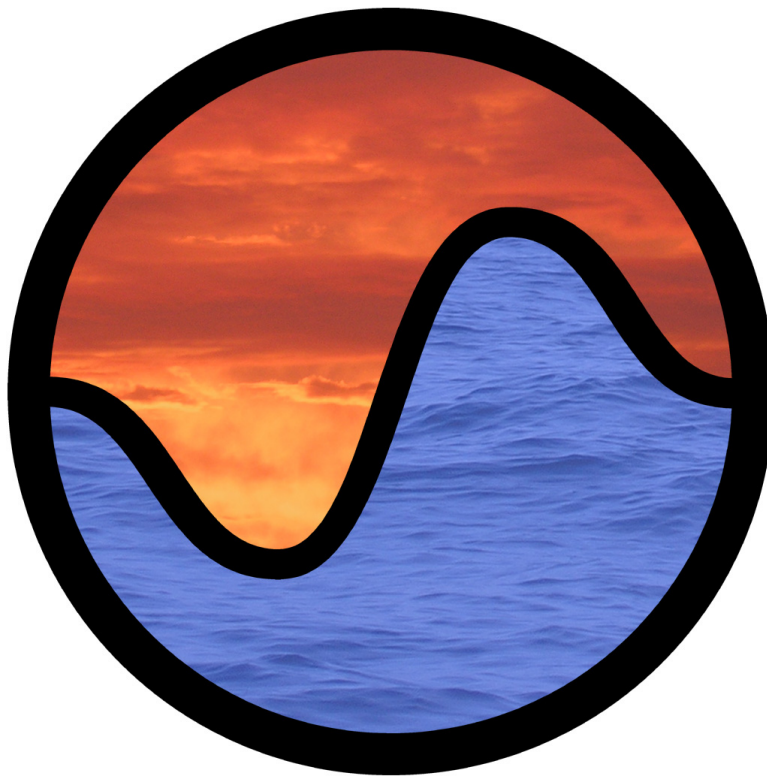


SonicBirth Manual



Contents


1	License	1
2	Introduction	2
3	Installation	3
3.1	The framework	3
3.2	The plugins	3
3.3	The application	3
3.3.1	Entering the registration number	3
4	Quick tips	4
5	Overview	5
5.1	Terms	5
5.1.1	Wire	5
5.1.2	Element	5
5.1.3	Argument	5
5.1.4	Circuit	6
5.1.5	Root circuit	6
5.1.6	Model vs plugin	7
5.2	Workflow	7
5.2.1	Loading a sound	7
5.2.2	Setting up inputs and outputs	7
5.2.3	Inserting elements and patching them	7
5.2.4	Feedback	7
5.2.5	Inserting arguments	8
5.2.6	Setting up the plugin attributes	8
5.2.7	Exporting the plugin	9
5.2.8	Sharing your creation	9
6	Basic windows	10
6.1	Document	10
6.1.1	Level	10
6.1.2	Size	10
6.1.3	Mini	11
6.2	Sound	11
6.3	MIDI	11
6.4	Settings	12
6.5	Information	12
7	Important elements	13
7.1	Circuits	13
7.1.1	Root circuit	13
7.1.2	Subcircuit	14
7.1.3	Piecewise circuit	14

7.2	Internal arguments	15
7.2.1	Bit precision	15
7.2.2	Interpolation precision	15
7.2.3	Midi settings	15
7.3	Arguments	15
7.3.1	Boolean	16
7.3.2	Indexed	16
7.3.3	Slider	16
7.3.4	Points	17
7.3.5	Points Envelope	17
7.3.6	Points Frequency	17
7.3.7	XY Pad	17
7.3.8	Audiofile argument	17
7.4	Midi arguments	18
7.4.1	Midi slider	18
7.4.2	Midi mono note	18
7.4.3	Midi multi note	18
7.4.4	Midi multi note env	18
7.5	FFT elements	18
8	Circuit examples	19
8.0.1	The simplest circuit: passthrough	19
8.0.2	Stereo to mono	19
8.0.3	A multiband reverb	20
8.0.4	A software synth	21
9	Custom gui	24
9.1	Designing the interface	24
9.1.1	Placing the elements	24
9.1.2	Selecting the colors	24
9.1.3	Loading a background picture	24
9.2	Testing the interface	25
10	List of elements	26
10.1	Algebraic	26
10.1.1	Addition	26
10.1.2	Add Many	26
10.1.3	Substraction	26
10.1.4	Multiplication	26
10.1.5	Division	26
10.1.6	Constant Addition	26
10.1.7	Constant Substraction	26
10.1.8	Constant Multiplication	27
10.1.9	Constant Division	27
10.1.10	Constant Substraction Alt.	27
10.1.11	Constant Division Alt.	27

10.1.12	Negation	27
10.1.13	Inverse	27
10.1.14	Absolute	27
10.1.15	Absolute/Sign	27
10.1.16	Direction	28
10.1.17	Multiply-Add	28
10.1.18	Multiply-Sub	28
10.1.19	Neg. Multiply-Add	28
10.1.20	Neg. Multiply-Sub	28
10.2	Function	28
10.2.1	Modulus	28
10.2.2	Ceil	28
10.2.3	Floor	28
10.2.4	Nearest integer	29
10.2.5	Exponential	29
10.2.6	Natural logarithm	29
10.2.7	Logarithm base 10	29
10.2.8	Power	29
10.2.9	Square root	29
10.2.10	Reverse square root	29
10.2.11	Quadratic Bezier	29
10.3	Trigonometric	30
10.3.1	Sinus	30
10.3.2	Cosinus	30
10.3.3	Tangent	30
10.3.4	Sinus and cosinus	30
10.3.5	Hyperbolic sinus	30
10.3.6	Hyperbolic cosinus	30
10.3.7	Hyperbolic tangent	30
10.3.8	Arc sinus	30
10.3.9	Arc cosinus	31
10.3.10	Arc tangent	31
10.3.11	Arc tangent (x, y)	31
10.3.12	Inverse hyperbolic sinus	31
10.3.13	Inverse hyperbolic cosinus	31
10.3.14	Inverse hyperbolic tangent	31
10.4	Arguments	31
10.4.1	Boolean	31
10.4.2	Slider	31
10.4.3	Indexed	32
10.4.4	Points	32
10.4.5	Points Envelope	32
10.4.6	Points Frequency	32
10.4.7	XY Pad	32
10.4.8	Audio file argument	32
10.4.9	Keyboard Tap	32

10.5	Midi arguments	32
10.5.1	Midi slider	32
10.5.2	Midi note state	33
10.5.3	Midi mono note	33
10.5.4	Midi multi note	33
10.5.5	Midi multi note envelope	33
10.5.6	Midi XY Pad	33
10.6	Display	33
10.6.1	Display value	33
10.6.2	Display Osc.	33
10.6.3	Display Osc. (Var. res.)	33
10.6.4	Display Meter	34
10.7	Analysis	34
10.7.1	Envelope Follower	34
10.7.2	Look ahead	34
10.7.3	BPM Counter	34
10.7.4	Debug	34
10.7.5	Debug Osc.	34
10.8	Comparators	34
10.8.1	Minimum	34
10.8.2	Maximum	35
10.8.3	Sort	35
10.8.4	Less	35
10.8.5	Equal	35
10.8.6	Greater	35
10.9	Converters	35
10.9.1	Msec to samples	35
10.9.2	Samples to msec	35
10.9.3	Linear to db	35
10.9.4	Db to linear	36
10.10	Delays	36
10.10.1	Delay	36
10.10.2	Delay (samples)	36
10.10.3	Delay Sinc (samples)	36
10.11	Generators	36
10.11.1	Sine Wave	36
10.11.2	Fast Sine Wave	36
10.11.3	Saw Wave	36
10.11.4	Triangle Wave	37
10.11.5	Square wave	37
10.11.6	Linear Noise	37
10.11.7	White Noise	37
10.11.8	Pink Noise	37
10.11.9	Random	37
10.11.10	Random ramp	37
10.11.11	FFT Generator	37

10.12	Filters	38
10.12.1	DC Blocker	38
10.12.2	Parametric Eq.	38
10.12.3	Peak	38
10.12.4	Notch	38
10.12.5	Lowpass	38
10.12.6	Highpass	38
10.12.7	Resonant lowpass	38
10.12.8	Resonant highpass	39
10.12.9	Crossover	39
10.12.10	Bandstop	39
10.12.11	Bandpass	39
10.12.12	Lowpass (fast)	39
10.12.13	Highpass (fast)	39
10.12.14	Resonant lowpass (fast)	39
10.12.15	Resonant highpass (fast)	39
10.12.16	Crossover (fast)	40
10.12.17	Bandstop (fast)	40
10.12.18	Bandpass (fast)	40
10.12.19	Allpass	40
10.12.20	Feedback Comb	40
10.12.21	Feedforward Comb	40
10.12.22	Formant filter	40
10.13	Feedback	41
10.13.1	Feedback	41
10.14	Distortion	41
10.14.1	Valve	41
10.14.2	Scraper	41
10.14.3	Scraper (quick)	41
10.15	Audio file	41
10.15.1	Audio file	41
10.15.2	Audio player	41
10.16	FFT	42
10.16.1	FFT Sync	42
10.16.2	Forward FFT	42
10.16.3	Inverse FFT	42
10.16.4	Complex to Polar	42
10.16.5	Polar to Complex	42
10.16.6	Convolve	42
10.16.7	Points To FFT	42
10.16.8	Audio file To FFT	43
10.17	Miscellaneous	43
10.17.1	Circuit	43
10.17.2	Constant	43
10.17.3	Equation	43
10.17.4	Piecewise circuit	43



10.17.5	Samplerate Doubler	43
10.17.6	Timer	43
10.17.7	Timer loop	44
10.17.8	Freeverb	44
10.17.9	Cleaner	44
10.17.10	Points apply	44
10.17.11	Bufferizer	44
10.17.12	Trigger	45
10.17.13	Change Slower	45
10.17.14	Convolving reverb	45
10.17.15	AudioUnit	45
10.17.16	AudioUnit (midi)	45
10.17.17	Granulate effect	45
10.17.18	Granulate effect with pitch	46
10.18	Internal	46
10.18.1	Interpolation Precision	46
10.18.2	Bit Precision	46
10.18.3	Midi settings	46

List of Figures

1	Wire anchors	5
2	A typical element	5
3	A typical argument	6
4	The circuit hierarchy	6
5	A simple feedback loop with delay and amount controls	8
6	Initial state of the document window	10
7	The sound window	11
8	The MIDI window	11
9	The settings window for the look ahead element	12
10	The information window for the parametric EQ element	12
11	The boolean argument	16
12	The indexed argument	16
13	The slider argument	16
14	A passthrough circuit	19
15	A stereo to mono circuit	19
16	A multiband reverb circuit	20
17	Inside multi midi note	21
18	A rainy software synth	22
19	Custom gui settings	24

1 License

Copyright ©2005, Antoine Missout. All rights reserved.

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

Permission is granted to use a registration number you bought on up to three computers you own.

Permission is granted to redistribute a plugin exported with SonicBirth, as long as the plugin is redistributed as it was exported, without modification. Keep in mind that potential users will have to download the framework from the official website.

2 Introduction

Welcome to SonicBirth! This manual will guide you through the installation and use of SonicBirth. The suggested learning path is to install the software, try the plugins, see how they work, then modify them or create your own. SonicBirth is the first application dedicated to the creation of AudioUnit plugins. Although most of the work is done transparently by the application itself, it is necessary to grasp a few basic technical aspects in order to create plugins. Once you'll have gone through it once, and tried it out a bit, you will find that it is a small price to pay for the immense freedom that SonicBirth allows you to have over your sound.

SonicBirth works by patching basic elements together to form complex audio effects and instruments. This design is done in a standalone application, from which you can export your model to a plugin. This plugin can then be used as any other AudioUnit.

3 Installation

First things first: getting SonicBirth to work. This is very easy. Just launch the installer, and click install. It will automatically remove any previous version of SonicBirth and install the new one. Here is a small explanation of what is installed:

3.1 The framework

The framework contains SonicBirth's engine. It is shared between all the plugins and the application. Installation of the framework is mandatory. The application and the plugins won't load if it is not correctly installed.

3.2 The plugins

Installing the prebuilt plugins is optional. You are encouraged to install them.

3.3 The application

The installation of the application is also optional, although to enter your registration number you will have to launch it at least once.

3.3.1 Entering the registration number

To register both for the plugins and the application, simply launch the application, click register, **copy/paste** your name and registration number **exactly as you received it in your order email**. Once the registration is confirmed, you'll be rid of the "demo version" sound.

4 Quick tips

Some tips *en vrac* :

- SonicBirth was designed with AudioUnits in mind, therefore the VST interfaces may be slightly inferior in quality, though they should sound the same. To export in VST format, simply hold the option key while choosing the export menu item.
- Logic 7.0 cannot load cocoa AudioUnit views, which SonicBirth uses. Please upgrade to Logic 7.1.
- The points box: left/right arrows to change interpolation type, double-click to create a point, press delete to remove it.
- Swamp buffer: C4 plays, D4 records.
- Double-click a wire anchor to delete it.
- Drag a wire away from its element input to delete it.
- Even if the MIDI server is configured, you will not get any sound unless the sound server is started.
- Right or control-click to insert an element at mouse point.
- Multi midi note [env]: Do not forget to stop/start for changes to take effect.
- Use the batch export feature of SonicBirth to quickly upgrade all your plugins.
- The subcircuit you will enter for a piecewise circuit depends on the row selected in the settings window, not on the actual value of the input (since this value could be unknown, if connected to a sound input for example).

5 Overview

The main purpose of SonicBirth is to allow you to create your own AudioUnits. You'll be using the application to do the actual plugin design, then exporting it as a plugin, and then installing it for use in your host application. Before we go on, we will introduce some basic terminology.

5.1 Terms

5.1.1 Wire



Figure 1: Wire anchors

Wires are used to connect the basic blocks together. You can anchor a wire at any point by clicking on it. To remove the anchor, double-click on it.

5.1.2 Element

An element is a basic block you can work with. It can have either none or multiple inputs, and one or more outputs. For example, an *Addition* element has two inputs and one output.



Figure 2: A typical element

5.1.3 Argument

An argument is a special type of element. *Slider*, *Boolean*, *Indexed* are your basic arguments. *MIDI slider*, *MIDI mono note*, *MIDI multi note* are the basic MIDI arguments. The difference with an argument is that it will appear as a parameter in your final exported plugin, whereas elements won't show. You can set custom names for arguments. These names will also appear in the plugin.



Figure 3: A typical argument

5.1.4 Circuit

A circuit (or subcircuit) is also a special type of element. A circuit is a group of elements patched together with wires (hence the name circuit). A circuit can be inserted into another circuit. The buttons *Next* and *Prev.* level allow you to go into and out of a circuit. You can specify the name as well as the number of inputs and outputs of a circuit. Be aware that an argument hidden inside a subcircuit (a circuit inside another one) will not show up in the plugin.

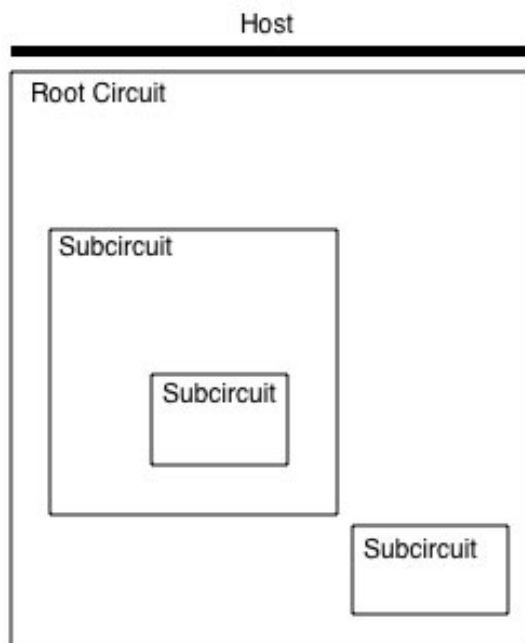


Figure 4: The circuit hierarchy

5.1.5 Root circuit

The root circuit is simply the circuit at the base of your AU plugin. It will receive audio from the host on its inputs, and the host will get audio back from

its outputs. The root circuit also has many specific attributes that a basic subcircuit doesn't have: author, comments, latency, tail time, etc. The root circuit may also be called plugin since it is the representation of the plugin that is created.

5.1.6 Model vs plugin

The model is the `.sbc` file, while the plugin is the exported `.component` file.

5.2 Workflow

Now that the terms are given, here are the basic steps for creating a plugin.

5.2.1 Loading a sound

If you intend to make a plugin that will process sound from the host, you should begin by loading a sound to test your circuit with. Obviously, if you are making a plugin that creates sound from scratch, there's no need to load up a sound. Loading a sound is done by clicking on the em Open button on the sound panel. If the sound panel isn't on screen, you can use the *Window* menu to bring it on screen. You can load AIFF or WAV files. The audio device used for playback is the default audio device.

Please note: If you load a sound that is not stored at the same sample rate than your audio device, SonicBirth will **not** resample it. It will play at a different pitch. This is to make sure no resampling artifacts are introduced that could be confused with your circuit artifacts.

5.2.2 Setting up inputs and outputs

What you will want to do first is specify the number of inputs and outputs of your root circuit. You can do so by clicking on the background, which will select the currently displayed circuit. Its settings will appear in the Settings window, allowing you to fill in the appropriate textfields.

5.2.3 Inserting elements and patching them

This is the heart of the design. Insert elements, patch them together, create new sounds. You can patch elements together - create wires - by clicking an output, dragging the mouse, then releasing it on the input of another element. However, you cannot connect the input of an element to the output of the same element to create a circular or feedback loop, unless...

5.2.4 Feedback

Unless you use a feedback element in between, which inserts a small, variable delay. You can make a simple reverb by using a few feedback loops with different delays and gains.

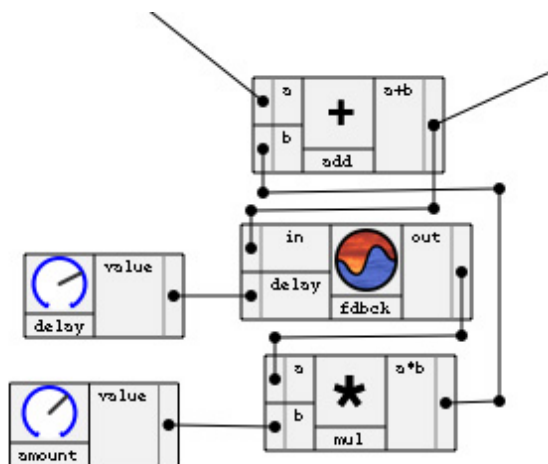


Figure 5: A simple feedback loop with delay and amount controls

5.2.5 Inserting arguments

If you want your plugin to have variable parameters, you have to insert arguments which you should connect to the inputs of the elements you want to control from the host that will load the plugin. You can for example connect the output of a slider to an input of a multiplication, and connect an audio feed the other input of the multiplication ; the result of the multiplication will then be to allow for a variable linear gain.

5.2.6 Setting up the plugin attributes

There are a few root circuit attributes you must specify before exporting it as a plugin:

- Name: This will be the name of the plugin (the file), as well as the name that will show up in the host. Usually you will want to keep it short, to think of a good name, and to avoid special characters.
- Subtype: This is a four character identifier that must be unique across all plugins. It is case sensitive. You should only use basic lower and upper case letters, numbers and punctuation in this field. Subtypes beginning by a capital 'S' are reserved for the prebuilt plugins.

By default, SonicBirth will create a random one, which should suffice in most cases. In the rare case where two plugins have the same subtype, one of them will not load. The subtype is unavoidable, it comes from the AudioUnit standard.

5.2.7 Exporting the plugin

Ready to try our your plugin in your usual audio host? Use the *File* menu, click export, and choose a destination. You can then install it in the appropriate location. Simple. (To export in VST format, simply hold the option key while choosing the export menu item.)

5.2.8 Sharing your creation

Made something great? Would like to share it? Please do so! But keep in mind a few things: you can distribute the circuit model file (which has a file extension `.sbc`) or the actual exported plugin. If you redistribute the exported plugin, you have to redistribute it exactly as it was exported (though you can compress it in a format suitable for redistribution, obviously). Users will still have to install SonicBirth. This is because if you give the plugin to someone who doesn't have SonicBirth installed, it will not work since the framework will not be present. For this reason, other users will need to download SonicBirth through <http://sonicbirth.com>. This will also ensure the person has the latest version available, which might contain important fixes. Doing so will ensure the best user experience for the plugins you created.

6 Basic windows

This section describes the basic windows which constitute the graphical interface. The sound, MIDI settings and information panels can be displayed on screen with the *Window* menu.

6.1 Document

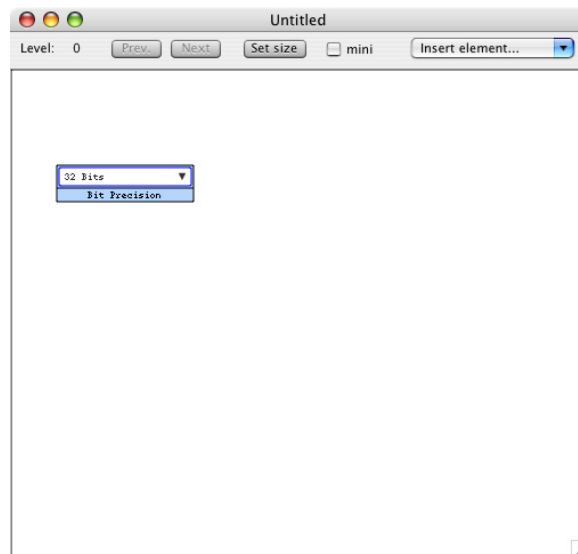


Figure 6: Initial state of the document window

The document window is the window displaying your circuits. Note that if the circuit minimum size is greater than the current window size (scrollbars are visible), you can click and drag the background to scroll. At the top of the window, there are a few gui elements that apply to the current circuit.

6.1.1 Level

This is the current nesting level. The root circuit is always at level 0. If the displayed level is greater than that, it means the window is displaying a circuit inside another circuit (a subcircuit). You can click *prev.* to see the circuit's parent, or *next*, if a circuit is currently selected, to enter it.

6.1.2 Size

This sets the circuit minimum size to the current size of the window, thus displaying scrollbars if it gets smaller. When working in the root circuit in

custom gui mode, this will set the custom gui size for the plugin as it will be loaded by your host.

6.1.3 Mini

The mini attribute toggles the way elements are displayed. You will find it useful for large or complex circuits. Or if you just like minimalism. You can then use the information window to distinguish between inputs and outputs.

6.2 Sound



Figure 7: The sound window

This panel displays information on the current output device, current loaded file, and audio processing state. For the device, it shows the number of channels, the sample rate and the latency. The latency by default is set to 100ms.¹

Even if no sound is loaded, you can still press play. If your circuit has any inputs, it will receive silence. You can load a sound in AIFF or WAV format, and use it to test your circuit. Check section 5.2.1 for more details.

6.3 MIDI



Figure 8: The MIDI window

This panel allows you to select which MIDI source is used to send MIDI events to the MIDI arguments of your circuits. This list will be updated when a source is added or removed while SonicBirth is running.

¹The default latency is deliberately high: SonicBirth is not intended to be used as an audio processing application, but really just an AudioUnit design application.

6.4 Settings

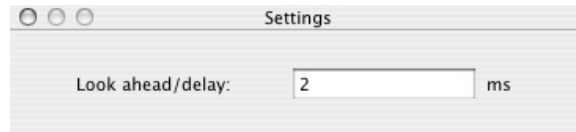


Figure 9: The settings window for the look ahead element

The settings panel shows specific informations about the currently selected element, or the displayed circuit if no element is selected. More details about these settings are given in the next section for the important elements. Most elements don't have any settings.

6.5 Information

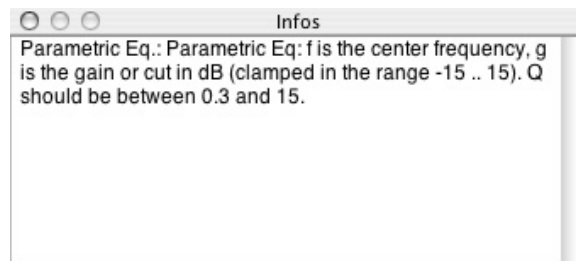


Figure 10: The information window for the parametric EQ element

The information panel displays useful information about either the selected element or the circuit (if no element is selected). Think of it as an inline help.

7 Important elements

This section gives more in-depth information about important elements you will use while designing an AudioUnit.

7.1 Circuits

7.1.1 Root circuit

This is the element that is at the base of your plugin model. This is what is seen by the host. There is always one and only one element of this kind, at level 0. It is also the element with the most settings:

- **Name:** This is the name of the circuit as it will appear on disk and in the plugin host. It is limited in length (128 characters) and you cannot use special characters in it.
- **Author:** This is an optional field where you can write the author's name. Useful if you intend to share your model file.
- **Company:** This is the company name that will appear in the generic graphical interface, and anywhere the host may use it.
- **Description:** Here you should give a short description of your plugin, though most (all?) hosts ignore this information.
- **Subtype:** This should be a unique four characters identifier. This comes from the way AudioUnits are listed by the operating system. You can use upper and lower case letters, numbers and most punctuation marks. Subtypes beginning by a capital 'S' are reserved for prebuilt plugins.
- **Inputs:** Here you can enter the number of inputs your plugin will have.
- **Outputs:** Here you can enter the number of outputs your plugin will have. A special case is when both the input and output count as one. In this case, the plugin will duplicate your circuit as needed, that is, you will be able to use it in an n to n (2/2, 3/3, 4/4, ...) configuration in your host.
- **Latency:** This is used by the host to compensate for any latency in the circuit (delay line, lookahead, etc). This should be as precise as possible. If you are making an AudioUnit where the delay is intended, leave this to zero, since you do not want the host to compensate for it. The total latency given to the host will be the sum of the latency in milliseconds and the latency in samples (converted at runtime to milliseconds, depending on sample rate).
- **Tailtime:** This is also used by the host to know how long your plugin can generate sound once its inputs have been silenced. You can use a conservative (but not extreme) value, it needs not be precise. For example, in a reverb circuit you might set this to 2000 or 3000 ms.

- **Comments:** Like the author field, this is optional, and should mainly be used in case of redistribution, or as a small note pad.
- **Circuit min. size:** This specifies the size under which scrollbars will appear in the circuit window. Should normally be set to a size where the circuit would be incorrectly displayed if the window is smaller.
- **Tempo and beat:** The root circuit option, if enabled, will add two inputs to the root circuit, tempo and beat. This information is given by the host, if supported. Otherwise those inputs will be 0.
- **Sidechain:** Some hosts support AudioUnits with multiple busses. SonicBirth can make plugins with 2 busses: the main one, and a sidechain one. Considering some hosts may not support sidechains, you should add a toggle to use or not the sidechain, if possible.
- **Arguments:** You can specify the order in which the arguments will show up in the generic view in the host. Even if you use a custom gui, you should take a few seconds to reorder your arguments in case the host does not support custom gui.
- **Presets:** If you want to include presets with your plugins, you can specify them. Clicking create will take a snapshot of the arguments current values and store them. You can then specify the name for the preset you create. As with the arguments, you can specify the order in which they appear. By clicking set, the preset will be applied to the circuit, restoring the saved values.
- **Custom gui button and size:** By enabling the custom gui button, you will be able to specify your own gui, using a background picture (or not) and by placing the argument in the gui design mode, then testing it in the runtime mode.
- **Front, contour and back colors:** Even if you do not want to make a custom gui, you can change the colors of the arguments to something that suits you better.

7.1.2 Subcircuit

A subcircuit is a circuit which can be inserted in another one. It might be useful to group several elements in a logical group. It is also a way to hide arguments from the host. You can specify a few comments on a subcircuit explaining what it contains, if you want. It makes it easier to reuse.

7.1.3 Piecewise circuit

This is a complex circuit to answer a problem. Is is best explained by an example. Let's say you have a circuit, where you want the type of filter applied, lowpass or highpass, to be user selectable by an indexed argument. You could

calculate both then use a comparator to select the value that will be used depending on the argument. But by doing so, you will calculate the value of the other filter for nothing, especially since the filter type will not be changed often (compared to the sample rate of the audio).

The answer is to create a piecewise circuit, which has a range input. In its settings window, you can specify many ranges. For each of them is associated a subcircuit. When you click *Next* on the document window, it will enter the currently selected range.

The number of inputs and outputs the subcircuit will have can be specified in the piecewise circuit window. For our example, a range would be created for the lowpass and one for the highpass, then you would enter each subcircuit/range to configure it.

The main difference with the comparator solution is the correct filter is selected one time by rendering cycle (typically a few milliseconds) and the other is not calculated.

7.2 Internal arguments

Some elements are automatically inserted in your root circuit depending on its content. You cannot delete or insert these elements yourself.

7.2.1 Bit precision

Your root circuit will always contain an element of type *Bit precision*. This is used to set the current precision of the rendering engine.

7.2.2 Interpolation precision

If any of the elements in the root circuit, or any subcircuits, interpolates the data, this element will appear. There are two modes: no interpolation and linear interpolation. Using no interpolation is faster, but may have a lower quality sound. Using linear interpolation will sound better but will use more cpu. Depending on the amount of interpolation done, and the number of elements you use that interpolate, the difference in speed can vary.

7.2.3 Midi settings

If you use any MIDI arguments, this element will be automatically added to your root circuit. Instead of having a channel and controller parameter for each one of your midi arguments, these settings are controlled by this central element.

7.3 Arguments

The arguments are the elements that will appear as parameters in your final plugin. There are three types of basic elements, with more to come for future releases. Every argument has name that can be user specified, and a realtime

switch which tells the host if the associated parameters should be changed while playing audio or not (some hosts may completely ignore this).

7.3.1 Boolean

A boolean argument is an on/off switch.



Figure 11: The boolean argument

For both states, you can specify the value it will output. You can change its state directly by clicking on the drawn switch. The other argument can be controlled in the same way.

7.3.2 Indexed

An indexed argument is a popup menu with different choices.

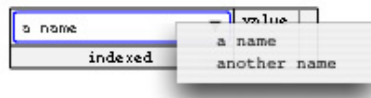


Figure 12: The indexed argument

You can specify these choices by creating, deleting and ordering the items using the settings window. For each item, any value can be given. This value will be outputted when the item is selected.

7.3.3 Slider

A slider argument is a variable value with a maximum and a minimum.



Figure 13: The slider argument

It will appear as a slider in a generic view and as a circular knob in a custom view (in the host that loads the plugin). The *Show value* button, when selected, will make the current value appear under the knob.

The *Type* attribute is a hint for the host so it can know what type of value the slider is representing. It is optional and will only be showed in a generic gui. For a slider controller gain, you should specify if the value is in linear units of decibels.

The *Mapping* attribute can be either linear or logarithmic. If you have a slider with minimum 20 and maximum 20000 (a frequency range), in linear mapping, the middle value would be around 10000, and in logarithmic mapping it would be around 450.

7.3.4 Points

An element allowing you to draw a line, with either step, linear or spline interpolation. Used to drive other elements. This argument will not appear in the generic view of exported plugins, therefore in order to be usable you will have to build a custom gui for your plugin. Internally the points are represented in a 1 by 1 box, with the lower left corner at position (0,0) and upper right corner at position (1,1). The *Points apply* element can translate and scale this box. Other elements may use this information differently.

7.3.5 Points Envelope

A variation on the *Points* element where attack, sustain and release have different sections.

7.3.6 Points Frequency

A variation on the *Points* element with markers for frequency, as used by the *Points To FFT* and *FFT generator* elements.

7.3.7 XY Pad

This a basically two sliders represented together by a box and a point. One slider takes its value by the horizontal position of the point while the other checks the vertical position.

7.3.8 Audiofile argument

Allows the user to load an audio file. The number of outputs is decided at design time. If the loaded file has too many channels, the extra ones are ignored. If it does not have enough, the extra outputs will have empty sound (silence). In case of a single channel loaded in a multiple output configuration, the channel will be given on all outputs.

7.4 Midi arguments

MIDI input is supported by the use of MIDI arguments. These arguments will receive MIDI events and change their output accordingly.

7.4.1 Midi slider

The *Midi slider* is the same as a normal slider, except it can also be controlled by midi events. To easily find which controller should be used, you can set it to learn. It will then set the controller attribute to the first controller event it receives. If you have multiple MIDI sliders, do not set them all to learn by default, since they will all switch to it simultaneously when they receive the first controller event... unless you want it like that. For example, a multiband equalizer could have some bands attached to the same controller instead of individually.

7.4.2 Midi mono note

The *Midi mono note* element outputs the last note that was pressed, with its velocity. If multiple note are pressed, it will output the last one pressed, and if the note is depressed, it will output the previous note. When no note is pressed, the specified default values are outputted. If the *Hold note* button is selected, it will not revert to the default value when the last note is depressed but rather hold it.

7.4.3 Midi multi note

Like the piecewise circuit element the *Midi multi note* is a complex element to answer a problem. If you have a sine generator and want to be able to have more than one note played simultaneously, you have to duplicate the sine generator for each additional note pressed. This is exactly what *Midi multi note* does. It contains a subcircuit which you can edit, and this circuit is duplicated on the fly for each note pressed. The output of each currently active subcircuit is summed and then outputted as one output.

7.4.4 Midi multi note env

This is a variation of *Midi multi note*, where the envelope is specified visually, through a *Points Envelope* argument.

7.5 FFT elements

Coming soon . . . Until then, you can take a look at the graphic equalizer plugin.

8 Circuit examples

8.0.1 The simplest circuit: passthrough



Figure 14: A passthrough circuit

This is the first thing you should try. The detailed steps to follow are:

1. Click on the background to select the root circuit.
2. In the settings window, specify one input and one output.
3. Drag a link between the input to the output.
4. Use the sound panel to load a sound.
5. Click play.

You should now hear the sound you just loaded.

8.0.2 Stereo to mono

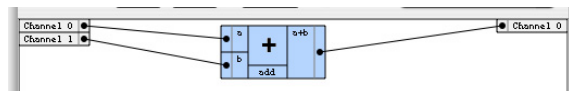


Figure 15: A stereo to mono circuit

Once you made the passthrough circuit, try this:

1. Click on the background to select the root circuit.
2. In the settings window, specify two inputs.
3. Use the *Insert element...* popup menu to insert an *Addition* element, from the *Algebraic* category.
4. Connect both inputs of the circuit to the inputs of the addition element.
5. Connect the output of the addition element to the output of the circuit.
6. Make sure the loaded sound is actually stereo. If not load another sound.
7. Click play.

You should hear the mono summation of your original file. By the way, you can do all these modifications while the circuit is playing.

8.0.3 A multiband reverb

From now on, only the general steps will be given, so you should understand the basics when trying this.

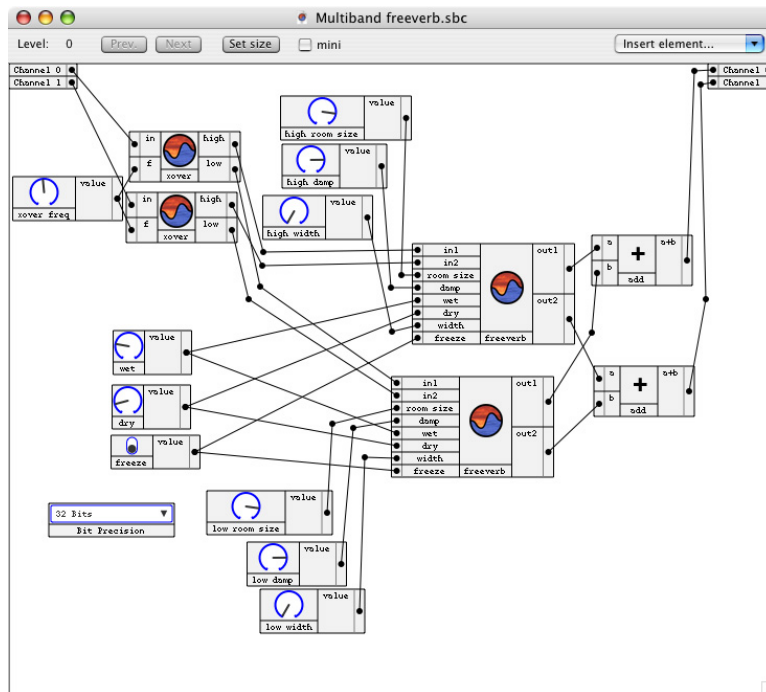


Figure 16: A multiband reverb circuit

1. Create a two input, two output root circuit.
2. Insert two crossover elements, one for each channel.
3. Use the same frequency configured slider for both.
4. Insert two freeverb elements.
5. Connect the two high outputs of both crossovers to one freeverb element.
6. Same for the low outputs.
7. Now it is up to you to decide if you want to double every settings for the freeverb, or if you want some to be in common. For example, the wet and dry settings could be shared. With one of the freeverb elements selected, read from the information window what parameters and range are expected for this element.

8. Insert two addition elements.
9. Connect the left outputs of both freeverb elements to one addition.
10. Same for the right outputs.
11. Connect the outputs of the addition elements to the output of the circuit.
12. Test it.

There is an infinite variation for this. You could add a highpass after the reverbing elements to cut the rumbles. You could have some sliders as MIDI sliders instead. You could add feedback loops to add extra reverberation. Or a valve element to add some distortion in the path. Or an atan element to slightly compress the signal. Or use an envelope follower on the input signal to control the room size of the freeverb element. As you see, it can get as complicated as you want. For very big circuits, you will want to try the mini mode.

8.0.4 A software synth

A software synth is usually splitted in two stages, the sound generation, and the sound processing. The sound generation will be done inside a MIDI multi note element. The sound processing is done after. As for the multiband reverb, the steps presented here are general.

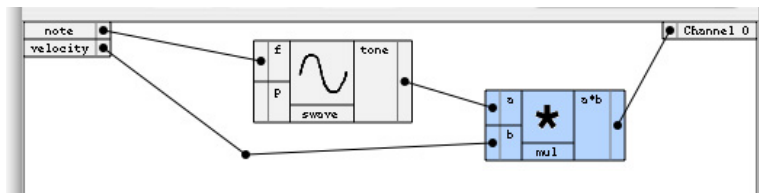


Figure 17: Inside multi midi note

1. Create a zero input, one output root circuit.
2. Insert a MIDI multi note element.
3. Give it one output.
4. Insert and connect a slider to the its attack input.
5. Insert and connect a slider to the its release input.
6. Go inside it by clicking the *Next* button while it is selected.
7. There will be two inputs available: the frequency and velocity.

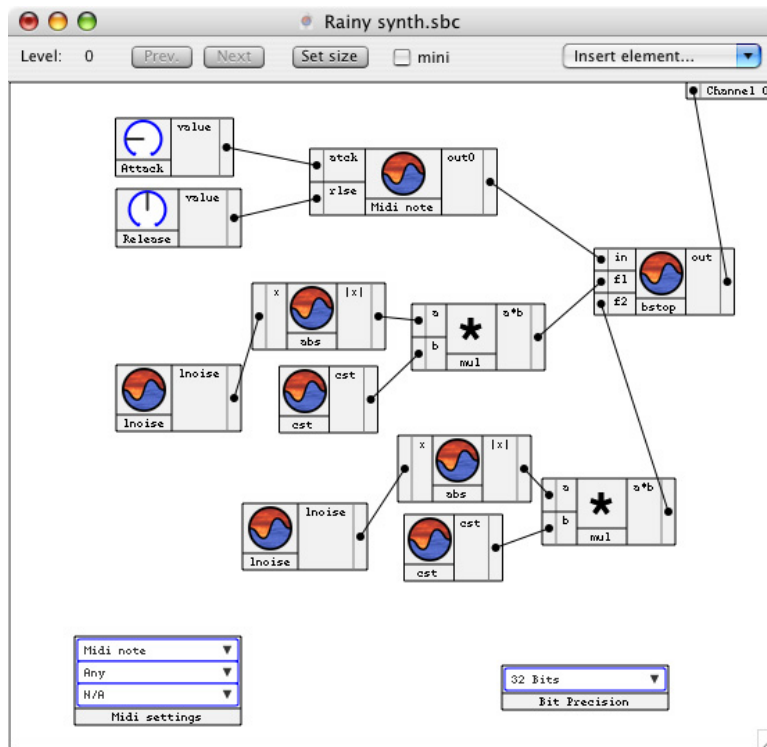


Figure 18: A rainy software synth

8. Insert a sine generator.
9. Connect the frequency input to the sine generator frequency input.
10. Insert a multiplication element.
11. Connect the velocity input and the output of the sine generator to the inputs of the multiplication element.
12. Connect the output of the multiplication element to the output of the subcircuit.
13. Click *Prev* to return to the root circuit.
14. Insert a bandstop element (we'll use it in an abnormal way to add some distortion and stereo width).
15. Insert a linear noise generator
16. Insert an absolute
17. Insert a multiplication and a constant.

18. Set the constant to 20000.
19. Connect the output of the MIDI multi note to the other input of the bandstop.
20. Connect the output of the noise generator to the absolute element.
21. Connect the output of the constant and the absolute element to the multiplication element.
22. Connect the output of the multiplication unit to one frequency input of the bandstop.
23. Repeat for the other frequency input.
24. Connect the output of the bandstop to the output of the root circuit.
25. Press a note and hear a rainy sine sound.

If you want to pass arguments to the element inside the MIDI multi note, simply give it some inputs and connect the arguments to them. You can only connect the outputs of an argument to the inputs of another argument.

9 Custom gui

A great sounding plugin is very good, but with a nice looking gui, it is even better! That is why SonicBirth allows you to make a custom gui.

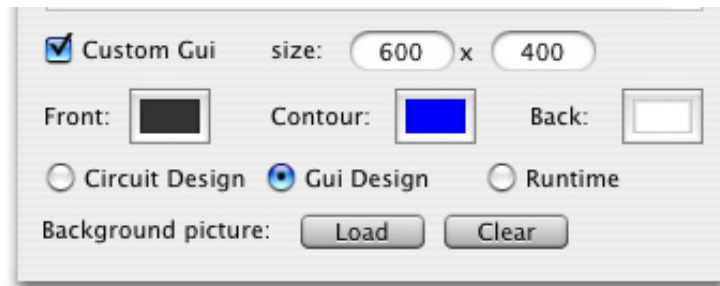


Figure 19: Custom gui settings

9.1 Designing the interface

The first step to take is activating the *Custom gui* button on the settings window of the root circuit, then switch to gui design mode. Set a suitable size. Bear in mind that not everyone has a very high resolution monitor, so keep it small if you plan on sharing it.

9.1.1 Placing the elements

The next step is to place the arguments. Simply drag & drop them where you want them to appear. Simple as that. But do not make them overlap.

9.1.2 Selecting the colors

The arguments are drawn with three selectable colors. These colors can be set below 100% opaque. In fact, you can set the opaque value to 0% for both the background and contour colors and use the background image instead.

9.1.3 Loading a background picture

The arguments will not display any labels such as minimum and maximum value, or the argument name. These are expected to be done in the background image. One trick is to take a screenshot of the window when the arguments are placed (using Command-Shift-3), then load this as the starting point in your graphical application. Save it as either jpg, gif, png or tiff format. (*Since the image file will be included as it is on disk in the circuit model file, using jpg is strongly suggested as it will keep the size smaller*)

Back in SonicBirth, click load, select your background picture, and you are done!

9.2 Testing the interface

In the runtime mode, you can test your interface as it will behave in the host. Not much to say.

10 List of elements

10.1 Algebraic

10.1.1 Addition

Outputs the sum of both inputs.

Inputs : a, b.

Outputs : $a+b$.

10.1.2 Add Many

Outputs the summation of 3 to 16 inputs.

Inputs : in0, in1, in2, in3.

Outputs : out.

10.1.3 Substraction

Outputs $(a - b)$.

Inputs : a, b.

Outputs : $a-b$.

10.1.4 Multiplication

Outputs $(a * b)$.

Inputs : a, b.

Outputs : $a*b$.

10.1.5 Division

Outputs (a / b) .

Inputs : a, b.

Outputs : a/b .

10.1.6 Constant Addition

Adds a constant number.

Inputs : i.

Outputs : o.

10.1.7 Constant Substraction

Substracts a constant number.

Inputs : i.

Outputs : o.

10.1.8 Constant Multiplication

Multiply by a constant number.

Inputs : i.

Outputs : o.

10.1.9 Constant Division

Divides by a constant number.

Inputs : i.

Outputs : o.

10.1.10 Constant Substraction Alt.

Constant number substracted by input.

Inputs : i.

Outputs : o.

10.1.11 Constant Division Alt.

Constant number divided by input.

Inputs : i.

Outputs : o.

10.1.12 Negation

Outputs -x.

Inputs : x.

Outputs : -x.

10.1.13 Inverse

Outputs 1/x.

Inputs : x.

Outputs : 1/x.

10.1.14 Absolute

Outputs $|x|$ (x if $x \geq 0$, -x if $x < 0$).

Inputs : x.

Outputs : $|x|$.

10.1.15 Absolute/Sign

Outputs $|x|$ (x if $x \geq 0$, -x if $x < 0$), and sign of x (1 for pos, -1 for neg).

Inputs : x.

Outputs : $|x|$, sign.

10.1.16 Direction

Outputs 1 if input is augmenting, -1 if it is descending, 0 otherwise.

Inputs : x.

Outputs : dir.

10.1.17 Multiply-Add

Outputs $(a*b)+c$.

Inputs : a, b, c.

Outputs : $(a*b)+c$.

10.1.18 Multiply-Sub

Outputs $(a*b)-c$.

Inputs : a, b, c.

Outputs : $(a*b)-c$.

10.1.19 Neg. Multiply-Add

Outputs $-((a*b)+c)$.

Inputs : a, b, c.

Outputs : $-((a*b)+c)$.

10.1.20 Neg. Multiply-Sub

Outputs $-((a*b)-c)$.

Inputs : a, b, c.

Outputs : $-((a*b)-c)$.

10.2 Function**10.2.1 Modulus**

Outputs $x \% y$ ($x \bmod y$).

Inputs : x, y.

Outputs : $x\%y$.

10.2.2 Ceil

Rounds to smallest integral value not less than x.

Inputs : x.

Outputs : o.

10.2.3 Floor

Rounds to largest integral value not greater than x.

Inputs : x.

Outputs : o.

10.2.4 Nearest integer

Rounds to nearest integer.

Inputs : x.

Outputs : o.

10.2.5 Exponential

Outputs $e^{\hat{x}}$.

Inputs : x.

Outputs : $e^{\hat{x}}$.

10.2.6 Natural logarithm

Outputs $\ln(x)$.

Inputs : x.

Outputs : $\ln(x)$.

10.2.7 Logarithm base 10

Outputs $\log_{10}(x)$.

Inputs : x.

Outputs : $\log_{10}(x)$.

10.2.8 Power

Outputs $x^{\hat{y}}$.

Inputs : x, y.

Outputs : $x^{\hat{y}}$.

10.2.9 Square root

Outputs \sqrt{x} .

Inputs : x.

Outputs : \sqrt{x} .

10.2.10 Reverse square root

Outputs $\text{rsqrt}(x)$.

Inputs : x.

Outputs : $\text{rsqrt}(x)$.

10.2.11 Quadratic Bezier

Outputs $(1-t)^2a + 2t(1-t)b + t^2c$, where t is clamped to 0 .. 1. See http://en.wikipedia.org/wiki/Bezier_curve.

Inputs : t, a, b, c.

Outputs : o.

10.3 Trigonometric

10.3.1 Sinus

Outputs the sinus of the input.

Inputs : x .

Outputs : $\sin x$.

10.3.2 Cosinus

Outputs the cosinus of the input.

Inputs : x .

Outputs : $\cos x$.

10.3.3 Tangent

Outputs the tangent of the input.

Inputs : x .

Outputs : $\tan x$.

10.3.4 Sinus and cosinus

Outputs both the sinus and the cosinus of the input.

Inputs : x .

Outputs : $\sin x, \cos x$.

10.3.5 Hyperbolic sinus

Outputs the hyperbolic sinus of the input.

Inputs : x .

Outputs : $\sinh x$.

10.3.6 Hyperbolic cosinus

Outputs the hyperbolic cosinus of the input.

Inputs : x .

Outputs : $\cosh x$.

10.3.7 Hyperbolic tangent

Outputs the hyperbolic tangent of the input.

Inputs : x .

Outputs : $\tanh x$.

10.3.8 Arc sinus

Outputs the arc sinus of the input.

Inputs : x .

Outputs : $\text{asin } x$.

10.3.9 Arc cosinus

Outputs the arc cosinus of the input.

Inputs : x.

Outputs : $\text{acos } x$.

10.3.10 Arc tangent

Outputs the arc tangent of the input.

Inputs : x.

Outputs : $\text{atan } x$.

10.3.11 Arc tangent (x, y)

Outputs the arc tangent of y/x .

Inputs : x, y.

Outputs : $\text{atan}(y/x)$.

10.3.12 Inverse hyperbolic sinus

Outputs the inverse hyperbolic sinus of the input.

Inputs : x.

Outputs : $\text{asinh } x$.

10.3.13 Inverse hyperbolic cosinus

Outputs the inverse hyperbolic cosinus of the input.

Inputs : x.

Outputs : $\text{acosh } x$.

10.3.14 Inverse hyperbolic tangent

Outputs the inverse hyperbolic tangent of the input.

Inputs : x.

Outputs : $\text{atanh } x$.

10.4 Arguments

10.4.1 Boolean

Boolean button with value for true and for false.

Outputs : value.

10.4.2 Slider

Basic slider with min/max.

Outputs : value.

10.4.3 Indexed

Indexed popup button with multiple values.

Outputs : value.

10.4.4 Points

Set of points making a line. Double-click to insert a point. Use left and right arrow to change interpolation type. Delete key to remove a point.

Outputs : pts.

10.4.5 Points Envelope

Set of points defining attack, loop and release. Double-click to insert a point. Use left and right arrow to change interpolation type. Delete key to remove a point.

Outputs : pts.

10.4.6 Points Frequency

Set of points suitable for fft conversion or generation. Double-click to insert a point. Use left and right arrow to change interpolation type. Delete key to remove a point.

Outputs : pts.

10.4.7 XY Pad

Two sliders combined in a XY pad.

Outputs : x, y.

10.4.8 Audio file argument

User selectable audio file with preset channel count.

Outputs : chan 0.

10.4.9 Keyboard Tap

Emits a single 1 when receiving a key stroke.

Outputs : tap.

10.5 Midi arguments

10.5.1 Midi slider

Outputs the value of the a midi parameter in a user supplied range.

Outputs : value.

10.5.2 Midi note state

Outputs the state (pressed or not) and velocity (0 to 1) for each specified note.

10.5.3 Midi mono note

Outputs the note and velocity (in the range 0 - 1) of the current presser note. You can select to hold the last note pressed.

Outputs : note, velo, numb.

10.5.4 Midi multi note

Duplicates the subcircuit on the fly for each pressed note, up to a maximum of 16. The outputs of these subcircuits are summed, then outputed. You can specify the attack and release time in milliseconds. Both are clamped to 0 .. 60000 milliseconds. You must stop/start for changes to take effect.

Inputs : atck, rlse.

10.5.5 Midi multi note envelope

Duplicates the subcircuit on the fly for each pressed note, up to a maximum of 16. The outputs of these subcircuits are summed, then outputed. You can specify the attack, loop and release time in milliseconds. All are clamped to 0 .. 60000 milliseconds. You must stop/start for changes to take effect.

Inputs : atck, loop, rlse, pts.

10.5.6 Midi XY Pad

Two sliders combined in a XY pad, midi controllable. (Y controller is always the one following the X controller).

Outputs : x, y.

10.6 Display

10.6.1 Display value

Shows the first value of the input of each audio cycle. Can be used in the plugin interface.

Inputs : in.

10.6.2 Display Osc.

An oscilloscope that can be used in the plugin interface.

Inputs : in.

10.6.3 Display Osc. (Var. res.)

An oscilloscope that can be used in the plugin interface.

Inputs : in, res.

10.6.4 Display Meter

A value meter that can be used in the plugin interface.

Inputs : in.

10.7 Analysis

10.7.1 Envelope Follower

Envelope follower with variable attack and release time, in milliseconds. The output is in positive linear units.

Inputs : in, atck, rlse.

Outputs : env.

10.7.2 Look ahead

Basic look ahead: outputs the current envelope of the input signal, and the delayed signal.

Inputs : in.

Outputs : out, env.

10.7.3 BPM Counter

BPM counter analyses its input and outputs the current bpm. Uses the last 4 events to average the bpm. Is considred an event when its input passes from under or equal to 0.5 to over 0.5.

Inputs : in.

Outputs : bpm.

10.7.4 Debug

Shows the first value of the input of each audio cycle.

Inputs : in.

10.7.5 Debug Osc.

An oscilloscope for debugging purpose.

Inputs : in.

10.8 Comparators

10.8.1 Minimum

Outputs the smallest of both inputs.

Inputs : a, b.

Outputs : min(a,b).

10.8.2 Maximum

Outputs the largest of both inputs.

Inputs : a, b.

Outputs : $\max(a,b)$.

10.8.3 Sort

Outputs the min and max of both inputs.

Inputs : a, b.

Outputs : min, max.

10.8.4 Less

If $(a < b)$ the outputs c else outputs d.

Inputs : a, b, c, d.

Outputs : o.

10.8.5 Equal

If $(a = b)$ the outputs c else outputs d.

Inputs : a, b, c, d.

Outputs : o.

10.8.6 Greater

If $(a > b)$ the outputs c else outputs d.

Inputs : a, b, c, d.

Outputs : o.

10.9 Converters

10.9.1 Msec to samples

Converts milliseconds into samples.

Inputs : ms.

Outputs : `smpl`.

10.9.2 Samples to msec

Converts samples into milliseconds.

Inputs : `smpl`.

Outputs : ms.

10.9.3 Linear to db

Converts linear values into decibels.

Inputs : `lin`.

Outputs : `db`.

10.9.4 Db to linear

Converts decibels into linear values.

Inputs : db.
Outputs : lin.

10.10 Delays

10.10.1 Delay

Delays the input signal by a variable time (maximum is user specified - clamped to 60 seconds). The dly input is in seconds.

Inputs : in, dly.
Outputs : out.

10.10.2 Delay (samples)

Delays the input signal by a variable time in samples (max: 100000).

Inputs : in, dly.
Outputs : out.

10.10.3 Delay Sinc (samples)

Delays the input signal by a variable time in samples, using 16 points Blackman windowed sinc interpolation (min delay: 8 samples, max delay: 100000 samples).

Inputs : in, dly.
Outputs : out.

10.11 Generators

10.11.1 Sine Wave

Generates a sine wave, of frequency f (hz) and phase p (0 to 2pi).

Inputs : f, p.
Outputs : tone.

10.11.2 Fast Sine Wave

Generates a sine wave, of frequency f (hz). Low cpu usage.

Inputs : f.
Outputs : tone.

10.11.3 Saw Wave

Generates a saw wave, of frequency f (hz) and phase p (0 to 2pi).

Inputs : f, p.
Outputs : sawave.

10.11.4 Triangle Wave

Generates a triangle wave, of frequency f (hz) and phase p (0 to 2π).

Inputs : f , p .

Outputs : $twave$.

10.11.5 Square wave

Generates a square wave, of frequency f (hz) and phase p (0 to 2π).

Inputs : f , p .

Outputs : $swave$.

10.11.6 Linear Noise

Generates linear noise (random values between -1 and 1).

Outputs : $lnoise$.

10.11.7 White Noise

Generates white noise (random values between -1 and 1, with gaussian distribution).

Outputs : $wnoise$.

10.11.8 Pink Noise

Generates pink noise ($1/f$ noise).

Outputs : $pnoise$.

10.11.9 Random

Generates random values between -1 and 1, changing the value at frequency f .

Inputs : f .

Outputs : rnd .

10.11.10 Random ramp

Generates random values between -1 and 1, changing the value at frequency f , interpolating between values.

Inputs : f .

Outputs : rnd .

10.11.11 FFT Generator

Generates waves using FFT based on input points..

Inputs : pts .

Outputs : snd .

10.12 Filters

10.12.1 DC Blocker

Cuts frequencies below 20 hz.

Inputs : in.

Outputs : out.

10.12.2 Parametric Eq.

Parametric Eq: f is the center frequency, g is the gain or cut in dB (clamped in the range -15 .. 15). Q should be between 0.3 and 15.

Inputs : in, f, g, Q.

Outputs : out.

10.12.3 Peak

Peaking eq, with constant Q (10) and dB gain (20 dB).

Inputs : in, f.

Outputs : out.

10.12.4 Notch

Notch with constant bandwidth (0.1 octave).

Inputs : in, f.

Outputs : out.

10.12.5 Lowpass

Lowpass filter, 12/db octave (Butterworth), with variable cutoff frequency (clamped to [1, 20000]).

Inputs : in, f.

Outputs : out.

10.12.6 Highpass

Highpass filter, 12/db octave (Butterworth), with variable cutoff frequency (clamped to [1, 20000]).

Inputs : in, f.

Outputs : out.

10.12.7 Resonant lowpass

Resonant lowpass filter, 12/db octave (Butterworth), with variable cutoff frequency (clamped to [20, 20000]). Resonance should be between $\sqrt{2}$, that is 1.414..., for no resonance, and 0.1 for max resonance.

Inputs : in, f, r.

Outputs : out.

10.12.8 Resonant highpass

Resonant highpass filter, 12/db octave (Butterworth), with variable cutoff frequency (clamped to [20, 20000]). Resonance should be between $\sqrt{2}$, that is 1.414..., for no resonance, and 0.1 for max resonance.

Inputs : in, f, r.

Outputs : out.

10.12.9 Crossover

Linkwitz-Riley 24db/octave crossover.

Inputs : in, f.

Outputs : high, low.

10.12.10 Bandstop

24db/octave bandstop.

Inputs : in, f1, f2.

Outputs : out.

10.12.11 Bandpass

24db/octave bandpass.

Inputs : in, f1, f2.

Outputs : out.

10.12.12 Lowpass (fast)

Same as lowpass, but only checks its parameter once per audio cycle.

Inputs : in, f.

Outputs : out.

10.12.13 Highpass (fast)

Same as highpass, but only checks its parameter once per audio cycle.

Inputs : in, f.

Outputs : out.

10.12.14 Resonant lowpass (fast)

Same as resonant lowpass, but only checks its parameters once per audio cycle.

Inputs : in, f, r.

Outputs : out.

10.12.15 Resonant highpass (fast)

Same as resonant highpass, but only checks its parameters once per audio cycle.

Inputs : in, f, r.

Outputs : out.

10.12.16 Crossover (fast)

Same as crossover, but only checks its parameter once per audio cycle.

Inputs : in, f.

Outputs : high, low.

10.12.17 Bandstop (fast)

Same as bandstop, but only checks its parameters once per audio cycle.

Inputs : in, f1, f2.

Outputs : out.

10.12.18 Bandpass (fast)

Same as bandpass, but only checks its parameters once per audio cycle.

Inputs : in, f1, f2.

Outputs : out.

10.12.19 Allpass

Allpass, with variable delay, clamped between 0 and 1 sec. Equivalent to a feedback comb filter followed by a feedforward comb filter. $y(t) = a*x(t) + x(t-dly) - b*y(t-dly)$

Inputs : in, a, b, dly.

Outputs : out.

10.12.20 Feedback Comb

Feedback comb, with variable delay, clamped between 0 and 1 sec. $y(t) = a*x(t) - b*y(t-dly)$

Inputs : in, a, b, dly.

Outputs : out.

10.12.21 Feedforward Comb

Feedforward comb, with variable delay, clamped between 0 and 1 sec. $y(t) = a*x(t) + b*x(t-dly)$

Inputs : in, a, b, dly.

Outputs : out.

10.12.22 Formant filter

Formant filter with variable phonemes. Choose one on v1, another on v2, and you can mix between the two using input m [0,1]. Here are the phonemes: phoneme 0: eee (beet) phoneme 1: ihh (bit) phoneme 2: eh (bet) phoneme 3: aaa (bat) phoneme 4: ahh (father) phoneme 5: aww (bought) phoneme 6: uhh (but) phoneme 7: uuu (foot) phoneme 8: ooo (boot) phoneme 9: rrr (bird) phoneme 10: lll (lull) phoneme 11: mmm (mom) phoneme 12: nnn (nun)

Inputs : i, v1, v2, m.
Outputs : o.

10.13 Feedback

10.13.1 Feedback

Allows a feedback loop, with fixed delay (0.010000 seconds).

Inputs : in.
Outputs : out.

10.14 Distortion

10.14.1 Valve

Valve distortion simulation. Level and character range is 0 to 1. Level is how much the signal is driven against the limit of the valve. Character is the hardness of the sound.

Inputs : in, lv, ch.
Outputs : out.

10.14.2 Scraper

Degrades quality of sampling rate, and bit depth (parameters in the range [0, 1], where 0 is lowest quality and 1 is original signal).

Inputs : in, sr, bd.
Outputs : out.

10.14.3 Scraper (quick)

Degrades quality of sampling rate, and bit depth (parameters in the range [0, 1], where 0 is lowest quality and 1 is original signal). Less precise but faster version.

Inputs : in, sr, bd.
Outputs : out.

10.15 Audio file

10.15.1 Audio file

Loads an audio file and outputs each channel.

10.15.2 Audio player

Audio player plays the audio from the buffer when trig is non-zero. Start and end represents the playing offset. If end is smaller than start, speed is negated. Both these values should be between 0 and 1. Playing will loop if the loop

input is non-zero. Speed give the playing speed, should be between -5 and 5. A negative speed means the buffers is played reversed.

Inputs : trig, start, end, loop, speed, buf.

Outputs : o.

10.16 FFT

10.16.1 FFT Sync

States the fft size and block positions.

Outputs : sync.

10.16.2 Forward FFT

Outputs the forward fft of the input, with variable block size (delay).

Inputs : sync, in.

Outputs : real, imag.

10.16.3 Inverse FFT

Outputs the inverse fft of the input.

Inputs : sync, real, imag.

Outputs : out.

10.16.4 Complex to Polar

Converts complex fft blocks into polar fft blocks.

Inputs : sync, real, imag.

Outputs : ampl, phas.

10.16.5 Polar to Complex

Converts polar fft blocks into complex fft blocks.

Inputs : sync, ampl, phas.

Outputs : real, imag.

10.16.6 Convolve

Convolve two fft signals (in the complex plane).

Inputs : sync, r1, i1, r2, i2.

Outputs : ro, io.

10.16.7 Points To FFT

Transforms a points function into frequency amplitudes. Range is in db.

Inputs : sync, pts, range.

Outputs : real, imag.

10.16.8 Audio file To FFT

Transforms an audio file to an fft block (considering samples up to half the fft block size).

Inputs : sync, af.

Outputs : real, imag.

10.17 Miscellaneous**10.17.1 Circuit**

A circuit inside another.

10.17.2 Constant

A constant number.

Outputs : o.

10.17.3 Equation

A mathematical equation. You can use these functions:

$\sin(x)$, $\cos(x)$, $\tan(x)$, $\text{asin}(x)$, $\text{acos}(x)$, $\text{atan}(x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, $\text{asinh}(x)$, $\text{acosh}(x)$, $\text{atanh}(x)$, $\text{atan2}(x,y)$, $\text{pow}(x,y)$, $\text{min}(x,y)$, $\text{max}(x,y)$, $\text{mod}(x,y)$, $\text{abs}(x)$, $\text{floor}(x)$, $\text{ceil}(x)$, $\text{nearint}(x)$, $\text{inv}(x)$, $\text{exp}(x)$, $\log(x)$, $\log_{10}(x)$, $\text{sqrt}(x)$, $\text{revsqrt}(x)$.

Speed: compare the execute plan for: $-i0*2*3$ and $i0*(2*3) - i0*i0*i0*i0$ and $(i0*i0)*(i0*i0)$

You can also enter an equation in the constant element ($\text{sqrt}(2)$ for example).

Inputs : i0.

Outputs : o.

10.17.4 Piecewise circuit

Depending on the value of the range input, a specific subcircuit is executed. This value is checked once per block of samples. The subcircuit which is entered when clicking next depends on the selected row in the settings window.

Inputs : range.

10.17.5 Samplerate Doubler

Audio processing inside this circuit is done at double sample rate, which can be useful (sounds better) for some type of calculations (filters).

10.17.6 Timer

Outputs the time in seconds, when run is 1, outputs 0 otherwise. Time is reset then run switches to 1.

Inputs : run.

Outputs : time.

10.17.7 Timer loop

Outputs the time in seconds, looping back to 0 when arriving to the max time specified. Loops back to 0 when arriving to the max time specified. If the max is equal or smaller than 0, then it does not loop. Time is reset then run switches to 1.

Inputs : run, max.

Outputs : time.

10.17.8 Freeverb

Jezaar's freeverb. (Room size: 0.5 to 0.999, damp: 0 to 1, wet: 0 to 3, dry: 0 to 2, width: 0 to 1, freeze: 0/no or 1/yes)

Inputs : in1, in2, room size, damp, wet, dry, width, freeze.

Outputs : out1, out2.

10.17.9 Cleaner

Removes denormals, infinities and nan (not a number). Use this if your circuit can potentially create those numbers (division by 0, $\tan(\pi/2)$, etc). The signal can optionally be clamped to [-1 .. 1].

Inputs : in.

Outputs : out.

10.17.10 Points apply

Applies the points function using x and y as origin, width and height as size.

Inputs : i, x, y, w, h, pts.

Outputs : o.

10.17.11 Bufferizer

Buffer object with three modes: silence (0), play (1), record (2). In playing mode, start and end represents the playing offset. If end is smaller than start, speed is negated. Both these values should be between 0 and 1. Playing will loop if the loop input is non-zero. Speed give the playing speed, should be between -5 and 5. A negative speed means the buffers is played reversed. In record mode, the buffer is filled from start up to its capacity.

Inputs : in, mode, start, end, loop, speed.

Outputs : out.

10.17.12 Trigger

Trigger sends either the on value or off value depending on its internal state. Its state is initialized as off. If the t (trigger) input is higher than the tt (trigger threshold) input, its state is turned on. If the r (reset) input is higher than the rt (reset threshold) input, its state is turned off. In case both events are occurring simultaneously, the state is turned on.

Inputs : tt, rt, t, r, on, off.

Outputs : o.

10.17.13 Change Slower

Slows the rate of change of the input on a specified amount of milliseconds.

Inputs : i.

Outputs : o.

10.17.14 Convolver reverb

Convolver reverb, 4096 samples latency. Max reverb length: 2457600 samples. Memory usage at max reverb length: 75.4 MBs

Inputs : in, ir.

Outputs : out.

10.17.15 AudioUnit

Wraps a third party audiounit.

Inputs : tempo, beat.

10.17.16 AudioUnit (midi)

Wraps a third party audiounit.

Inputs : tempo, beat.

10.17.17 Granulate effect

Granulator effect. Delay is max delay in seconds to create grain from (max 10 seconds). DRandomness is a value between 0 and 1 affecting the grain's delay. Ramp is a value between 0 and 100. At 0 no attack or decay is used. At 100 it gives a triangular envelope, at 50 a trapezoidal envelope. Voices is the number of simultaneous grains (max 100). Length is the duration of grains in seconds (max 1 second). LRandomness is a value between 0 and 1 affecting the grain's length. Silence is the duration of silence between grains in seconds (max 10 seconds). SRandomness is a value between 0 and 1 affecting the silence's length.

Inputs : i, delay, drndness, ramp, voices, length, lrndness, silence, srndness.

Outputs : o.

10.17.18 Granulate effect with pitch

Granulator effect with variable pitch. Delay is max delay in seconds to create grain from (max 10 seconds). DRandomness is a value between 0 and 1 affecting the grain's delay. Ramp is a value between 0 and 100. At 0 no attack or decay is used. At 100 it gives a triangular envelope, at 50 a trapezoidal envelope. Voices is the number of simultaneous grains (max 100). Length is the duration of grains in seconds (max 1 second). LRandomness is a value between 0 and 1 affecting the grain's length. Silence is the duration of silence between grains in seconds (max 10 seconds). SRandomness is a value between 0 and 1 affecting the silence's length. Pitch is the grain playing speed (0.5 to 5) - can be negative. PRandomness is a value between 0 and 1 affecting the pitch.

Inputs : i, delay, drndness, ramp, voices, length, lrndness, silence, srndness, pitch, prndness.

Outputs : o.

10.18 Internal

10.18.1 Interpolation Precision

Sets the interpolation type of the audio engine.

10.18.2 Bit Precision

Sets the bit precision of the audio engine.

10.18.3 Midi settings

MIDI settings central control.