SonicBirth Manual

Version 1.5

Antoine Missout / SonicBirth Contributors

Contents

1 License **1**

2 Introduction **2**

3 Installation **3**
3.1 The application . . . 3
3.2 The plugins . . . 3
3.3 The framework . . . 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 . . . 5
5.1.5 Root circuit . . . 6
5.1.6 Model vs plugin . . . 6
5.2 Workflow . . . 6
5.2.1 Loading a sound . . . 6
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 Multiple wires . . . 7
5.2.6 Inserting arguments . . . 8
5.2.7 Setting up the plugin attributes . . . 8
5.2.8 Exporting the plugin . . . 8
5.2.9 Plugin formats . . . 9
5.2.10 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 . . . 10
6.2 Sound . . . 10
6.3 MIDI . . . 11
6.4 Settings . . . 11
6.5 Information . . . 11
6.6 Profiler . . . 11

7 Important elements **13**
7.1 Circuits . . . 13

7.1.1 Root circuit . . . 13
7.1.2 Subcircuit . . . 15
7.1.3 Piecewise circuit . . . 15
7.2 Internal arguments . . . 15
7.2.1 Bit precision . . . 15
7.2.2 Interpolation precision . . . 16
7.2.3 Midi settings . . . 16
7.3 Arguments . . . 16
7.3.1 Boolean . . . 16
7.3.2 Indexed . . . 16
7.3.3 Slider . . . 17
7.3.4 Points . . . 17
7.3.5 Points Envelope . . . 17
7.3.6 Points Frequency . . . 17
7.3.7 XY Pad . . . 18
7.3.8 Audiofile argument . . . 18
7.3.9 Keyboard Tap . . . 18
7.4 Midi arguments . . . 18
7.4.1 Midi slider . . . 18
7.4.2 Midi mono note . . . 18
7.4.3 Midi multi note . . . 19
7.4.4 Midi multi note env . . . 19
7.5 FFT elements . . . 19

8 Circuit examples 20
8.0.1 The simplest circuit: passthrough . . . 20
8.0.2 Stereo to mono . . . 20
8.0.3 A multiband reverb . . . 21
8.0.4 A software synth . . . 22

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 Subtraction . . . 26
10.1.4 Multiplication . . . 26
10.1.5 Division . . . 26
10.1.6 Constant Addition . . . 27
10.1.7 Constant Subtraction . . . 27

10.1.8 Constant Multiplication 27
10.1.9 Constant Division 27
10.1.10 Constant Subtraction Alt. 27
10.1.11 Constant Division Alt. 27
10.1.12 Negation 28
10.1.13 Inverse 28
10.1.14 Absolute 28
10.1.15 Absolute/Sign 28
10.1.16 Direction 28
10.1.17 Multiply-Add 28
10.1.18 Multiply-Sub 29
10.1.19 Neg. Multiply-Add 29
10.1.20 Neg. Multiply-Sub 29
10.2 Function 29
10.2.1 Modulus 29
10.2.2 Ceil 29
10.2.3 Floor 30
10.2.4 Nearest integer 30
10.2.5 Exponential 30
10.2.6 Natural logarithm 30
10.2.7 Logarithm base 10 30
10.2.8 Power 30
10.2.9 Square root 31
10.2.10 Reverse square root 31
10.2.11 Quadratic Bezier 31
10.3 Trigonometric 31
10.3.1 Sinus 31
10.3.2 Cosinus 31
10.3.3 Tangent 32
10.3.4 Sinus and cosinus 32
10.3.5 Hyperbolic sinus 32
10.3.6 Hyperbolic cosinus 32
10.3.7 Hyperbolic tangent 32
10.3.8 Arc sinus 32
10.3.9 Arc cosinus 33
10.3.10 Arc tangent 33
10.3.11 Arc tangent (x, y) 33
10.3.12 Inverse hyperbolic sinus 33
10.3.13 Inverse hyperbolic cosinus 33
10.3.14 Inverse hyperbolic tangent 33
10.4 Arguments 34
10.4.1 Boolean 34
10.4.2 Slider 34
10.4.3 Indexed 34
10.4.4 Points 34
10.4.5 Points Envelope 34

10.4.6 Points Frequency . . . 35
10.4.7 XY Pad . . . 35
10.4.8 Audio file argument . . . 35
10.4.9 Keyboard Tap . . . 35
10.5 Midi arguments . . . 35
10.5.1 Midi slider . . . 35
10.5.2 Midi note state . . . 35
10.5.3 Midi mono note . . . 35
10.5.4 Midi multi note . . . 36
10.5.5 Midi multi note envelope . . . 36
10.5.6 Midi XY Pad . . . 36
10.6 Display . . . 36
10.6.1 Display value . . . 36
10.6.2 Display Osc. . . . 36
10.6.3 Display Osc. (Var. res.) . . . 37
10.6.4 Display Meter . . . 37
10.6.5 Graphic Object . . . 37
10.7 Analysis . . . 37
10.7.1 Envelope Follower . . . 37
10.7.2 Look ahead . . . 37
10.7.3 BPM Counter . . . 37
10.7.4 Debug . . . 38
10.7.5 Debug Osc. . . . 38
10.8 Comparators . . . 38
10.8.1 Minimum . . . 38
10.8.2 Maximum . . . 38
10.8.3 Sort . . . 38
10.8.4 Less . . . 39
10.8.5 Equal . . . 39
10.8.6 Greater . . . 39
10.9 Converters . . . 39
10.9.1 Msec to samples . . . 39
10.9.2 Samples to msec . . . 39
10.9.3 Linear to dB . . . 39
10.9.4 dB to linear . . . 40
10.10 Delays . . . 40
10.10.1 Delay . . . 40
10.10.2 Delay (samples) . . . 40
10.10.3 Delay Sinc (samples) . . . 40
10.11 Generators . . . 41
10.11.1 Sine Wave . . . 41
10.11.2 Fast Sine Wave . . . 41
10.11.3 Saw Wave . . . 41
10.11.4 Triangle Wave . . . 41
10.11.5 Square wave . . . 41
10.11.6 Linear Noise . . . 42

10.11.7 White Noise . . . 42
10.11.8 Pink Noise . . . 42
10.11.9 Random . . . 42
10.11.10 Random ramp . . . 43
10.11.11 Triggered Random . . . 43
10.11.12 FFT Generator . . . 43
10.12 Filters . . . 43
10.12.1 DC Blocker . . . 43
10.12.2 Parametric Eq. . . . 44
10.12.3 Peak . . . 44
10.12.4 Notch . . . 44
10.12.5 Lowpass . . . 44
10.12.6 Highpass . . . 44
10.12.7 Resonant lowpass . . . 45
10.12.8 Resonant highpass . . . 45
10.12.9 Crossover . . . 45
10.12.10 Bandstop . . . 45
10.12.11 Bandpass . . . 45
10.12.12 Lowpass (fast) . . . 46
10.12.13 Highpass (fast) . . . 46
10.12.14 Resonant lowpass (fast) . . . 46
10.12.15 Resonant highpass (fast) . . . 46
10.12.16 Crossover (fast) . . . 46
10.12.17 Bandstop (fast) . . . 46
10.12.18 Bandpass (fast) . . . 47
10.12.19 Allpass . . . 47
10.12.20 Feedback Comb . . . 47
10.12.21 Feedforward Comb . . . 47
10.12.22 Formant filter . . . 47
10.12.23 Parametric Eq. (lin. phase) . . . 48
10.12.24 Peak (lin. phase) . . . 48
10.12.25 Notch (lin. phase) . . . 48
10.12.26 Lowpass (lin. phase) . . . 49
10.12.27 Highpass (lin. phase) . . . 49
10.12.28 Highpass (lin. phase, sub-bass) . . . 49
10.12.29 Bandpass (lin. phase) . . . 49
10.12.30 Bandstop (lin. phase) . . . 50
10.13 Feedback . . . 50
10.13.1 Feedback . . . 50
10.13.2 Feedback Zero . . . 50
10.14 Distortion . . . 50
10.14.1 Valve . . . 50
10.14.2 Scraper . . . 51
10.14.3 Scraper (quick) . . . 51
10.15 Audio file . . . 51
10.15.1 Audio file . . . 51

10.15.2 Audio player 51
10.16 FFT 51
10.16.1 FFT Sync 51
10.16.2 Forward FFT 52
10.16.3 Inverse FFT 52
10.16.4 Complex to Polar 52
10.16.5 Polar to Complex 52
10.16.6 Convolve 52
10.16.7 Points To FFT 52
10.16.8 Audio file To FFT 53
10.17 Miscellaneous 53
10.17.1 Circuit 53
10.17.2 Constant 53
10.17.3 Equation 53
10.17.4 Piecewise circuit 54
10.17.5 Samplerate Doubler 54
10.17.6 Timer 54
10.17.7 Timer loop 55
10.17.8 Freeverb 55
10.17.9 Cleaner 55
10.17.10 Points apply 55
10.17.11 Bufferizer 55
10.17.12 Trigger 56
10.17.13 Flip Flop 56
10.17.14 Change Slower 56
10.17.15 Change Slower 2 56
10.17.16 Moving Average 57
10.17.17 XOver 57
10.17.18 Ambisonic encoder 57
10.17.19 Convolving reverb 57
10.17.20 AudioUnit 58
10.17.21 AudioUnit (midi) 58
10.17.22 Granulate effect 58
10.17.23 Granulate effect with pitch 58
10.17.24 Visible Comment 58
10.18 Internal 59
10.18.1 Interpolation Precision 59
10.18.2 Bit Precision 59
10.18.3 Midi settings 59

1 License

Copyright ©2005–2026, Antoine Missout and Daniel Courville.

SonicBirth is free software: it may be used and redistributed free of charge, but its source code is not publicly available. All rights reserved.

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

Permission is granted to redistribute plugins exported with SonicBirth. Keep in mind that end users will need to have the SonicBirth framework installed to use exported plugins. Direct them to the SonicBirth project page to download the installer.

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 an application dedicated to the creation of audio plugins for macOS. It exports plugins in three formats: AudioUnit (AU), VST2, and VST3. 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, VST2, or VST3 plugin.

SonicBirth 1.5 requires macOS 12.4 or later and runs natively on both Apple Silicon (arm64) and Intel (x86_64) Macs.

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 application

The SonicBirth application is the design environment where you build and test your circuits before exporting them as plugins.

3.2 The plugins

Installing the prebuilt plugins is optional. You are encouraged to install them – they can serve as starting points and examples.

3.3 The framework

The SonicBirth framework is the shared runtime engine used by all exported plugins. It must be installed on any machine where exported plugins will be used. The SonicBirth installer takes care of this automatically.

4 Quick tips

Some tips *en vrac* :

- SonicBirth can export plugins in AudioUnit, VST2, and VST3 formats. Use the *File* menu to export; hold the Option key to choose an alternate format.
- The points box: left/right arrows to change interpolation type, double-click to create a point, press delete to remove it.
- Double-click a wire anchor to delete it.
- Drag a wire away from its element input to delete it. If several wires are connected to that input, the most recently added one is picked up.
- Several wires can be connected to the same input: their values are added together.
- Hold the Command key while dragging a wire to connect several outputs to several inputs at once.
- 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 to a new format or version.
- 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).
- The Profiler window lets you identify which elements are using the most CPU time. Use the A/B snapshot feature to compare two circuit configurations.
- The Visible Comment element can be placed anywhere in your circuit to leave notes for yourself or collaborators.
- The Equation element supports conditional select functions: `iflt(a,b,x,y)` returns `x` if `a < b`, else `y`. On Apple Silicon, the equation engine uses NEON SIMD to process four samples at once.

5 Overview

The main purpose of SonicBirth is to allow you to create your own audio plugins. 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

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.

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.

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.

The circuit hierarchy

5.1.5 Root circuit

The root circuit is simply the circuit at the base of your 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, company, 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` (AU), `.vst` (VST2), or `.vst3` (VST3) 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 *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

A simple feedback loop with delay and amount controls

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. A *Feedback Zero* element is also available for ultra-low-latency feedback (approximately one sample of delay).

5.2.5 Multiple wires

Multiple wires on one input. An input can receive more than one wire: SonicBirth simply adds the values of all the wires connected to it. Releasing a new wire on an input that is already connected adds a wire instead of replacing the existing one. To replace a connection, select the old wire and delete it, or drag it away from the input. This applies to number connections, including the outputs of a circuit. Points, audio buffer and FFT sync connections cannot be added together: a new wire replaces the previous one. An input with a single wire costs nothing extra; the sum is only calculated for inputs that have two wires or more.

Connecting several wires at once. Hold the Command key while you drag a wire, either when you click or when you release the mouse. SonicBirth connects the output you started from to the input you release on, then the next output to the next input, and so on, until either element runs out of outputs or inputs. For example, a Command-drag from output `acn0` of an Ambisonic

encoder to input 0 of the root circuit connects all 64 ambisonic channels in one gesture. This also works when you start dragging from an input. Command-clicking a connector does not change the selection.

5.2.6 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.

5.2.7 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 AU 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. For VST2 and VST3, unique identifiers are generated automatically from the bundle identifier.

  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 in an AU host. The subtype is unavoidable, it comes from the AudioUnit standard.

5.2.8 Exporting the plugin

Ready to try your plugin in your usual audio host? Use the *File* menu, click export, and choose a destination. SonicBirth exports AudioUnit plugins by default. Hold the Option key while clicking *Export* to choose VST2 or VST3 format instead. You can also use *Batch Export* to export your entire library to a new format or location at once.

5.2.9 Plugin formats

- **AudioUnit (AU)**: The native macOS plugin format. Offers the tightest integration with Logic Pro, GarageBand, and other Apple-ecosystem hosts.
- **VST2**: Widely supported by third-party DAWs (Ableton Live, Reaper, Cubase, etc.). Supports both 32-bit and 64-bit audio processing.
- **VST3**: The modern Steinberg standard. Supports 32-bit and 64-bit processing, precise parameter automation, and side-chain detection.

5.2.10 Sharing your creation

Made something great? Would like to share it? Please do so! You can distribute either the circuit model file (which has a file extension `.sbc`) or the actual exported plugin file. Keep in mind that end users must have the SonicBirth framework installed on their machine for exported plugins to load. Direct them to the SonicBirth project page to download the installer, which installs the framework.

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

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

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

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.

6.4 Settings

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

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.

6.6 Profiler

The profiler window shows how much CPU time each element consumes during audio processing. It displays a hierarchy of elements with their individual percentage of total processing time and average time in microseconds. Sub-circuits can be expanded to reveal their contents.

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

- **Start/Stop**: Begin or end profiling. Starting the profiler also starts the sound server if it is not already running.
- **A/B Snapshot**: Capture the current profile as snapshot A or B, then compare the two. The delta column shows the change in CPU usage between the two snapshots. This is useful for measuring the effect of circuit changes.
- **Auto-highlight**: When enabled, selecting an element in the profiler also selects it in the circuit view.

7 Important elements

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

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 or developer name that will appear in the plugin host's plugin list and in the generic GUI. It is used for all three export formats (AU, VST2, VST3).
- Version: The plugin version string (e.g. "1.0.0"). Used in the plugin's metadata.
- Description: Here you should give a short description of your plugin, though most hosts ignore this information.
- Subtype: This should be a unique four characters identifier for AudioUnit export. 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. VST2 and VST3 identifiers are generated automatically.
- 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 a plugin 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 plugins 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. It 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 (32-bit float or 64-bit double). In exported plugins, this setting is fixed at export time.

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. Every argument has a 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.

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.

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.

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 or 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 is 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.

7.3.9 Keyboard Tap

Emits a single 1 when receiving a key stroke. Can be used to trigger events or step through states from within the circuit.

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.

7.4.2 Midi mono note

The *Midi mono note* element outputs the last note that was pressed, with its velocity. If multiple notes are pressed, it will output the last one pressed, and if the note is released, 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 released 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

FFT-based processing elements let you work in the frequency domain. A typical FFT chain starts with an *FFT Sync* element that defines the block size, followed by a *Forward FFT* to decompose the signal, your processing elements, and finally an *Inverse FFT* to reconstruct the time-domain signal. See the FFT section in the element list for full details.

8 Circuit examples

8.0.1 The simplest circuit: passthrough

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

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.

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 split 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.

Inside multi midi note

A rainy software synth

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 its attack input.
5. Insert and connect a slider to 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.
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.

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 64 inputs.

Inputs : in0, in1, in2, in3.

Outputs : out.

10.1.3 Subtraction

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 Subtraction

Subtracts 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 Subtraction Alt.

Constant number subtracted 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 mod 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 eˆx.

Inputs : x.

Outputs : eˆx.

10.2.6 Natural logarithm

Outputs ln(x).

Inputs : x.

Outputs : ln(x).

10.2.7 Logarithm base 10

Outputs log10(x).

Inputs : x.

Outputs : log10(x).

10.2.8 Power

Outputs xˆy.

Inputs : x, y.

Outputs : xˆy.

10.2.9 Square root

Outputs sqrt(x).

Inputs : x.

Outputs : sqrt(x).

10.2.10 Reverse square root

Outputs rsqrt(x) (1/sqrt(x)).

Inputs : x.

Outputs : rsqrt(x).

10.2.11 Quadratic Bezier

Outputs $(1-t)^2 \cdot a + 2 \cdot t \cdot (1-t) \cdot b + t^2 \cdot c$, where t is clamped to 0..1.

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 : asin x.

10.3.9 Arc cosinus

Outputs the arc cosinus of the input.

Inputs : x.

Outputs : acos x.

10.3.10 Arc tangent

Outputs the arc tangent of the input.

Inputs : x.

Outputs : atan x.

10.3.11 Arc tangent (x, y)

Outputs the arc tangent of y/x.

Inputs : x, y.

Outputs : atan(y/x).

10.3.12 Inverse hyperbolic sinus

Outputs the inverse hyperbolic sinus of the input.

Inputs : x.

Outputs : asinh x.

10.3.13 Inverse hyperbolic cosinus

Outputs the inverse hyperbolic cosinus of the input.

Inputs : x.

Outputs : acosh x.

10.3.14 Inverse hyperbolic tangent

Outputs the inverse hyperbolic tangent of the input.

Inputs : x.

Outputs : 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 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 to 1) of the current pressed 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 outputted. 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 outputted. 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 with variable resolution 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.

10.6.5 Graphic Object

A purely graphical element that can be placed in the custom GUI. It produces no audio output and has no inputs. Useful for decorative shapes and visual structure in a custom interface.

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. An event is considered 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) then outputs c else outputs d.

Inputs : a, b, c, d.

Outputs : o.

10.8.5 Equal

If (a = b) then outputs c else outputs d.

Inputs : a, b, c, d.

Outputs : o.

10.8.6 Greater

If (a $>$ b) then 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: 200000).

Inputs : in, dly.

Outputs : out.

10.10.3 Delay Sinc (samples)

Delays the input signal by a variable time in samples, using 16-point 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 2π). The algorithm setting (checked once per audio buffer) selects the computation method: 0 = standard sin() (default, maximum precision); 1 = Bhaskara I approximation (approximately twice as fast, max error 0.16%, -56 dB).

Inputs : f, p.

Outputs : tone.

10.11.2 Fast Sine Wave

Generates a sine wave, of frequency f (Hz). Low CPU usage via a recursive oscillator.

Inputs : f.

Outputs : tone.

10.11.3 Saw Wave

Generates a band-limited sawtooth wave, of frequency f (Hz) and phase p (0 to 2π). Uses 4th-order PolyBLEP anti-aliasing with 2x oversampling.

Inputs : f, p.

Outputs : sawave.

10.11.4 Triangle Wave

Generates a band-limited triangle wave, of frequency f (Hz) and phase p (0 to 2π). Uses 4th-order PolyBLAMP anti-aliasing with 2x oversampling.

Inputs : f, p.

Outputs : twave.

10.11.5 Square wave

Generates a band-limited square wave, of frequency f (Hz) and phase p (0 to 2π). Uses 4th-order PolyBLEP anti-aliasing with 2x oversampling.

Inputs : f, p.

Outputs : swave.

10.11.6 Linear Noise

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

Outputs : lnoise.

10.11.7 White Noise

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

Outputs : wnoise.

10.11.8 Pink Noise

Generates pink noise (1/f noise).

Outputs : pnoise.

10.11.9 Random

Generates random values at frequency f. Output changes instantly at each trigger.

Three generation modes (selected by the type input, checked once per audio buffer):

- 0 – Random: picks any value in [-1, 1]
- 1 – Drunk: random walk bounded to [-1, 1]; range scales the maximum step size
- 2 – Markov: transitions between evenly-spaced states; range scales the maximum jump distance, bias shifts jumps toward lower (<0) or higher (>0) states

Inputs : f (trigger frequency in Hz), type, range (step size 0–1), bias (directional tendency -1 to +1, Markov only), states (number of discrete states 2–25, Markov only).

Outputs : rnd (value in [-1, 1]), state (current Markov state index, 0 otherwise).

10.11.10 Random ramp

Like Random, but the output interpolates linearly between successive values rather than changing instantly.

Inputs : f, type, range, bias, states.

Outputs : rnd, state.

10.11.11 Triggered Random

Triggered random generator. On each rising edge of the trigger input, generates a new value according to the selected mode. Output holds until the next trigger.

Mode 0 (Coin): outputs 0 or 1 with equal probability. Mode 1 (Random): picks any value in $[-1, 1]$. Mode 2 (Drunk): random walk bounded to $[-1, 1]$ with reflection; `range` scales the maximum step size. Mode 3 (Markov): transitions between evenly-spaced states; `range` scales the maximum jump distance, `bias` shifts jumps toward lower (< 0) or higher (> 0) states.

`type`, `range`, `bias`, and `states` are read once per audio buffer.

Inputs : t (trigger), type, range, bias, states.

Outputs : out, state.

10.11.12 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 to 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 $\sqrt{2}$ (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 $\sqrt{2}$ (1.414...) for no resonance, and 0.1 for max resonance.

Inputs : in, f, r.

Outputs : out.

10.12.9 Crossover

Linkwitz-Riley 24 dB/octave crossover.

Inputs : in, f.

Outputs : high, low.

10.12.10 Bandstop

24 dB/octave bandstop.

Inputs : in, f1, f2.

Outputs : out.

10.12.11 Bandpass

24 dB/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:

0: eee (beet)	7: uuu (foot)
1: ihh (bit)	8: ooo (boot)
2: ehh (bet)	9: rrr (bird)
3: aaa (bat)	10: lll (lull)
4: ahh (father)	11: mmm (mom)
5: aww (bought)	12: nnn (nun)
6: uhh (but)	

Inputs : i, v1, v2, m.

Outputs : o.

10.12.23 Parametric Eq. (lin. phase)

Linear-phase version of the Parametric Eq.: same magnitude response (f center frequency, g gain or cut in dB, Q between 0.3 and 15) but with no phase distortion. Implemented as a symmetric FIR, so it adds a constant latency of 4096 samples (about 93 ms at 44100 Hz); set the circuit latency to compensate.

Inputs : in, f, g, Q.

Outputs : out.

10.12.24 Peak (lin. phase)

Linear-phase peaking EQ, constant Q (10) and gain (20 dB); same magnitude as Peak, with no phase distortion. Constant latency of 4096 samples (about 93 ms at 44100 Hz).

Inputs : in, f.

Outputs : out.

10.12.25 Notch (lin. phase)

Linear-phase notch, constant bandwidth (0.1 octave); same magnitude as Notch, with no phase distortion. Constant latency of 4096 samples (about 93 ms at 44100 Hz).

Inputs : in, f.

Outputs : out.

10.12.26 Lowpass (lin. phase)

Linear-phase lowpass (Butterworth), with no phase distortion. The order input sets the slope in 12 dB/octave steps: order 0 is 12 dB/octave, up to order 7 which is 96 dB/octave. Constant latency of 4096 samples (about 93 ms at 44100 Hz), independent of the order.

Inputs : in, f, order.

Outputs : out.

10.12.27 Highpass (lin. phase)

Linear-phase highpass (Butterworth), with no phase distortion. The order input sets the slope in 12 dB/octave steps: order 0 is 12 dB/octave, up to order 7 which is 96 dB/octave. Constant latency of 4096 samples (about 93 ms at 44100 Hz), independent of the order. For corner frequencies below about 40 Hz, use the sub-bass variant.

Inputs : in, f, order.

Outputs : out.

10.12.28 Highpass (lin. phase, sub-bass)

Linear-phase highpass with a long (65537-tap) kernel for very low corner frequencies, down to about 20 Hz, where the standard linear-phase highpass runs out of resolution. Same f and order controls. Constant latency of 65536 samples (about 1.49 s at 44100 Hz, about 1.37 s at 48000 Hz); set the circuit latency to 65536 to compensate.

Inputs : in, f, order.

Outputs : out.

10.12.29 Bandpass (lin. phase)

Linear-phase bandpass between f1 and f2, with no phase distortion. The order input sets the edge slope in 12 dB/octave steps (order 0 is 12 dB/octave, up to order 7 which is 96 dB/octave). Constant latency of 4096 samples (about 93 ms at 44100 Hz).

Inputs : in, f1, f2, order.

Outputs : out.

10.12.30 Bandstop (lin. phase)

Linear-phase bandstop (rejects between f1 and f2), the exact complement of the linear-phase bandpass. The order input sets the edge slope in 12 dB/octave steps (order 0 is 12 dB/octave, up to order 7 which is 96 dB/octave). Constant latency of 4096 samples (about 93 ms at 44100 Hz).

Inputs : in, f1, f2, order.

Outputs : out.

10.13 Feedback

10.13.1 Feedback

Allows a feedback loop, with a fixed delay of approximately 10 milliseconds. Note that an additional, intrinsic delay is induced by the audio buffer size of the host and is unavoidable.

Inputs : in.

Outputs : out.

10.13.2 Feedback Zero

Allows a feedback loop with minimal latency (approximately 1 sample delay, around 0.02 ms at 44100 Hz). Useful for tight feedback loops where the standard 10 ms delay is too long. Use with care: no filtering between output and input can cause instability. Note that an additional, intrinsic delay is induced by the audio buffer size of the host and is unavoidable.

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 represent 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 gives the playing speed, should be between -5 and 5. A negative speed means the buffer 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. Connect the sync output to the sync input of all FFT elements in the same processing chain.

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. You can also enter a mathematical expression (e.g. `sqrt(2)`).

Outputs : o.

10.17.3 Equation

A mathematical equation element with up to 8 inputs (`i0` through `i7`) and one output (`o`). Supports the following functions:

Math: `sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), sinh(x), cosh(x), tanh(x), asinh(x), acosh(x), atanh(x), atan2(x,y), pow(x,y), min(x,y), max(x,y), mod(x,y), abs(x), floor(x), ceil(x), nearint(x), inv(x), exp(x), log(x), log10(x), sqrt(x), revsqrt(x), sign(x).`

Comparisons (return 1 or 0): `lt(a,b), le(a,b), gt(a,b), ge(a,b), eq(a,b), ne(a,b).`

Conditional select (return `x` if condition is true, `y` if false): `iflt(a,b,x,y), ifle(a,b,x,y), ifgt(a,b,x,y), ifge(a,b,x,y), ifeq(a,b,x,y), ifne(a,b,x,y).`

Note: When mixing signal inputs and constants, argument `a` must be a signal, not a constant. For example, `ifeq(i0, 1, i1, 0)` is valid; `ifeq(1, i0, i1, 0)` is not.

Clamp: `clamp(x, lo, hi)` – clamps `x` to the range [`lo`, `hi`].

Logical (return 1 or 0): `and(a,b)`, `or(a,b)`, `not(a)`.

Speed tip: compare the execute plan for:

- `i0*2*3` and `i0*(2*3)`
- `i0*i0*i0*i0` and `(i0*i0)*(i0*i0)`

On Apple Silicon (arm64), the equation engine is NEON SIMD accelerated, processing four samples per clock cycle for most operations.

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 types of calculations (filters).

10.17.6 Timer

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

Inputs : run.

Outputs : time.

10.17.7 Timer loop

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

Inputs : run, max.

Outputs : time.

10.17.8 Freeverb

Jezar'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(π/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 represent 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 gives the playing speed, should be between -5 and 5. A negative speed means the buffer 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 occur simultaneously, the state is turned on.

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

Outputs : o.

10.17.13 Flip Flop

Outputs input 'a' by default. Every time 't' crosses above 0.5, the output switches to the other input.

Inputs : t (trigger), a (first value), b (second value).

Outputs : o.

10.17.14 Change Slower

Slows the rate of change of the input over an amount of milliseconds specified in the settings panel.

Inputs : i.

Outputs : o.

10.17.15 Change Slower 2

Slows the rate of change with selectable easing. The type input (checked once per audio buffer) selects from 17 easing curves: 0=Sine (default), 1=Smootherstep, 2=Smoothstep, 3=Cubic, 4=Quadratic, 5=Quartic, 6=Quintic, 7=Circular, 8=Linear, 9=Elastic+Cubic, 10=Elastic+Smoothstep, 11=Elastic+Smootherstep, 12=In (accelerate), 13=Out (decelerate), 14=Fast Start, 15=Slow Start, 16=Exponential (asymptotic).

Inputs : i (value to smooth), time (transition duration in ms), type (easing type 0–16).

Outputs : o.

10.17.16 Moving Average

Computes the moving average of input 'i' over a time window specified by 'time' in milliseconds.

Inputs : i, time.

Outputs : o.

10.17.17 XOver

Switches from one input to the other at constant dB speed. sel is 0 or negative to select i1, positive to select i2. time1 is the time in seconds to fade from 0 to -140 dB. time2 is the time to switch direction.

Inputs : i1, i2, sel, time1, time2.

Outputs : o.

10.17.18 Ambisonic encoder

Encodes a mono signal into ambisonics, up to 7th order. The output format is AmbiX: ACN channel ordering, SN3D normalization, without Condon-Shortley phase. az is the azimuth in degrees (0 is front, 90 is left), el is the elevation in degrees (90 is up). order is a value from 0 to 7, rounded to the nearest integer and read at the start of each audio buffer. Only the first $(order + 1)^2$ channels are calculated, and the other outputs are silent: 4 channels at order 1, 16 at order 3, 64 at order 7. When the direction does not change during a buffer, the encoding gains are calculated only once for that buffer.

Inputs : in, az, el, order.

Outputs : acn0 to acn63.

10.17.19 Convolving reverb

Convolving reverb, 4096 samples latency. Max reverb length: 2457600 samples. Memory usage at max reverb length: 75.4 MB.

Inputs : in, ir.

Outputs : out.

10.17.20 AudioUnit

Wraps a third-party AudioUnit plugin.

Inputs : tempo, beat.

10.17.21 AudioUnit (midi)

Wraps a third-party AudioUnit plugin with MIDI support.

Inputs : tempo, beat.

10.17.22 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.

10.17.23 Granulate effect with pitch

Granulator effect with variable pitch. Same parameters as Granulate effect, plus: 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.

10.17.24 Visible Comment

A text annotation that can be placed anywhere in the circuit view. Useful for documenting your circuit, labeling sections, or leaving notes for collaborators.

The text is visible only inside SonicBirth and is not included in the exported plugin.

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 (32-bit float or 64-bit double).

10.18.3 Midi settings

MIDI settings central control.