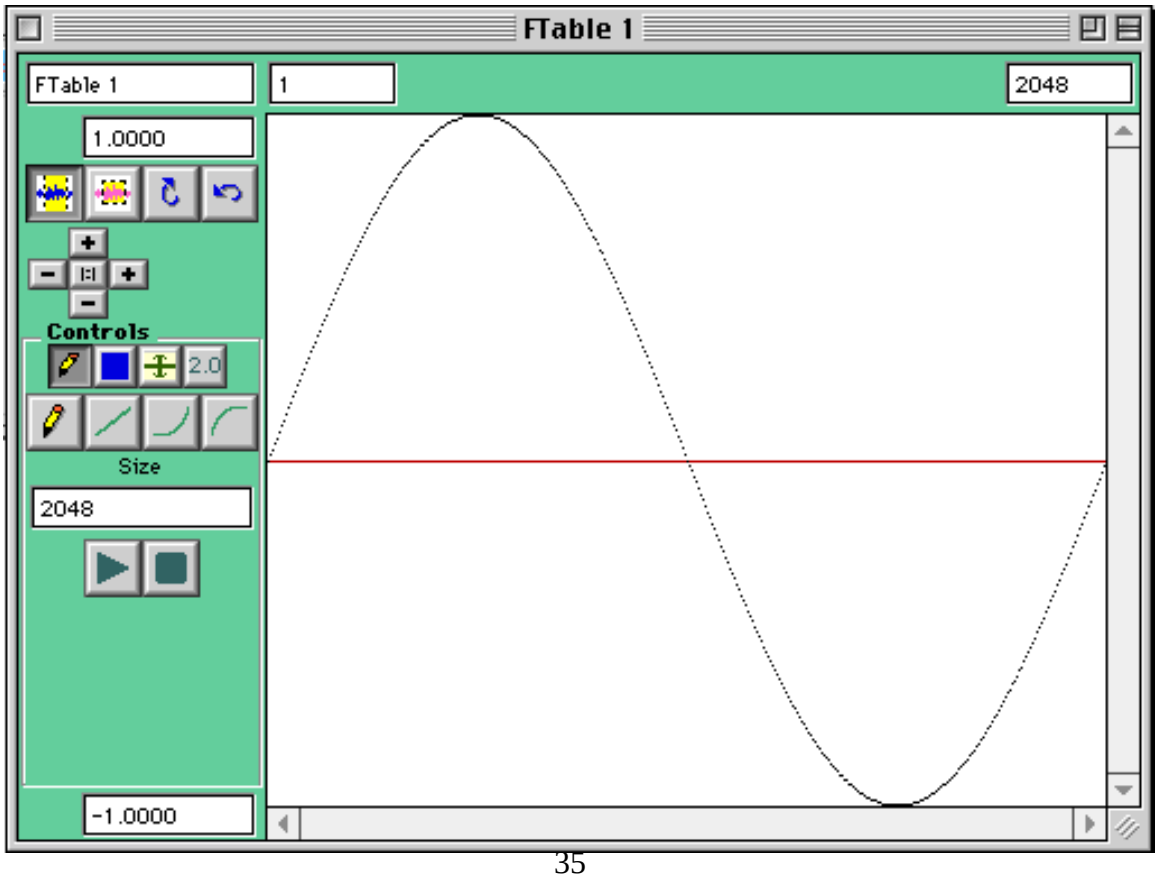
FTables

Function Tables

General

A function table is a list of floating point numbers. FTables serve many purposes in GrainWave. They may hold waveforms, scaling functions or envelope segments. The **FTable Reader** operator enables you to use FTables for whatever you wish. FTables can be edited graphically or mathematically using the **FTable Editor**.

When an operator requires an FTable, a popup menu will be shown listing all of the FTables current contained in the patch. If you wish to use one of these, simply choose it from the menu. If you need a new table, choose **Add** from the same menu. A new, default FTable will be generated. The shape of the FTable will be shown in the Ftable Button in the operator. To open an editor for the FTable, simply click on the FTable Button.



FTable Editor

The FTable Editor enables you to design your FTables. Standard draw, cut and paste operations are all possible. Only one value is allowed per horizontal step, however. The overall size of the table cannot be changed by cutting or pasting, only by using the **Size** edit field. If you cut a section out of a table, the remaining values will be shifted to the left and the table will be padded with zeroes to maintain its size. A paste will cause the rightmost values to be removed from the table.

The horizontal selection tool allows you to select all of values within a given range. The selection will be limited to values which are within the vertical boundaries of the window. Th box selection tool makes a more limited selection. Only values within the yellow box will be affected by any operation.

The flip buttons allow you to invert a selection either on the horizontal or vertical axis. These tools are particularly handy when you are designing envelope segments.

You can zoom in and out in either the vertical or horizontal directions using the zoom buttons. The 1:1 button will fit the entire table into the window. You can also zoom manually using the boundary edit fields located at the corners of the display. When you enter a new value into a boundary, the display will jump to that position. The mouse box displays the current x and y positions of the cursor while it is inside the display window.

Sub-Palettes

Four sub-palettes are contained inside the **Controls** section of the FTable Editor. When you open a new editor, the **Draw** sub-palette will be showing.

Draw Sub-Palette

The Draw sub-palette contains the drawing tools: Pencil, Line, Convex, and Concave. It also has the Table Size control and the playback controls. You can hear the sound of the table by using the play and stop controls. The table will be oscillated as if it were the table for a **Wave** operator at 200 Hz. The table will automatically be scaled to prevent clipping. This feature is enabled for all tables, though some may cause sounds that are unhealthy for your ears and your speakers.

If the FTable Editor is for a sound file, the sustain and release loop controls are also in the draw palette. Also, the Save-to-File button is available.

Fill Sub-Palette

The Fill sub-palette allows you to fill the selected region of the table with a mathematical function. The function can replace the existing data, or it can be added, subtracted, multiplied, or divided from the existing data. Some of the function require constants, such as an exponent or a partial number.

Transform Sub-Palette

The Transform sub-palette allows you to mathematically transform the selected data. For example, in the Draw sub-palette, the Sin control will fill the area with a sin. In this palette, Sin will take the sine of the existing data.

Data Sub-Palette

The Data sub-palette allows direct access to the data in the table. Simply choose an x coordinate and the corresponding y value will be displayed. You can enter new y values directly if you wish.

Table Functions

When an FTable Editor is open, the **Table Function** menu appears in the menu bar. These are special function generators which allow you to fill part or all of a FTable with a generated function. All of the functions only operate on the selected (yellow) portion of a table. The instructions for using each function generator are included in the generator dialog, so only an overview is presented here.

Partializer

The **Partializer** takes the selected portion of a table and treats it as one cycle of a fundamental waveform. You are then able to add harmonics at variable strengths and phases. If the base waveform is a single cycle of a sine wave, then the **Partializer** produces standard additive waveforms, in a similar manner to Gen 9, 10, and 19 in CSound.

Polynomializer

The **Polynomializer** creates a function of the basic form:

f(x) = ax^n +^(n - 1) + ... + yx + z.

You may also specify fractional exponents. The equation is evaluated over a user-specified range.

Chebyshev Generator

The **Chebyshev Generator** evaluates Chebyshev polynomials based on the desired partial strength. Chebyshev polynomials can be used with the Waveshaper operator.

Scale Generator

The **Scale Generator** can create step functions based on a series of ratios. The generator contains the following parameters:

Total Steps: The number of steps in the scale. In a MIDI scale, this would be 128.

Start Step: The initial step of the scale. In a 12 step equal-tempered scale, a start step of 1 would begin the scale on C, 3 on D, 5 on E, etc...

Step Size: The number of points in the function table that is devoted to each step. A standard MIDI mapping would have 1 point per step.

Reference Hz. and Reference Step: These determine the tuning of the scale. The reference step is an arbitrary location in the scale. The reference Hz. is the frequency of that step. In an equal-tempered MIDI scale, A440 corresponds to step 69. If the reference Hz. is 439, the entire scale is tuned down accordingly.

Scale Ratios: The ratios of each of the scale steps. The ratios are expressed relative to the first step of the scale. The first step is usually (1, 1). A fifth above would be (3, 2) or (1.5, 1). The steps are evaluated in the order they are entered, so it is possible to have non-sequential steps.

The **Scale Generator** repeats the sequence of steps once per octave. To enter an equal-tempered scale, it is necessary to have 12 steps. The generator will propagate those steps upward and downward from the reference step.

Loading and Saving FTable Data

An FTable Editor can load and save floating point data to/from a file. The data is interpreted as sequential 32-bit floating point numbers, without any header. Data of this type can be generated by CSound or by a specialized program.

Sound Files

General

GrainWave requires mono, 44.1 kHz., AIFF soundfiles for those operators which use soundfiles. Because of speed issues, a stereo file must be split into two mono files which are read independently. SoundHack is an excellent utility for splitting stereo files and converting other file formats into AIFF files. All of the soundfiles in a patch must be able to completely fit into RAM. Though GrainWave does not make any unasked for changes in the soundfiles it uses, I suggest using copies instead of originals.

Once a soundfile has been added to a patch, it can only be removed using the **Source Manager** (see **Palettes**).

Soundfile Editing

Soundfiles can be edited just like function tables. Additionally, when you open a soundfile in the editor, checkboxes will appear for the sustain and release loops for the file. You can set both loops using the selection tools. If you have the soundfile playing while you are setting the loops you will hear the loop as it will sound. You must explicitly save any changes you make to the soundfile (including loop points) by using the **Save File** button. The original file will be overwritten with the new changes. The addition of loop points may alter the size of the saved file, even though the sound data will be unchanged.

MIDI Reference

General

GrainWave is designed to be completely flexible within the constraints of MIDI. In general, MIDI data is separated into MIDI Events and MIDI Continuous Data.

MIDI Events

General

A MIDI Event is defined as a MIDI message which dictates a reaction from GrainWave. MIDI Events include note-ons, note-offs, and an assortment of specific MIDI commands.

Note Events

Note-on and note-off messages are used in a standard manner. Note-ons are the only way to trigger a note using MIDI. GrainWave accepts notes on all MIDI channels. Particular operators can be assigned to respond only to specific notes or ranges of notes. (see **Operator Reference: Triggerable Operators**). The note number and velocity of a note-on are available as continuous data within the triggered region. GrainWave reads note-ons with a velocity of 0 as note-offs.

Other Events

Reset All Controllers (Controller 121) will return the values of all of the controllers on the specified MIDI channel to 0.

All Notes Off (Controller 123) will end all current notes, regardless of the channel on which it is received.

Resync Timer (Controller 124) will cause the MIDI timer to catch up to the current buffer position. If there are audio buffer underruns, the timing will be adversely affected. This can restore the timing accuracy.

Program Changes will be interpreted by each region (see **Regions**).

MIDI Start (hex FA) will cause GrainWave to enter play mode.

MIDI End (hex FC) will cause GrainWave to exit play mode.

Continuous MIDI Data

General

Any MIDI message which updates a value in GrainWave is considered to be continuous. These messages include: controllers, aftertouch, pitch bend, and poly aftertouch. Note number and velocity are considered special cases of continuous data. All of the continuous data types can be accessed by a region using the **MIDI Reader** operator.

Controllers, Aftertouch, and Pitch Bend

GrainWave maintains the current values of controllers 0 - 120, poly and mono aftertouch, and pitch bend for all 16 channels. These values are maintained even when GrainWave is not in play mode. If you set a controller to 64 and then exit play mode, when you start playing again the controller value will still be 64.

14 Bit Controllers

GrainWave also reads controllers 0 - 63 as 14 bit controllers. The MSB is supplied by controller 0 - 31, and the LSB is supplied by the controller number + 32 (controllers 32 - 63).

Note Number and Velocity

Note number and velocity arrive bundled with note-on messages. Once a region is triggered by a note-on, the note number and velocity are available to that region through a **MIDI Reader** operator.

Palettes

General

GrainWave has four floating palettes which are used in conjunction with different parts of the program.

Operator Palette

The Operator Palette contains all of the operators available in GrainWave. Simply drag the desired operator in a region window.

Controls Palette

The Controls Palette the current function displays for GrainWave and contains the Master Volume control knob. The following information is displayed:

Volume: The current left and right output volumes are displayed on meters.
CPU: The amount of CPU power currently being used by GrainWave. Once the meter hits the yellow, you will begin to see other system functions degrading, like mouse movement and screen redrawing.
Memory: The amount of memory currently remaining in GrainWave's partition. If you run low, quit GrainWave and adjust its memory allocation by using ⌘I on the GrainWave program icon.
Playing LED: When GrainWave is playing, this will light up green.
MIDI LED: When GrainWave receives MIDI input this will light green.
Recording LED: While GrainWave is recording this will light green.

Source Manager

The source manager shows all of the FTables and sound files currently associated with a patch. FTables and sound files can be created or deleted using the source manager. Some relevant information is shown on each entry. The source manager also allows access to the **FTable Library** (see **Libraries**). FTables can be saved to the library or loaded from a library.

You may also reorder the Ftables or sound files in a patch using the Source Manager. Simply choose the Ftable or file that you wish to move and press “Renumber.” You will be asked to enter the number for its new location. This is useful for organizing Ftables and files for use with the “Use Source List” option.

Debugger

The debugger palette displays any **Debugger** operators which are currently enabled. It will only display data while you are in **Play Mode**. Each active **Debugger** operator will be listed. If the Max Polyphony is greater than 1, a duplicate entry will appear for each not of polyphony. These will be differentiated by a Poly #, starting at 0.

The current value of the **Debugger** will be listed under the value column. If the region containing the **Debugger** is not active, the value will be "Off". **Debugger** operators do slow a region's processing a bit, so they should be active only when necessary.

Libraries

General

Libraries are convenient repositories of frequently used kinds of data. The current libraries can be chosen in the **Preferences** dialog. New libraries can also be created here.

FTable Libraries

Each FTable Library stores up to 100 FTables. The FTables are saved by name and duplicate names are not allowed. Once you have chosen an FTable library, it will appear in the **Source Manager**. You can use the source manager to save an existing FTable to a library, or to load a new table from a library.

Scale Libraries

Each Scale Library stores the ratios and constants for up to 100 scales. When you create a new scale library, it will contain a standard equal-tempered MIDI scale. You can load a scale into the FTable editor by using the **From Lib.** button. You will be prompted to choose which scale you want. The **To Lib.** button allows you to save a scale you have created to the current scale library.

Customizing GrainWave to your System

General

GrainWave will perform differently on different computers. In order to provide the optimum environment for all users, I have given you control of many of the basic performance settings. These allow you to tailor GrainWave to your system and your needs.

Sound Input Settings

This dialog allows you to choose the source of sound input data into your computer. This is only available on pre-OS8.0 systems. After 8.0, you can set the input device on the Control Strip.

MIDI Setup

GrainWave uses Opcode's Open MIDI System™ (OMS) for MIDI communication. OMS is available for free from the Opcode Web site www.opcode.com. **Open OMS Setup** opens the current OMS studio setup document and allows you to alter it. **MIDI Connections** allows you to choose which external MIDI devices from which GrainWave will receive MIDI data. GrainWave automatically receives MIDI from any OMS-capable applications which are running on the same computer. Each program may have its own idiosyncrasies, however. For instance, Max requires that GrainWave already be running when Max start up in order for it to send to GrainWave.

Recording with GrainWave

Live Recording

GrainWave was designed to be a performance system. However, it is possible to record the audio output of GrainWave directly to your hard drive. These features are only available if your copy of GrainWave is registered

If your version of GrainWave is registered, the **Recording Setup** menu selection will be enabled. It will open a dialog box which allows you to choose name of the output file. When you are playing your patch, you can start recording by pressing the space bar. GrainWave will begin recording your output at the beginning of the file. Stop the recording by pressing the space bar again. If you record a second time, you will overwrite your first recording.

Standard MIDI File Rendering

GrainWave can also read a Standard MIDI File and turn it into a stereo AIFF sound file. This process does not occur in real-time, and therefore the complexity of the instruments and the amount of polyphony is not dependent upon the speed of your machine.

In general, SMF Rendering operates identically to real-time MIDI playback. GrainWave interprets whatever MIDI commands it finds in the file. GrainWave will only react to MIDI commands that are understood by the region. For instance, if there is data on MIDI channel 5, but no region uses that data, it will be ignored. When rendering, GrainWave also can provide absolutely exact timing.

Upgrading from GrainWave 2.x

General

This chapter is intended for experienced users of GrainWave 2.x who wish to know how GrainWave 3 differs and how to import their 2.x patches.

Structural Changes

GrainWave 3 has a new audio generation engine. The engine is much more flexible (though slightly slower) than the GrainWave 2 engine. In particular, connections between operators are now concrete objects. This means that more than one connection can be made to a single input. Also, feedback patterns are now completely legal. All feedback will be delayed by 4 samples at audio rate.

Here is a specific list of GrainWave 3 changes:

Buffers + resolutions: The audio buffer sizes in GrainWave are now optimized for OS8.1 and higher. The audio out buffer is always 256 samples. Also, there are fewer available operator resolutions: audio, 4, 16, buffer, and note rates.

Missing Operators: The SubMix operator is gone. Since you can now make multiple connections to a single input, it is no longer necessary. The RTSampler has been replaced by two more general operators, Ftable Writer and Ftable Player. Plugin is gone since version 2 plugins are not supported in version 3.

New Operators: There is a new Mathematical operator, which has many standard operations, including trig and log functions. There are two new effects (Flanger and Phaser) and two new filters (Resonant Lowpass and Allpass). There is a new Oscillator operator, which has band-limited square, triangle, and sawtooth waves. Mono Out sends the same signal to both output channels.

Global regions gone: Both Global regions are gone. Instead, you can pass signals between any two regions, provided that the

receiver is placed lower in the patch window than the sender. All regions can now be set to “Always On,” which behaves as the global regions did (only one note, triggered when you switch to play mode).

Source manager renumber: You can now reorder the lists of Ftables and sound files to make the use of source lists easier.

MIDI Map gone: The MIDI Map has been replaced by trigger maps inside every triggerable operator.

Keyboard triggering: You can now trigger notes using the computer keyboard.

Signal triggering: You can now trigger operators inside a region using signals from other operators.

Envelope changes: Envelope now only has attack, decay, and release segments, instead of an arbitrary number of attack and release segments. Also, Envelope is now a triggerable operator.

General panel gone: To set the regions variables, choose **Region Info** from the **Setup** menu.

Master volume knob: You can now adjust the master volume from the Controls palette.

Play while editing: You can edit patches while you are in play mode.

Using Version 2.x patches

GrainWave 3 has a patch converter which allows you to convert version 2.x patches to version 3 format. However, the patches will require a little editing once they are converted.

Stereo Out hookups: Any Stereo Out operators will not have their inputs hooked up. This is because the Stereo Out operator now takes separate left and right inputs.

Global Regions: The Global regions will appear as small standard regions in the version 3 patch, even if the version 2 patch did not use them. You also may need to move the Post-Global region further down in the patch window for proper functionality.

Plugin and RTSampler: Neither of these operators exists in version 3, so patches using them will not convert properly.

Future Plans and Credits

Future Plans

GrainWave is an ongoing project. Here are some of my upcoming additions:

More Effects: I have plans to tackle a reverb operator.

Pitch Tracker: To go with the envelope tracker.

Your input is valuable. If you let me know what you need in order to use GrainWave, I will do my best to oblige.

Credits

GrainWave 3 would not have been possible without the Mills College Center for Contemporary Music. In addition to providing a wonderful, creative environment, it also provided a key grant to support the development of GrainWave 2. Specifically, Dave Madole taught me programming, and Chris Brown has been invaluable for design suggestions and intensive software testing.

I also want to thank those of you who registered GrainWave 1 and 2. Your encouragement has pressed me to make GrainWave 3 as useful as possible.

Finally, my wife Catie bought me the computer, lent me the time to write the program, and even proof-read the manual.

Version Notes

3.0.0 6-1-99

-Initial release of version 3.

2.3.1 12-7-98

-Better MIDI timing, removed fudge factor.
-A number of bug fixes.
-Improved the delay tap operator.

2.3.0 7-14-98

-Rebuilt the UI for faster editing.
-Added the pre-global region.
-Added real-time table and sound-file switching.
-Improved the quality and flexibility of the Filter operator.
-Fixed bug in Delay Line.
-Consolidated the FTable palettes into the FTable Editor.
-Added direct data editing to the FTable Editor.
-Numerous bug fixes in the FTable Editor and Source Manager.
-Added direct operator editing in the Structure window.
-Fixed bug in Live Recording which improves performance.
-Playing and MIDI LED’s now light green for visibility.

2.2.0 4-14-98

-Added text editing and compilation.
-Fixed bug in submix and stereo out.
-Improved MIDI timing mechanism and added Fudge Factor.

2.1.0 2-17-98

-Added Standard MIDI File Rendering.

-Added a debugger.
-Added context-sensitive definitions and help.
-Added note rate resolution.
-Added Accumulator operator.
-Added several ftable generators (Partializer, Polynomializer, Chebyshev Generator) and moved the scale generation.
-Fixed OMS Connections dialog and hopefully improved OMS reliability.
-Added Combined Package file format.
-Improved display quality of structure diagram.
-Auto-numbering of operators is now individual to each operator type.
-Fixed bugs in region display.
-Added region pixel sizes to region display.
-Fixed bug in plug-in operator display.

2.0.7 12-08-97

-Added OMS Connection interface.
-Audio and MIDI settings now saved in Patch files.
-Added windowing to Grain operator.
-Fixed bug in multi-segment envelopes.
-Fixed fatal bug in Submix when the input resolution is high.
-Fixed bug in Wave which caused the truncating button to be inactive.
-Unaltered patches will not ask to be saved upon quitting.

2.0.6 11-03-97

-Added Pitch Shifter operator.
-Fixed fatal bug in startup which caused a crash if OMS was not installed and which prevented GrainWave from being opened with a patch file.
-Fixed bug which caused an occasional crash when stopping playing a patch with a RTStream operator.
-Fixed bug which caused improper display of Delay Tap time.
-Fixed Switch icon in structure diagram.
-Fixed bug in Butter which showed wrong menu choices on startup.

2.0.5 10-21-97

-Changed Random to a extremely long period random function.
-Fixed bug when playing samples.
-Fixed fatal bug while playing a moving Delay Tap.
-Fixed fatal bug when adding a new Delay Tap.
-Fixed bug in preference file.

2.0.4 10-10-97

-Added Distortion and Envelope Tracker operators.
-Added Plug-In operators and API.
-Added loading/saving of raw floating point data files into/out of ftables.
-Major optimization work.
-Fixed fatal bug when soundfiles exceeded available memory.
-Fixed bug reading some AIFF files (e.g. SoundEdit 16 files)
-Fixed bug writing loop points to AIFF files.
-Fixed bug when deleting a global inlet.
-Fixed menu bugs in Source Manager and MIDI Map.
-Fixed bug in structure diagram which caused some op's to be off the left edge of the diagram.
-Fixed bug in mixer menus when you reopened a region's window.
-Fixed bug when quitting when you had region windows open.
-Added a delay to the MIDI in LED making it easier to see.

2.0.3 9-17-97

-Fixed fatal bug in Global Inlet.
-Added a comments section to each region.

2.0.2 9-15-97

-Accelerated screen redrawing in Edit Mode.
-Fixed wrong icon for RT Stream.
-Fixed bug when starting GrainWave from a patch.

-Added Pulse operator.
-Added Multiply and Divide to Fill Palette.
-Made RT Stream operate in stereo.

2.0.1 9-5-97

-Fixed bug in MIDI Reader for pitch bends and 14bit controllers.
-Fixed fatal bug in Delay Tap.
-Fixed redraw bug in Grain.
-Added ZeroToOne scaling for file position to Grain and RTSampler.
-Added Beta scaling to FeedbackFM. If pressed, the feedback value is scaled by the current carrier frequency.

2.0.0 8-25-97

-Initial release.