

USER MANUAL

ViMiC - Virtual Microphone Control for Jamoma and Max/MSP



IMPLEMENTATION: Jonas Braasch, Tristan Matthews, Nils Peters

CONCEPT: Jonas Braasch, Nils Peters

DOCUMENTATION: Nils Peters

Draft Version, September 21, 2013

Contents

1	Introduction	2
1.1	Disclaimer	2
1.2	Acknowledgment	2
1.3	Requirements	2
1.4	Installation	2
1.5	Principles	2
1.5.1	Source - Microphone Relation	3
1.5.2	Late Reverb	5
1.6	Moving Sources	5
1.6.1	Rendering with Doppler effect	5
1.6.2	Rendering without Doppler effect	6
1.7	Getting started	6
2	The Jamoma Modules	9
2.1	The module jmod.sur.vimic~	9
2.1.1	Front-end	9
2.1.2	Inspector Interface	11
2.2	The module jmod.sur.vimic8~	20
2.2.1	Description message	21
2.3	The module jmod.sur.vimic8poly~	21
2.4	Rendermodes	22
2.4.1	Comparison & CPU load	22
2.5	The module jmod.sur.reverb~	23
2.5.1	Introduction	23
2.5.2	Parameter & Controlling	24
2.5.3	CPU-load	27
3	Appendix	28
3.1	Jamoma's MIDI to gain conversion	28
3.2	Coordinate System	29
3.3	Known Issues	30
4	References	31
4.1	jmod.sur.vimic~	32
4.2	jmod.sur.reverb~	37

Chapter 1

Introduction

1.1 Disclaimer

THE ViMiC APPLICATION IS A FREE SOFTWARE, LICENSED UNDER THE “NEW BSD” LICENSE¹. PLEASE CONSULT THE PROVIDED COPYING-FILE FOR DETAILS ABOUT WARRANTY AND LICENSE.

1.2 Acknowledgment

This work was funded by the Canadian Natural Sciences and Engineering Research Council (NSERC) and the Centre for Interdisciplinary Research in Music, Media and Technology (CIRMMT).

1.3 Requirements

- Max/MSP 6 or newer
- MacOS 10.5.8 or newer²

1.4 Installation

ViMiC for Max³ is available in the github repository: <http://github.com/Nilson/ViMiC-and-friends>

A tutorial to access and work with the Jamoma github repository is available at:
http://redmine.jamoma.org/wiki/jamoma/Working_with_GIT

To verify correct installation, load the help-patch *jmod.sur.vimic~.maxhelp*, located at */spatialization/sur.vimic~/.*. There should not be any errors in the max window.

1.5 Principles

ViMiC is a computer-generated virtual environment, where gains and delays between a virtual sound source and virtual microphones are calculated according to their distances, and the axis orientations of their microphone directivity patterns. Besides the direct sound component, a virtual microphone signal can also include early reflections and a late reverb tail, both dependent upon the sound absorbing and reflecting properties of the virtual surfaces.

¹http://en.wikipedia.org/wiki/BSD_licenses

²The external is also available for Windows OS (jcom.vimic~.mxe) but has not been extensively tested.

³ViMiC for Max 4.6.3 is available in the Jamoma Max 4 maintenance repository: https://jamoma.svn.sourceforge.net/svnroot/jamoma/branches/maintenance_max4_support

1.5.1 Source - Microphone Relation

Sound sources and microphones can be placed and moved in 3D as desired. Figure 1.2 shows an example of one sound source recorded with three virtual microphones. A virtual microphone has five degrees of freedom: (X, Y, Z, yaw, pitch) and a sound source has four: (X, Y, Z, yaw). The propagation path between a sound source and each microphone is accordingly simulated. Depending on the speed-of-sound c and the distance d_i between a virtual sound source and the i -th microphone, time-of-arrival and attenuation due to distance are estimated. This attenuation function, seen in Eq. 1.1 can be greatly modified by changing the exponent q . Thus, the effect of distance attenuation can be boosted or softened. The minimum distance to a microphone is limited to 1 meter in order to avoid high amplification.

$$g_i = \frac{1}{d_i^q} \quad d \geq 1 \quad (1.1)$$

As an alternative to this inverse proportional decrease, a second distance function is implemented. Here the decrease in dB per unit can be controlled by the k :

$$g_i = 10^{\frac{-k}{20} \cdot d_i} \quad (1.2)$$

Further attenuation happens through the chosen microphone characteristic and source directivity (see Fig. 1.3).

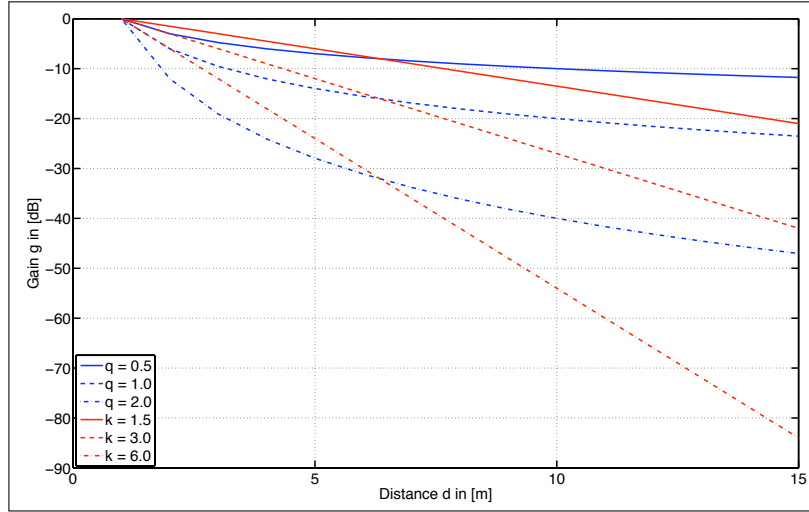


Figure 1.1: Comparison of the distance attenuation functions with different settings: “inverse proportional decrease” model (blue lines, Equation 1.1) and “exponential decrease” model (red lines, Equation 1.2)

For all common microphone characteristics, the directivity for a certain angle of incidence δ can be imitated by calculating Eq. 1.3 and applying a set of microphone coefficients from Table 1.1. By increasing the exponent w to a value greater than 1 will produce an artificially sharper directivity pattern. Unlike actual microphone characteristics, which vary with frequency, microphones in ViMiC are designed to apply the concept of microphone directivity without simulating undesirable frequency dependencies.

$$\Gamma = (a + (1 - a) \cdot \cos \delta)^w \quad 0 \leq a \leq 1 \quad (1.3)$$

Source directivity is known to contribute to immersion and presence. Therefore ViMiC is also equipped with a source directivity model. For the sake of simplicity, in a graphical control window, the source directivity can be modeled through a frequency independent gain factor for each radiation angle to a 1° accuracy.

Room model

ViMiC contains a shoe-box room model to generate time-accurate early reflections that increase the illusion of this virtual space and envelopment as described in the literature [10]. Early reflections are strong auditory cues in encoding the sound source distance. According to virtual room size and position of the microphones, adequate

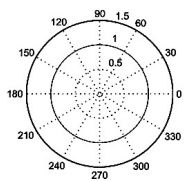
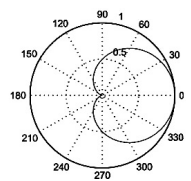
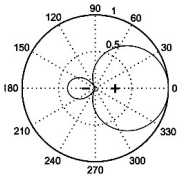
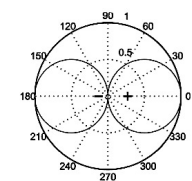
	Omnidirectional	Subcardioid	Cardioid	Supercardioid	Hypercardioid	Figure-8
a	1.0	0.7	0.5	0.33	0.3	0.0
$1 - a$	0.0	0.3	0.5	0.66	0.7	1.0
w	1	1	1	1	1	1
						

Table 1.1: Examples of common microphone directivity pattern and associated coefficients for calculating Equation 1.3. The microphone directivity can be continuously changed in real-time.

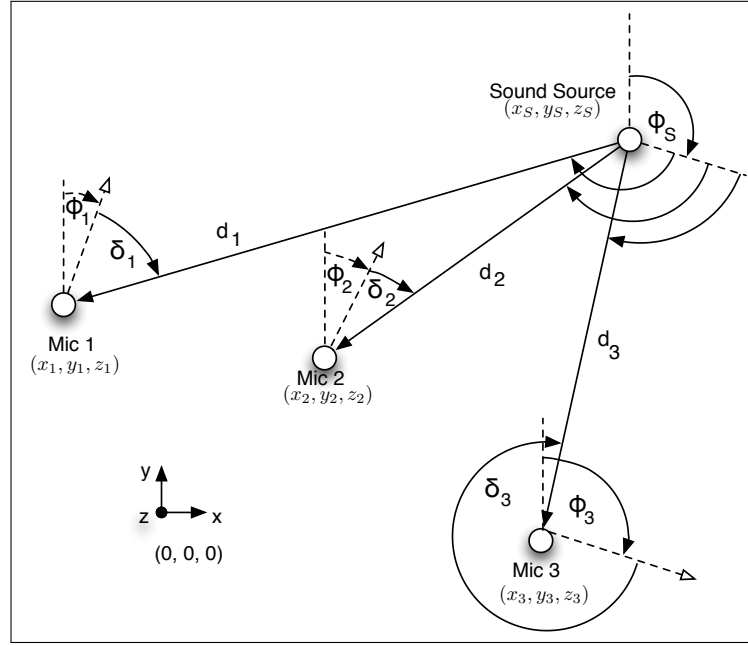


Figure 1.2: Geometric example

early reflections are rendered in 3D through the well-known image method [1]. For a general discussion of the image method, see, e.g. [5] and [3] for use in a rectangular room. Each image source is rendered according to the time of arrival, the distance attenuation, microphone characteristic and source directivity, as described in section 1.5.1. Virtual room dimensions (height, length, width) modified in real-time alter the reflection pattern accordingly. The spectral influence of the wall properties are simulated through high-mid-low shelf-filters. Because larger propagation paths increase the audible effect of air absorption, early reflections in ViMiC are additionally filtered through a 2nd-order Butterworth lowpass filter with adjustable cut-off frequency.

Also, early reflections must be discretely rendered for each microphone, as propagation paths differ. For eight virtual microphones, 56 paths are rendered if the 1st-order reflections are considered (8 microphones · [6 early reflections + 1 direct sound path]). Although time delays are efficiently implemented through a shared multi-tap delay line, this processing can be computationally intensive.

It has been shown that (in small rooms) increasing the level of individual early reflections in a simulated room environment the differences are first observed in sound coloration and then (after an increase of 2-4 dB) in spatial aspects, such as spatial impression [2]. Consequently in ViMiC the levels of direct sound and early reflections can be modified independently.

1.5.2 Late Reverb

The late reverberant field of a room is often considered nearly diffuse without directional information. Thus, an efficient late reverb model, based on a feedback delay network [7] with 16 modulated delay lines diffused by a Hadamard mixing matrix, is used. By feeding the outputs of the room model into the late reverb a diffused reverb tail is synthesized (see Fig. 1.3), for which timbral and temporal character can be modified. This late reverb can be efficiently shared across several rendered sound sources. For detailed information about the implementation of the Late reverb see chapter 2.5 on page 23.

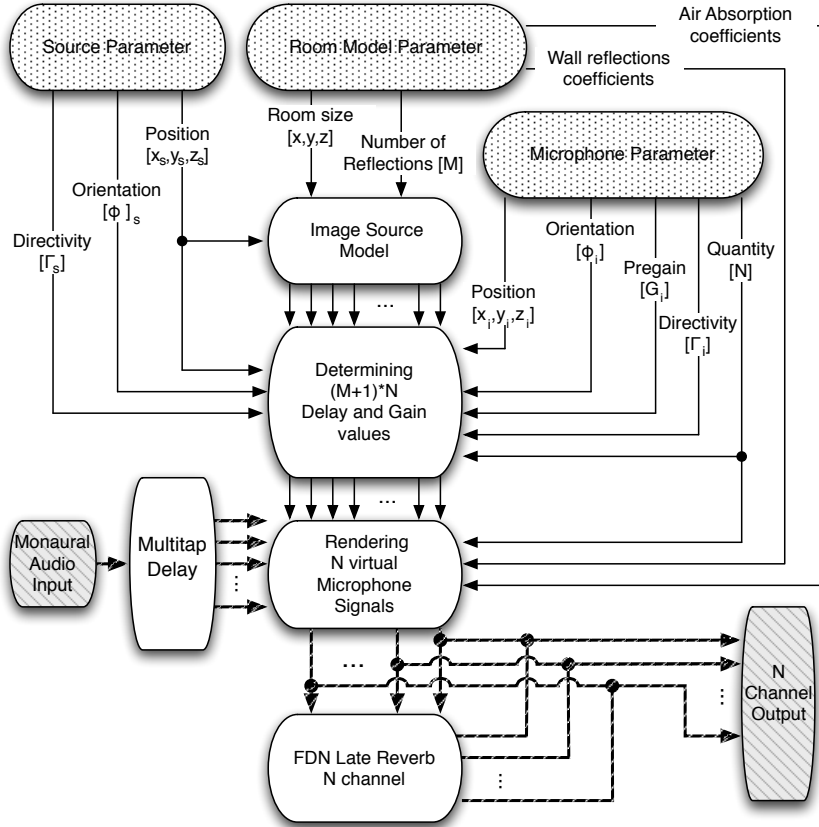


Figure 1.3: Flowchart of the Max/MSP processing

1.6 Moving Sources

In Figure 1.4 the sound source moved from (x, y, z) to (x', y', z') , changing the propagation paths to all microphones, and also, the time delay and attenuation. A continuous change in time delay engenders a pitch change (Doppler effect) that creates a very realistic impression of a moving sound source. Doppler effect might not always be desired. ViMiC accommodates both scenarios.

1.6.1 Rendering with Doppler effect

For each changed sound path, the change in time delay is addressed through a 4-pole interpolated delay-line, the perceived quality of which is significantly better than with an economical linear interpolation. To save resources, interpolation is only applied when moving the virtual sound source, otherwise the time delay is being rounded to the next non-fractional delay value. At $fs = 44.1$ kHz and a speed-of-sound of $c = 344$ m/s, the roundoff error is approximately 4 mm. Some discrete reflections might not be perceptually important due to the applied distance law, microphone characteristics, and source directivity. To minimize processor load, an amplitude threshold can be set to prevent the algorithm from rendering these reflections.

1.6.2 Rendering without Doppler effect

This render method works without interpolation: the time delays of the rendered sound paths remains static until one of the paths has been changed by more than a specified time delay. In this case, the sound paths of the old and the new sound position are cross-faded within 50 ms, in order to avoid strongly audible phase modulations.

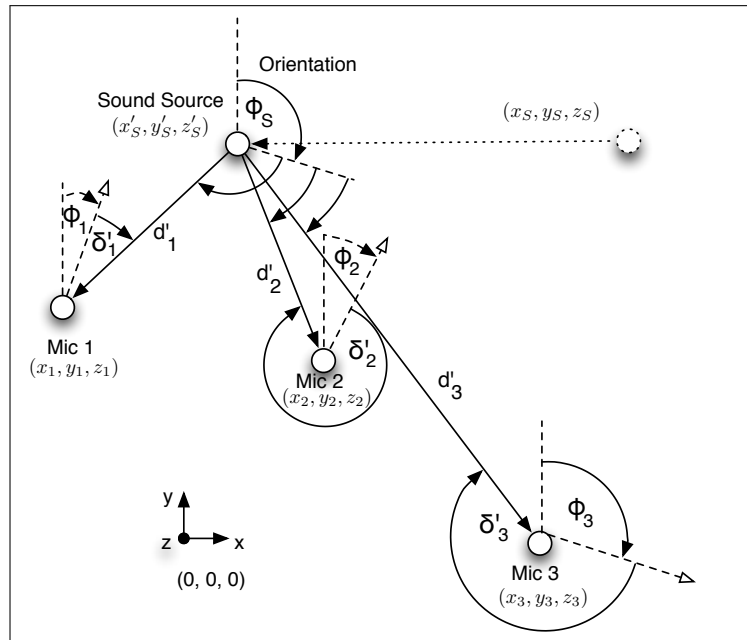


Figure 1.4: Moving sources, geometric example

1.7 Getting started

This manual explains the features of the ViMiC system developed for the Jamoma environment in Max/MSP. These four modules are entitled

- `jmod.sur.vimic~`, starting at page 9
- `jmod.sur.vimic8~`, starting at page 20
- `jmod.sur.vimic8poly~`, starting at page 21
- `jmod.sur.reverb~`, starting at page 23

All modules have a dedicated helpfile, e.g. `jmod.sur.vimic~.maxhelp`. Like all Jamoma modules, the ViMiC modules are fully controlled through a GUI and through external Open Sound Control messages [18]. In the following Sections, each GUI element is explained and its associated OSC message is provided.

For easier and more flexible controllability, ViMiC was structured as high-level modules using the Jamoma¹ framework for Max/MSP/Jitter [15]. Jamoma offers a clear advantage in its standardization of presets and parameter handling. ViMiC parameters have been sorted into three primary namespaces: *source*, *microphone* and *room*; and have specified attributes.

Jamoma modules can be created in Max/MSP in two ways, as explained below. For further information on Jamoma, please navigate to the growing tutorial section at jamoma.org.

Loading ViMiC as a subpatcher

1. Create a new empty Max patch.
2. Create a new object² and type `jmod.sur.vimic~ test`. This should create an object with two inlets and two outlets, similar to Figure 1.5. The module's OSC name assigned to this object is `/test`.

¹<http://www.jamoma.org>

²A new object box can be created by using Max5's shortcut `n`.

3. Double-clicking the object opens a small pop-up window, offering the user interface for this object.
4. Click on the upper left circle to close the interface.



Figure 1.5: Loading ViMiC as a subpatcher, the GUI is hidden and appears when double-clicking on it

Loading ViMiC in a bpatcher context

If we instead want to embed the interface in the patch, we can load the ViMiC module as a bpatcher:

1. Create a new empty Max patch.
2. Create a new bpatcher object.
3. Open the inspector, and set Patcher File to `jmod.sur.vimic~`
4. Assign a unique OSC name to the module by giving an argument, e.g. `test` (see Figure 1.6).
5. Close the bpatcher inspector.
6. The bpatcher should now have two inlets and two outlets.

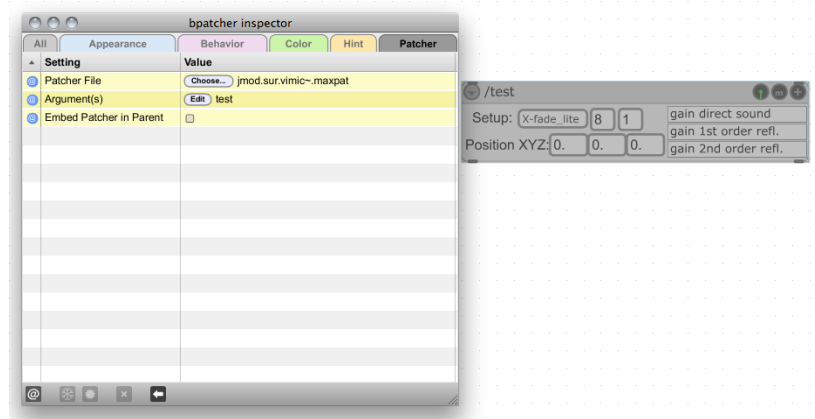


Figure 1.6: Loading ViMiC in a bpatcher

Alternatively to the steps 2 - 5, one can create a new object box and type `bpatcher @name jmod.sur.vimic~ @args test` (see Figure 1.7) for an example. This creation is facilitated by using Jamoma's keyboard shortcut `shift + B` which provides a pre-configured bpatcher object. Both display options provide the same kind of functionality, it is just a matter of taste or space in the Max patch



Figure 1.7: creating a bpatcher for ViMiC, using the object box

what display mode is preferred.

Connecting ViMiC

Audio connection The ViMiC module has two inlets and two outlets. The right inlet feeds the audio signal for processing in ViMiC, and the right outlet delivers the processed multichannel audio in the form of a Jamoma multicable. Jamoma multicable is an audio connection that can contains up to 32 discrete audio channels in one connection. This facilitates especially the work in context with with spatializaiton because the connection of multichannel audio modules is facilitated. For further information, see [16]. The multicable output contains up

to 24 discrete audio channels. Rather than manually connecting these 24 discrete audio channels to the desired destination (e.g., to the `dac~`), the Jamoma multicables provide more convenient solutions. In Figure 1.8 a simple noise signal is spatialized with ViMiC, and the processed multichannel audio signals are connected to the Jamoma module `jmod.sur.output~`, which distributes the multicable audio signal to the physical outputs of the sound card.

OSC-message connection As usual in Jamoma modules, the left inlet and the left outlet are reserved for OSC control messages to and from the object. Every parameter change, either due to a received control message or due to a manipulation in the Interface, will be reflected in the left outlet. In Figure 1.8, the left outlet is connected to a message box in order to display the last parameter change.

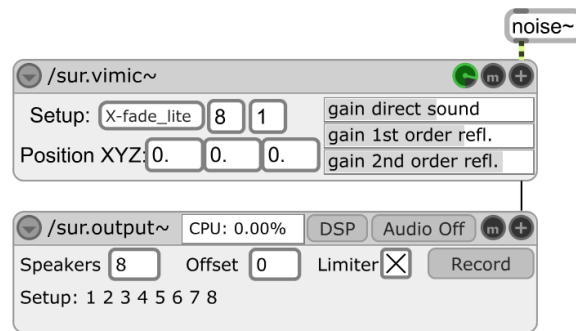


Figure 1.8: Here the multicable output is connected to `jmod.sur.output~`

Chapter 2

The Jamoma Modules

2.1 Parameter & controlling of jmod.sur.vimic~

2.1.1 Front-end

Identifier	Name	Description
A	Message inlet for OSC-messages	p. 8
B	Message outlet for OSC-messages	p. 8
C	Signal inlet	p. 8
D	Multichannel signal outlet	p. 8
1	Module pop-up menu	p. 9
2	Modules OSC name	p. 10
3	Rendermodes pop up menu	p. 10
4	Number of pre-configured microphones	p. 10
5	Order of rendered early reflections	p. 10
6	Sound Source Position	p. 10
7	Mute button	p. 10
8	Inspector button	p. 11, p. 11
9	Gain control for the direct sound component	p. 11
10	Gain control for the 1st-order early reflections	p. 11
11	Gain control for the 2nd-order early reflections	p. 11
12	Gain control for the output volume	p. 11

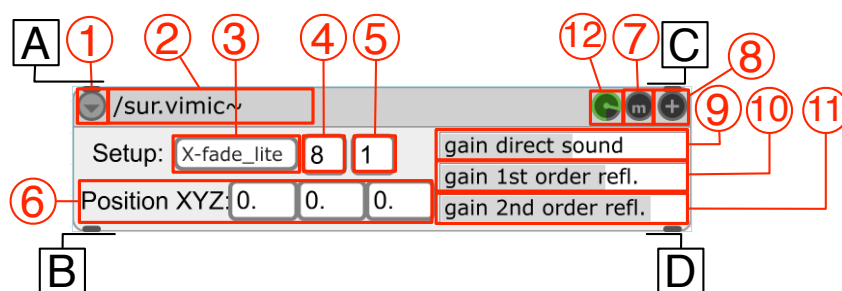


Figure 2.1: *jmod.sur.vimic~* main control interface

In Figure 2.1 the front end of the module is shown. Here the most often used parameter are shown, with the most often used parameters shown. Information about each controllable parameter follow, including its equivalent OSC-message

(1) - The module pop-up menu Provides several Jamoma specific option. For more information, the reader is directed to the Jamoma documentation. This menu also gives access to a variety of modules presets, holding popular multichannel microphone setups.

(2) - The module's OSC name represents the first OSC branch in a parameter namespace. The uniqueness of the module's OSC name becomes crucial if an OSC-message is not sent when the module is controlled remotely e.g. through *jmod.cuelist* as opposed to being sent directly to the module's left inlet. Jamoma will rename the modules to provide a unique identifier by adding a number at the end of the name. e.g. a duplicated name */test* becomes */test.1* and so on.

(3) - The Rendermode From a drop down menu, different rendermodes can be chosen. The rendermodes differ in the quality of calculation of early reflections and the handling of the Doppler effect for moving sound sources. At the moment the following methods, can be selected:

- ViMiC_XL • X.fade_XL • Panning
- ViMiC_lite • X.fade_lite • Static

For detailed information of the different render methods refer to [Section 2.4 on page 22](#).

Equivalent OSC-message

```
/rendermode <string>
```

(4) - Number of pre-configured microphones In a drop down menu, the maximum number of desired virtual microphones inside the ViMiC system can be selected. A change is performed only if the DSP is turned off.

Equivalent OSC-message

```
/microphones/amount <int> [1 .. 24]
```

(5) - Order of rendered early reflections In a drop down menu, the quantity of early reflections is specified by selecting the reflection order. The higher the order number, the more early reflections are rendered:

Order	Rendered reflections
0	0
1	6
2	18

Update: this parameter has been removed - the order of reflections is now automatically determined according to the gain settings of the early reflections (see p. 11).

(6) - Sound Source Position The coordinate triplet defines the position of the sound source in the virtual room. According to the SpatDIF initiative [\[12, 13\]](#) a position can be defined in spherical, cartesian, or openGL coordinates through external OSC-messages. Convention of the coordinate system are listed on [page 29](#).

Equivalent OSC-message

```
/source.1/position <float float float> [x y z]
/source.1/position <float float float> opengl [x y z] for an openGL coordinate system
/source.1/position <float float float> aed [azimuth elevation distance]
```

(7) - The mute button If muted, the ViMiC stops rendering, but still calculates the geometric model, so that when the module is unmuted, the virtual sound source appears at the most updated position.

Equivalent OSC-message

```
/audio/mute <boolean> [0, 1]
```

(8) - The inspector button By clicking on the \oplus button in the upper right corner, ViMiC's inspector window (Figure 2.2) opens to give control over other parameters described in Section 2.1.2.

Equivalent OSC-message

```
/view/panel
```

(9) - Gain control for the direct sound component This horizontal fader controls the gain of the direct sound component at each virtual microphone and follows Jamoma's MIDI gain convention. A midi value 100.0 is similar to 0.0 dB which refers to a linear gain of 1.0 (see Appendix 3.1 on page 28). The SHIFT-key enables fine tuning of the slider values. the gain value can also be relatively increased/decreased as shown below.

Equivalent OSC-message

```
/room/reflection/gain.0 <float> [0.0 .. 127.0]
/room/reflection/gain.0:/value/inc
/room/reflection/gain.0:/value/dec
```

(10) - Gain control for the 1st-order early reflections If the pop up menu (5) is set to the rendered reflection order of one or higher, this horizontal fader controls the gain of the 1st-order reflections at each virtual microphone. If no early reflections are rendered (when the parameter described in section (5) is set to zero), the fader is disabled.

Equivalent OSC-message

```
/room/reflection/gain.1 <midi value> [0.0 .. 127.0]
```

(11) - Gain control for the 2nd-order early reflections If the pop up menu (5) is set to the rendered reflection order of two, this horizontal fader controls the gain of the 2nd-order reflections at each virtual microphone. If no 2nd-order reflections are rendered (when the parameter described in section (5) on page 10 lower than two), the fader is disabled.

Equivalent OSC-message

```
/room/reflection/gain.2 <midi value> [0.0 .. 127.0]
```

(12) - Gain control for the output volume The green knob controls the output volume of the module. parameter handling is equivalent to gain controller Nr. (9), (10) and (11). When changing its state, the current gain value is displayed in the area of the OSC module name (see (2) in Figure 2.1). Gain values can be higher than 0 dB, see MIDI gain convention on Section 3.1.

Equivalent OSC-message

```
/audio/gain <midi value> [0.0 .. 127.0]
```

2.1.2 Inspector Interface

By clicking on the inspector button, or by sending the OSC message `/view/panel` to the module, the inspector interface appears (see Figure 2.2). The biggest part of the inspector window is filled with the channel strip for each virtual microphone (I), for which functionality is explained in the following.

Identifier	Name	Description
I	Channel strips for each microphone	p. 12
II	Room model parameters	p. 14
III	Source directivity settings	p. 15
IV	Settings for X-Fade Render-method	p. 17
V	Miscellaneous parameters	p. 18

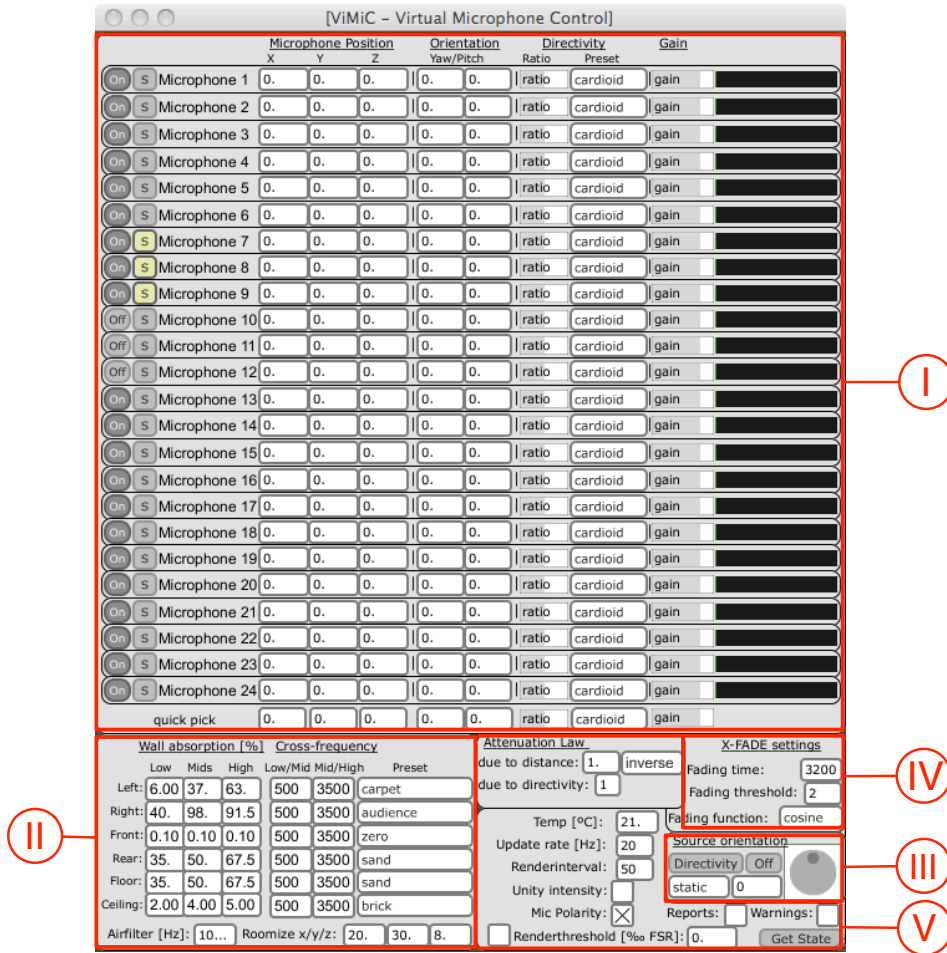


Figure 2.2: ViMiC inspector window

I - Microphone configuration

Figure 2.3 shows a part of the microphone configuration area. Like a channel strip in a mixing console, each strip has the same functionality. Therefore only one channel strip example is discussed. To access a specific microphone channel with an OSC-message, the displayed # in the OSC-namespaces has to be replaced by the number of the virtual microphone (1-24). The setting of (4) in the front-end (see page 10) determines the highest number a virtual microphone can have, e.g. if (4) is set to eight, only the first eight microphones are displayed. However, it is possible to change setting of higher numbered microphones through OSC messages without losing these settings. If (4) is changed later to use more virtual microphones, these settings will be recalled.

Identifier	Name	Description
I-1	Channel number	p. 12
I-2	Active button	p. 13
I-2b	Solo button	p. 13
I-3	Microphone position	p. 13
I-4	Microphone directivity ratio	p. 13
I-5	Microphone directivity presets	p. 13
I-6	Microphone orientation	p. 14
I-7	Microphone pre-gain	p. 14
I-8	VU-meter	p. 14
I-9	Quick pick configuration	p. 14

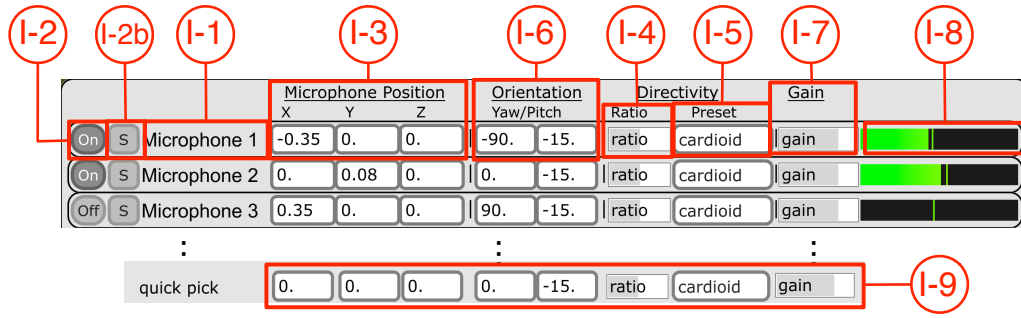


Figure 2.3: Channel Strip of Virtual Microphone

(I-1) - Channel number This displays the number of the related virtual microphone. The number of the virtual microphone is directly related to the output channel number; a parameter change in the channel strip of microphone Nr. 2 will affect the output signal Nr. 2 in the multicable.

(I-2) - The active button If active button is set to 0, the microphone will be muted. If the active button is set to 1, the microphone will be unmuted:

Equivalent OSC-message

```
/microphone.#/active <boolean> [0, 1]
```

Solo button The solo button acts as you might expect—it will mute all other microphone channels. Of course, it is possible to have more than one microphone in solo mode.

Equivalent OSC-message

```
/microphone.#/solo <int> [0, 1]
```

(I-3) - Microphone position The position of a microphone can either be set in cartesian or in spherical coordinates.

Equivalent OSC-message

```
/microphone.#/position <float float float> [x-coord. y-coord. z-coord.]
/microphone.#/position/x <float> [x-coord.]
/microphone.#/position/y <float> [z-coord.]
/microphone.#/position/z <float> [z-coord.]
/microphone.#/position <float float float> aed [azimuth elevation distance]
```

(I-4) - Microphone directivity ratio As described in Section 1.5.1 on page 3, the microphone directivity is defined by the equation:

$$\Gamma = (a + (1 - a) \cdot \cos \delta)^w \quad 0 \leq a \leq 1 \quad (2.1)$$

The horizontal slider controls the ratio of the parameter a of this equation.

Equivalent OSC-message

```
/microphone.#/directivity/ratio <float> [0.0 .. 1.0]
```

(I-5) - Microphone directivity presets Rather than manually setting the directivity of a virtual microphone, this drop down menu serves several predefined microphone directivity patterns. The following presets are available: omni, subcardioid, cardioid, supercardioid, hypercardioid, or figure-eight.

Equivalent OSC-message

```
/microphone.#/directivity/preset <string>
```

(I-6) - Microphone orientation A microphone can point in a certain direction. This feature becomes important when a virtual microphone is designed with a specified directivity. Yaw-angle ($-180.0^\circ..180.0^\circ$) and pitch-angle ($-90.0^\circ..90.0^\circ$) can be set. See also Jamoma’s coordinate system convention in Appendix 3.2 on page 29.

Equivalent OSC-message

```
/microphone.#/orientation <float float>[yaw-angle pitch-angle]
/microphone.#/orientation/yaw <float> [yaw-angle]
/microphone.#/orientation/pitch <float> [pitch-angle]
```

(I-7) - Microphone pre-gain This option can be used to modify the sensitivity of each microphone.

Equivalent OSC-message

```
/microphone.#/gain <float> [0.0 .. 127.0]
```

(I-8) - VU-meter The VU-meter is a classical AFL-VU-meter in that it shows the signal energy after the output volume control described on page 11. The integration time is around 300 ms.

To save some processing power, the VU-meter can be switched off by ticking the entry *Defeat Signal Meters* in the modules pop-up menu (see (1), on page 9).

(I-9) - Quick pick configuration For those situations when uniform parameters for all microphones are desired, rather than configuring each microphone individually, a quick pick configuration will apply desired parameter to all microphones.

For example, if all microphones should have a cardioid directivity, just select the cardioid directivity from the quick pick area.

II - Room model parameter

When early reflections are selected in the front-end menu (see (5) on page 10), the room model becomes audible. Early reflections are calculated according to the position of the sound source, the microphones and the room model (see 1.5.1 on page 3). Below the controllable room parameter are briefly explained.

The screenshot shows a control interface for room model parameters. It features a table with columns: Low, Mids, High, Low/Mid, Mid/High, and Preset. There are two rows for 'Left' and 'Right' channels. Below the table are controls for 'Airfilter [Hz]' and 'Roomize x/y/z'. Annotations in red circles point to specific elements: II-1 points to the 'Left' label, II-2 points to the 'High' column header, II-3 points to the 'Preset' column header, II-4 points to the 'Roomize x/y/z' controls, and II-5 points to the 'Airfilter [Hz]' control.

	Low	Mids	High	Low/Mid	Mid/High	Preset
Left:	99.9	99.9	99.9	500	3500	---
Right:	99.9	99.9	99.9	500	3500	---

Below the table, there are controls for 'Airfilter [Hz]: 10...' and 'Roomize x/y/z: 20. 30. 8.'.

Figure 2.4: Room model parameter

Identifier	Name	Description
II-1	Wall identifier	p. 14
II-2	Absorption parameter	p. 14
II-3	Absorption preset	p. 15
II-4	Room size	p. 15
II-5	Air absorption filter	p. 15

(II-1) - Wall identifier Indicates the name of the wall to which the parameter on the right belongs.

(II-2) - Absorption parameter The absorptive behavior of the room boundaries is modeled through a high-mid-low shelf filter. The wall absorption coefficient (α in architectural acoustics) in each band, the is defined in percent. The bigger the number, the higher the absorptive character. The crossover frequencies between the low, mid, and high band can be set as well.

Equivalent OSC-message

```
/room/absorption.##/low <float> [0.1 .. 99.9]
/room/absorption.##/mid <float> [0.1 .. 99.9]
/room/absorption.##/high <float> [0.1 .. 99.9]
/room/absorption.##/low_mid_frequency <int> [10 .. 9999]
/room/absorption.##/mid_high_frequency <int> [10 .. 9999]
```

The placeholder ## in the OSC-messages must be replaced with the wall identifiers `left`, `right`, `front`, `rear`, `floor`, or `ceiling`.

(II-3) - Absorption preset As explained above, several surface presets are available to set the Absorption parameter. a preset can be accessed with the message:

```
/room/absorption.left/preset <string>
```

Located in the ViMiC folder, the presets are stored in the textfile entitled `VimicSurfaceProperties.txt`. This file can be extended. A preset is stored as follows.

Absorption preset

```
<string>, <LF absorption> <MF absorption> <HF absorption> <L-M cross freq> <M-H cross freq>;
```

Example: The preset *audience* is stored as:

```
audience , 40. 98. 91.5 500 3500;
```

(II-4) - Room size The size of the virtual room is set in width, length and height. The unit is in meters. Each dimension is limited to 40 meters.

Equivalent OSC-message

```
/room/size/xyz <float float float> [width length height]
```

(II-5) - Air absorption filter As explained in Section 1.5.1 on page 3, the early reflections are filtered to simulate the damping of the air. This 2nd-order lowpass filters all early reflections in the same way.

Equivalent OSC-message

```
/room/reflection/airfilter <int> [500..19000]
```

III - Source Directivity

Identifier	Name	Description
III-1	Source directivity activator	p. 15
III-2	Source directivity inspector button	p. 16
III-3	Directivity mode	p. 16
III-4	Orientation wheel	p. 16
III-5	Directivity editor	p. 16
III-6	Editor tools	p. 16
III-7	Directivity presets	p. 17

(III-1) - Source directivity activator With the toggle, the source directivity calculation can be switched on/off.

Equivalent OSC-message

```
/source.1/orientation/active <boolean> [0,1]
```

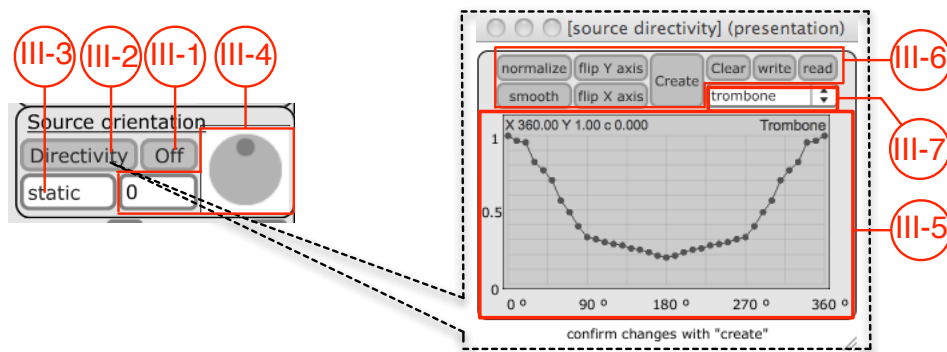



Figure 2.5: Source directivity options

(III-2) - Source directivity inspector button This button opens the source directivity editor, explained in (see III-5 and follow).

Equivalent OSC-message

```
/source .1/ directivity /openEditor
```

(III-3) - Directivity mode Three directivity modes are implemented:

static: The source orientation remains static when the sound source moves. In this case, the orientation can be set via the Orientation wheel or the equivalent OSC-message (see Section 2.1.2).

center: During movement, the source is always oriented towards the center of origin.

follow: During movement, the source is automatically oriented in the direction of the movement.

Equivalent OSC-message

```
/source .1/ orientation /mode <string> [ static , center , follow ]
```

(III-4) - Orientation wheel If the Orientation mode is set to *static*, this wheel and the connected number box can be used to control the orientation of the sound source. The knob of the wheel shows the direction where the front (the 0° direction) of the sound source is pointing. The Orientation is defined in the range of [−180° to 180°]. Values outside this range are automatically wrapped. If the Source directivity activator (see Paragraph 2.1.2) is disabled, the Orientation wheel is also automatically disabled in the GUI.

Equivalent OSC-message

```
/source .1/ orientation /yaw <int> [−180 .. 180]
```

(III-5) - Directivity editor Using the mouse, the directivity of a sound source can be freely designed. It works like a breakpoint function editor. A line can be transformed into a curve by pressing the ALT-key while dragging the line with the mouse. Selecting a breakpoint while holding SHIFT will delete this point. To confirm an edited source directivity, press the *create* button (see III-6).

(III-6) - Editor tools These tools for creative editing of the source directivity:

smooth: smoothes out hard corners.

normalize: normalizes the directivity between the values 0.0 and 1.0.

flip Y axis: horizontally flip at the 0.5 point of the attenuation axis.

flip X axis: vertically flip at the 180 degree point.

clear: resets the editor display to a straight line at the 1.0 value.

create: confirms the edited curve.

write: saves a curve in a separate file.

read: loads a directivity from an external file. If a directivity function was stored by using the *write* button, it can be recalled either through the interface by using the *read* button, or via OSC.

Equivalent OSC-message

```
/source .1/ directivity /loadFile <string>
```

(III-7) - Directivity presets Some presets are included which can be either loaded through the pop-up menu or by using the following OSC-message with the name of the preset as the argument.

Equivalent OSC-message

```
/source .1/ directivity /preset <string>
```

IV - X-fade setting

The following parameters are specific for the `x_fade` render methods. More information for the `x_fade` render methods can be found in section 1.6 and in section 2.4. These parameters can only be changed if the sound source is currently not in motion.

(IV-1) Fading threshold This parameter sets the distance in samples to the extend to which one of the propagation paths has to change, in order to trigger a cross-fade from the current position to a new position.

Equivalent OSC-message

```
/rendermode /xfade /threshold <int> [1 .. 512]
```

(IV-2) Fading time These parameter controls the length of the cross-fade, which is triggered when one of the propagation paths changes more than the sample value set in IV-1. The unit of this parameter is samples.

Equivalent OSC-message

```
/rendermode /xfade /fadelength <int> [10 .. 9999]
```

(IV-3) Fading function The cross-fades can have different envelopes; the following envelope functions are implemented:

1. `cosine` - Cosine envelope, (equal intensity)
2. `cosine_squared` - Cosine squared envelope
3. `linear` - Linear envelope, (equal amplitude)
4. `tanh` - Tangent hyperbolic envelope
5. `sqrt` - Square-root envelope

Equivalent OSC-message

```
/rendermode/xfade/fadefunction <string> [cosine, cosine_squared, linear, tanh, sqrt]
```

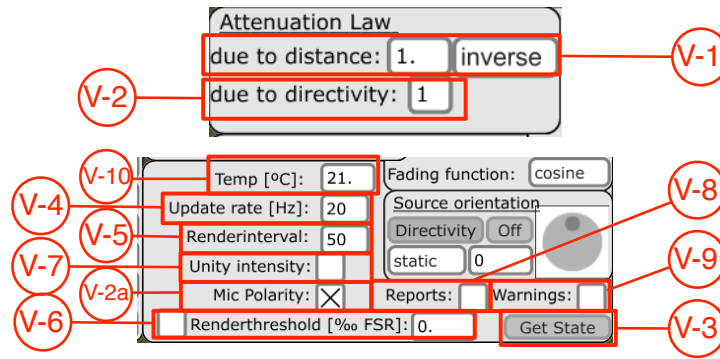


Figure 2.6: Miscellaneous parameter options

V - Miscellaneous parameters

Identifier	Name	Description
V-1	Distance law settings	p. 18
V-2	Microphone directivity exponent	p. 18
V-2a	Microphone polarity	p. 19
V-3	“Get State” button	p. 19
V-4	ViMiC update rate	p. 19
V-5	ViMiC renderinterval	p. 19
V-6	Renderthreshold	p. 19
V-7	Intensity normalization	p. 19
V-8	Warning button	p. 20
V-9	Report button	p. 20
V-10	Room temperature	p. 20
V-11	Minimize Delay	p. 20

(V-1) - Distance law exponent Two distance functions are implemented, see section 1.5.1 for a comparison of the two distance functions.

Equivalent OSC-message

```
/room/distance/mode <string> [inverse , exponential]
```

Inverse proportional decrease: The parameter sets the exponent q in the attenuation function described in equation 1.1 on page 3. Thus, the effect of distance attenuation can be boosted or softened. According to the radiation of a point source in the free field, this factor has usually a value of 1.0.

Equivalent OSC-message

```
/room/distance/power <float> [0.0 .. 9.0]
```

Exponential decrease: The parameter k from equation 1.2 can be modified.

Equivalent OSC-message

```
/room/distance/dbUnit <float> [0.0 .. 60.0]
```

(V-2) - Microphone directivity exponent This parameter sets the exponent w in the attenuation function described in Equation 1.3 on page 3. Increasing the exponent w to a value greater than 1 will produce an artificially sharper directivity pattern. Furthermore, by applying even numbered parameters, anti-phased gain values are avoided. Note that only integer values are allowed for this exponent.

Equivalent OSC-message

```
/microphones/directivity/power <int> [0 .. 9]
```

(V-2a) Microphone polarity / In-phase restriction If a microphone directivity ratio is smaller than 0.5 (e.g. a figure-of-eight microphone setting), for certain angles of incidence, a negative attenuation factor will be calculated. In order to avoid these anti-phase components, this toggle can be disabled to keep the allowed microphone attenuation between zero and one and set all negative factors to zero.

By restricting the microphone’s polarity and with an appropriate microphone arrangement, the “Polarity-restricted cosine” panning function can be achieved as described in [9].

Equivalent OSC-message

```
/microphones/polarity <int> [0, 1]
```

(V-3) - “Get State” button By pressing this button, the inner state of the ViMiC-external is posted into the Max window. This is especially helpful for finding bugs.

Equivalent OSC-message

```
/getState
```

(V-4) - ViMiC update rate To save resources, not every parameter change causes ViMiC to update (re-render) its virtual microphone signals. This parameter sets the frequency for how often ViMiC is forced to re-render its virtual microphone signals.

Equivalent OSC-message

```
/updaterate <int> [0 .. 100]
```

Alternatively to this update rate, ViMiC can be forced to re-render the virtual microphone signals from an external clock through the OSC command:

```
/update
```

(V-5) - ViMiC renderinterval This parameter is used for the ViMiC Rendermodes and determines how many signal blocks are used in order to fulfill the delayline interpolation. The higher the number, the more time it takes for the interpolation. If the number of blocks is too small, audible artifacts appear.

Equivalent OSC-message

```
/rendermode/interval <int> [1 .. 200]
```

(V-6) - ViMiC renderthreshold Some discrete reflections might not be perceptually important due to the applied distance law, microphone characteristics, and source directivity. To minimize processor load, an amplitude threshold can be set to prevent the algorithm from rendering these reflections. This amplitude threshold can be set very precisely in one-tenth of a percent of the full scale range. This is an experimental parameter.

Equivalent OSC-message

```
/rendermode/threshold/active <boolean> [0, 1]
/rendermode/threshold <float> [0.0 .. 1000.0]
```

(V-7) - Intensity normalization By activating, all calculated gain values g_i are normalized in order to have unity intensity according to equation 2.2. This is an experimental option.

$$I = \sum_{i=1}^n g_i^2 = 1 \quad (2.2)$$

Equivalent OSC-message

```
/rendermode/normalizeGain/active <boolean> [0, 1]
```

(V-8) - The warning button If this button is ticked, different postings appear in the Max window. For example, if the sound source position exceeds the virtual room, ViMiC will post a warning message, but will not stop the rendering.

Equivalent OSC-message

```
/warning <int> [0, 1]
```

(V-9) - The report button If this button is ticked, every parameter change will cause postings to the Max Window.

Equivalent OSC-message

```
/report <int> [0, 1]
```

(V-10) - Room temperature As described in Section 1.5.1, the inter-channel time differences, depending on the spatial relation of the source and each microphone, are calculated for a given speed of sound. The speed of sound depends on the medium, and for air it primarily depends on the temperature, e.g. [17]. By default the temperature unit of ViMiC's virtual room is assumed to be in degree Celsius. However, Jamoma's dataspace feature allows the value to be set also in degree Fahrenheit or Kelvin.

Equivalent OSC-message

```
/room/temperature <float> [-20 .. 30]
```

(V-11) - Minimize Delay If this option is enabled, for each sound source, only the relative time-of-flight delays between all microphones with respect to the source location are rendered. For instance, if in a stereo setting, the time-of-flight delays between source and the two microphones are 5 ms and 6 ms, the rendered relative delays are 0 ms and 1 ms. This option also helps to minimize the perception of a doppler effect.

Equivalent OSC-message

```
/rendermode/minimizeDelay/active <boolean> [0, 1]
```

2.2 The module jmod.sur.vimic8~

This module contains basically 8 jmod.sur.vimic~ modules in one module. This has the advantage that a lot of settings, such as the microphone position and the orientation have to be made only once—rather than eight times. Because the module renders 8 sources, it has a lot more inputs than the regular jmod.sur.vimic8~. As usual in Jamoma, the first inlet is reserved for communication via OSC. The next eight inlets are conventional signal inlets to connect audio sources directly to the rendering algorithm. The rightmost inlet is reserved for Jamoma's multicables whereas its first eight signals are internally separated and routed to the rendering algorithm. Because there are now eight sources to be controlled, the OSC namespace is slightly extended for the sources, compared to the namespace in jmod.sur.vimic~. All messages related to /source.1 are extended with an index starting from 1 to 8, e.g:

OSC-message example

```
/source.2/gain 100.0
/source.5/orientation 23.0 0.0
```

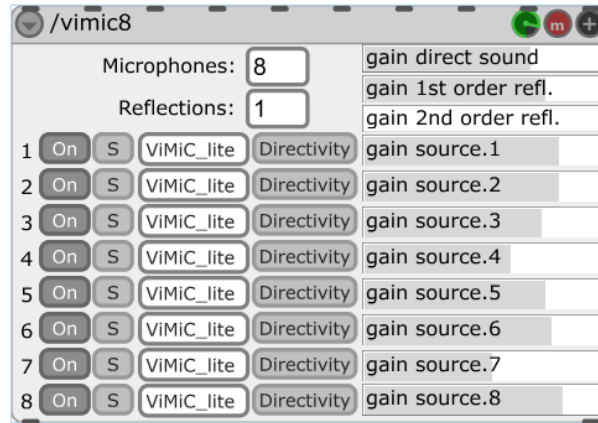


Figure 2.7: The module *jmod.sur.vimic8~*

2.2.1 Description message

In order to make the GUI more informative, the text displayed in the gain slider of each source can be modified with the following OSC message:

OSC-message to set a description string

```
/source.#/name <string> /source.1/name piano
```

This string will also appear in the window that controls the source directivity.

2.3 The module *jmod.sur.vimic8poly~*

The module *jmod.sur.vimic8poly~* is basically the same as *jmod.sur.vimic8~*, but uses the multithreading functionalities of Max5 offered by the *poly~* external. If you are using a multi-core computer, the rendering of the eight sound sources can be distributed amongst different CPUs. There are two new parameters to control the multi-threading:

multithreading OSC-message

```
/parallelization/active <boolean> [0, 1]
/parallelization/numThreads <int> [0 .. 8]
```

If */parallelization/numThreads* is set to 0, then Max/MSP will automatically use as many cores as available. An integer number higher than 0 specifies explicitly how many cores are going to be used. These two parameters can also be manipulated over the panel (Figure 2.8).

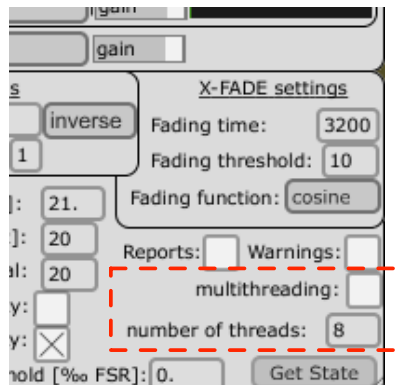


Figure 2.8: Multithreading options in *jmod.sur.vimic8poly~*

2.4 Rendermodes

ViMiC holds several DSP rendermodes that one can chose from according to the artistic aspiration or limitation of the DSP resources (see item (3) on page 10). The following section describes each of these modes.

ViMiC_XL If a sound moves, ViMiC_XL renders the sound rays using delay-line interpolation, which causes an audible Doppler effect (see section 1.6). Each wall of the virtual room can be simulated with a separate surface properties by setting the filter coefficients for the wall as described in section 2.1.2.

X_fade_XL This mode renders moving sound sources with the cross-fade method, described in section 1.6. The special X-Fade settings in the ViMiC inspector (see section 2.1.2 now become relevant. As the suffix XL in the name X_fade_XL states, the early reflections are rendered with respect to separate surface properties for each wall.

ViMiC_lite & X_fade_lite The render methods ViMiC_lite and X_fade_lite differ from ViMiC_XL and X_fade_XL in the quality of the room model: Instead of having the choice of individual wall absorption parameters for each wall, the entire virtual room is being modelled with the wall absorption of the “left” wall, the first wall in the ViMiC Room settings in the ViMiC inspector panel (see Section 2.1.2).

Static This rendermode is similar to ViMiC_lite, but no delay-line interpolation is performed. This setting is a “cheap and dirty” rendering method. It’s worth a try if one is short in DSP resources or if sources are not moving.

Panning This mode does not apply the calculated time delays to the signals. It just takes the gain values according to distance, source directivity and microphone characteristic into account. No early reflections are rendered, regardless of the settings in in (5), page 10, because the room reflections would appear without delay. If the gain normalization option is active (V-7), and microphone characteristic is set to omni, Distance Based Amplitude Panning (DBAP) [8] is created.

2.4.1 Comparison & CPU load

Rendermode	Source is...	No Reflections			1st-order Reflections			2nd-order Reflections		
		8 mics	16 mics	24 mics	8 mics	16 mics	24 mics	8 mics	16 mics	24 mics
ViMiC_XL	stationary	7.4%	12.3%	14.6%	15.5%	28%	40.7%	30.6%	59.2%	+++
	moving	8.6%	14.5%	18.3%	23.4%	42.5%	63.4%	50.6%	+++	+++
X_fade_XL	stationary	7.6%	12.5%	15.6%	17.3%	31%	45.5%	35.8%	68%	+++
	moving	8.2%	14%	16.4%	20.9%	37.6%	54.6%	43.4%	86%	+++
ViMiC_lite	stationary	7.2%	12%	214.4%	10.8%	26.5%	25.5%	15.4%	28%	40.0%
	moving	8.3%	13.8	17.6%	17.4%	31%	44.9%	33.2%	60%	+++
X_fade_lite	stationary	8%	13.2%	16.5%	13.5%	26%	34.0%	22.5%	42%	60.9%
	moving	8.2%	14%	17.0%	14.0%	25.8 %	34.6%	22.2%	41.2%	58.3%
Panning	stationary	6.4%	10.5%	12.2%	-	-	-	-	-	-
	moving	6.5%	10.5%	12.4%	-	-	-	-	-	-

Table 2.1: Max/MSP Processor load for one sound source at different render settings, Max/MSP overdrive option disabled, $f_s = 44.1kHz@16Bit$, measured on a mac mini 1.66 GHz intel core duo

+++ indicates processor overload

Max/MSP Version 4.6.3

Values are taken from the internal Max/MSP “CPU Utilization” information

2.5 The module `jmod.sur.reverb~`

2.5.1 Introduction

The late reverb is implemented as a feedback delay network (FDN) with 16 delay lines that models later reflections as an exponentially decaying random gaussian process, characterized by a spectral envelope. A flowchart of the algorithm is displayed in Figure 2.9. Because late reverberation decays can be approximated with a stochastic model, the use of FDNs for artificial reverberation is legitimate, assuming that the acoustic model has sufficient density in the frequency domain and sufficient number of reflections in the time domain. An FDN late reverb cannot guarantee the same accuracy as a convolution-reverb using a measured impulse response, but provides a more efficient parametrization for dynamic and flexible control of the reverberation effect, such as reverberation time, modal density or reverb coloration.

Any FDN can be represented as a set of digital delay lines for which the inputs and outputs are connected by a feedback/mixing matrix. This matrix provides diffusion by mixing the energy of every input to every output. The matrix is supposed to be unitary (or orthogonal). A system is said to be unitary if its matrix transfer function is unitary for any complex variable z on the unit circle. Another requirement is that the matrix has to be lossless. For this implementation, a 16x16 Hadamard matrix is used. The advantage of this matrix is that it can be efficiently implemented with a butterfly algorithm using $N \log 2N = 64$ additions (Figure 2.10). As proposed in [6], the delay-lines in the FDN can be modulated by low frequency oscillators in terms of delay-length and feedback attenuation coefficient.

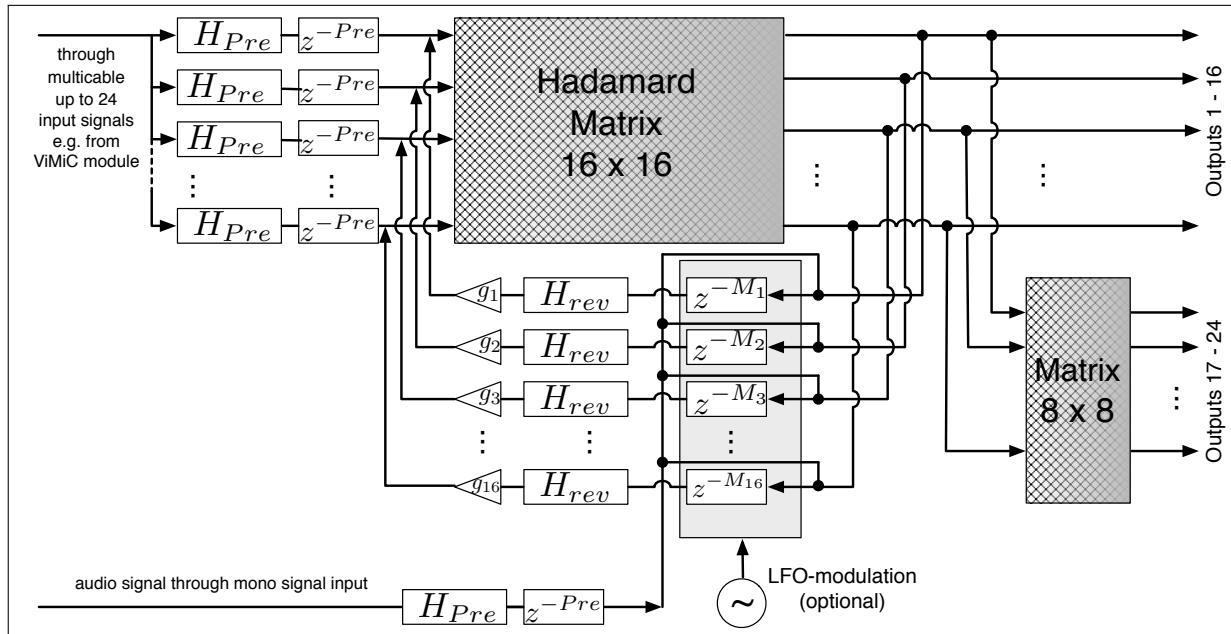


Figure 2.9: FDN-reverb flowchart

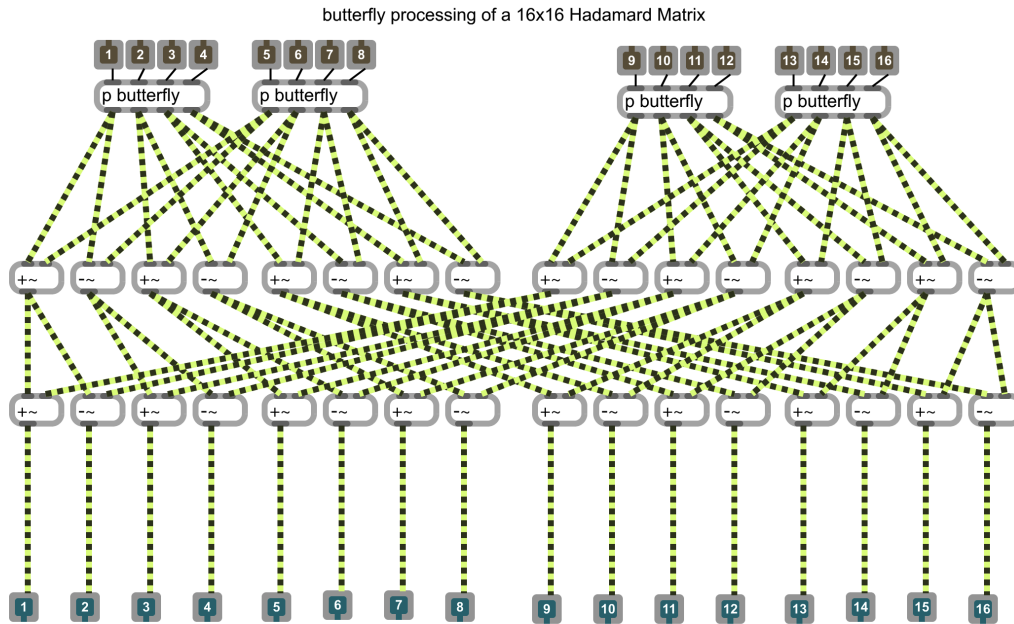


Figure 2.10: Butterfly implementation of the FDN 16x16 Hadamard matrix

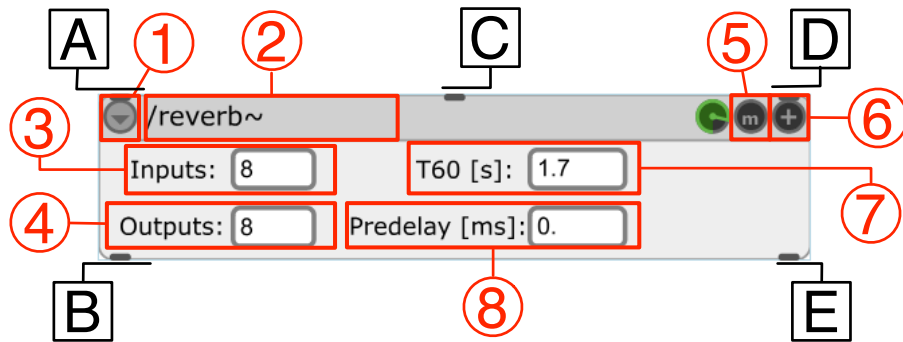


Figure 2.11: *jmod.sur.reverb~* main interface

2.5.2 Parameter & Controlling

Front-end

Identifier	Name	Description
A	Message inlet for OSC-messages	p. 24
B	Message outlet for OSC-messages	p. 24
C	Multichannel signal inlet	p. 24
D	Mono signal inlet	p. 24
E	Multicable signal outlet	p. 24
1	Module pop-up menu	p. 25
2	Modules OSC name	p. 25
3	Number of input channels	p. 25
4	Number of output channels	p. 25
5	Mute button	p. 25
6	Inspector button	p. 25
7	Reverb time T_{60}	p. 25
8	Pre-delay	p. 25

Reverb connections As depicted in Figure 2.11, the reverb module has three inlets and two outlets. The connections A and B are used for OSC communication, whereas the others are meant to process audio signals. There

are two audio inlets, so that a mono audio signal (inlet D) coming from a dry sound source, or preprocessed multi-cable signals coming from ViMiC modules can be reverberated. Both inlets can be used in parallel as well.

Module pop-up menu This provides several Jamoma specific option. Reader are pointed to the Jamoma documentation.

Modules OSC name This field shows the OSC-name of the reverb module, a unique name that becomes important if an OSC-message is not send directly to the module's left inlet, but is controlled remotely e.g. through the `jmod.cuelist` module. No modules can have the same name.

Number of input channels The number of used inputs should be defined here. According to this value, some filters and delays from unused input will be disabled (z^{-pre} and H_{pre} in Figure 2.9). Thus DSP resources are optimized.

Equivalent OSC-message

```
/numInputs <int> [1 .. 16]
```

Number of output channels Determines how many audio channels the multi-cable contains at the output. A change in the number of outputs will only be effective when the DSP is switched off.

Equivalent OSC-message

```
/numOutputs <int> [1 .. 24]
```

Mute button If mute is activated, the reverb stops outputting the audio and internal DSP processes.

Equivalent OSC-message

```
/audio/mute <boolean> [0, 1]
```

Inspector button By clicking on the \oplus button in the upper right corner, the inspector window opens to give control over extended parameters described in 2.5.2.

Equivalent OSC-message

```
/view/panel
```

Reverb time T_{60} Reverberation time in ms. Technically speaking, it influences the gain values $g_1 - g_{16}$, depicted in Figure 2.9.

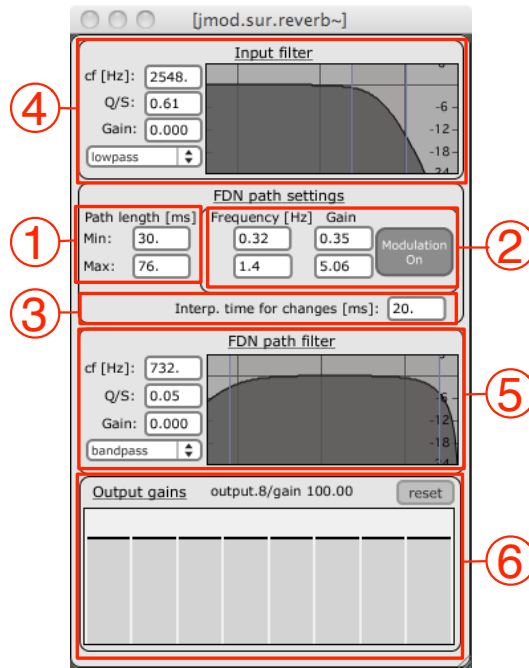
Equivalent OSC-message

```
/t60 <float> [0 .. 60.0]
```

Pre-delay This parameter delays the incoming signals to the reverb processing by the specified time in ms (depicted as z^{-pre} in Figure 2.9). This parameter is important to temporally align the late reverb correctly with the direct sound and early reflections rendered with ViMiC.

Equivalent OSC-message

```
/predelay <float> [0 .. 1000.0]
```



Inspector window

Identifier	Name	Description
1	Range of the FDN delay lengths	p. 26
2	FDN Modulation options	p. 26
3	Interpolation time	p. 26
4	Input filter settings	p. 27
5	FDN path filter settings	p. 27
6	Gain per output channel	p. 27

Range of the FDN delay lengths Minimal and maximal delay length of the feedback paths in ms. According to these settings, an algorithm will calculate orthogonal values for each of the 16 feedback paths internally.

Equivalent OSC-message

```
/fdn/delaylength <float float> [min-value max-value]
```

FDN Modulation options The feedback delay paths can be amplitude modulated to avoid ringing frequencies. For percussive sounds, this option sounds great. However, for sustained sounds such as guitar, or violins, the modulation might be audible and you might want to turn it off. According to the amplitude and frequency settings, an algorithm will calculate 16 orthogonal values within the given range to be used for the amplitude modulation.

Equivalent OSC-message

```
/modulation/active <boolean> [0,1]
/modulation/amplitude <float float> [min-value max-value]
/modulation/frequency <float float> [min-value max-value]
```

Interpolation time Interpolation time in ms for smooth interpolation of changes in the feedback path length (Section 2.5.2) and path modulation (Section 2.5.2).

Equivalent OSC-message

```
/interpolationtime <float> [0.0 .. 1000.0]
```

Pre-filter settings The input signal can be pre-filtered (H_{pre} in Figure 2.9) for instance to simulate air absorption.

Equivalent OSC-message

```
/input/frequency <float> [Hz]


```

FDN Filter settings These settings define the filter inside the feedback delay network (H_{rev} in Figure 2.9). Gain values higher than 0 dB (amplifications) are prohibited to ensure a stable reverb.

Equivalent OSC-message

```
/fdn/q <float> [Q or S value]
```

Gain per output channel The gain for each of the outputs can be manipulated independently. For instance, one could add more reverb to the surround channels than to the frontal loudspeakers if desired. Clicking on the “reset” button will set all outputs back to equal gain. The number of gain sliders will adapt automatically according to the defined number of outputs.

Equivalent OSC-message

```
/output.#/gain <float> [midigain]
```

2.5.3 CPU-load

The CPU-load depends up to a certain degree on the number of input/output channels and whether the length of the FDN-paths are being modulated by the LFOs (see section 2.5.2 on the preceding page).

Table 2.2: Max/MSP Processor load for one sound source at different late reverb settings, Max/MSP overdrive option disabled, $f_s = 44.1kHz@16Bit$, measured on a mac mini 1.66 GHz intel core duo. Numbers are taken from the internal Max/MSP “CPU Utilization” information.

Number of outputs		8	16	24
Inputs	FDN-modulation			
1	off	10 %	10.0 %	10.5%
1	on	11.5%	11.5%	12.0 %
8	off	10.6%	10.6%	11.0 %
8	on	12 %	12.0 %	12.5%
16	off	11 %	11.0 %	12.3%
16	on	12.5%	12.5%	13.0 %

Chapter 3

Appendix

3.1 Jamoma's MIDI to gain conversion

All messages controlling gain properties are expecting values in the MIDI range (0.0 - 127.0). The following table shows how the MIDI values correspond to a linear gain value. A MIDI value of 100 corresponds to a gain of 1. All MIDI values higher than 100 will consequently cause an amplification, whereas MIDI values below 100 will cause an attenuation.

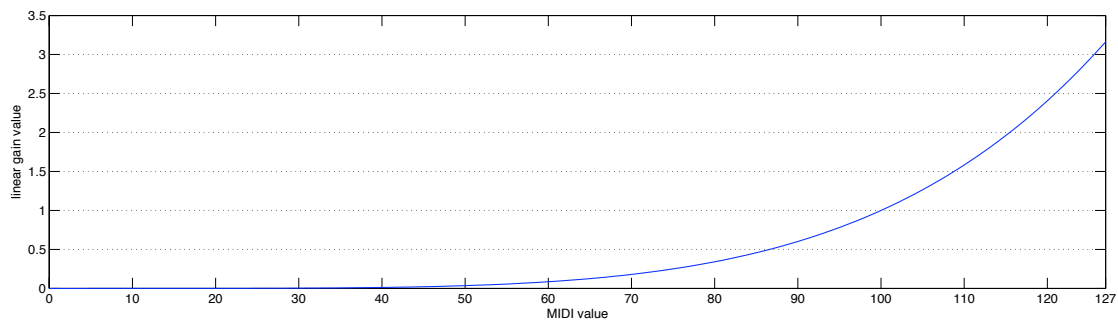


Figure 3.1: *Jamoma's MIDI to gain conversion*

3.2 Coordinate System

ViMiC uses a right-handed cartesian coordinate system, with limits defined by the virtual room size (see Section (II-4) on page 15).

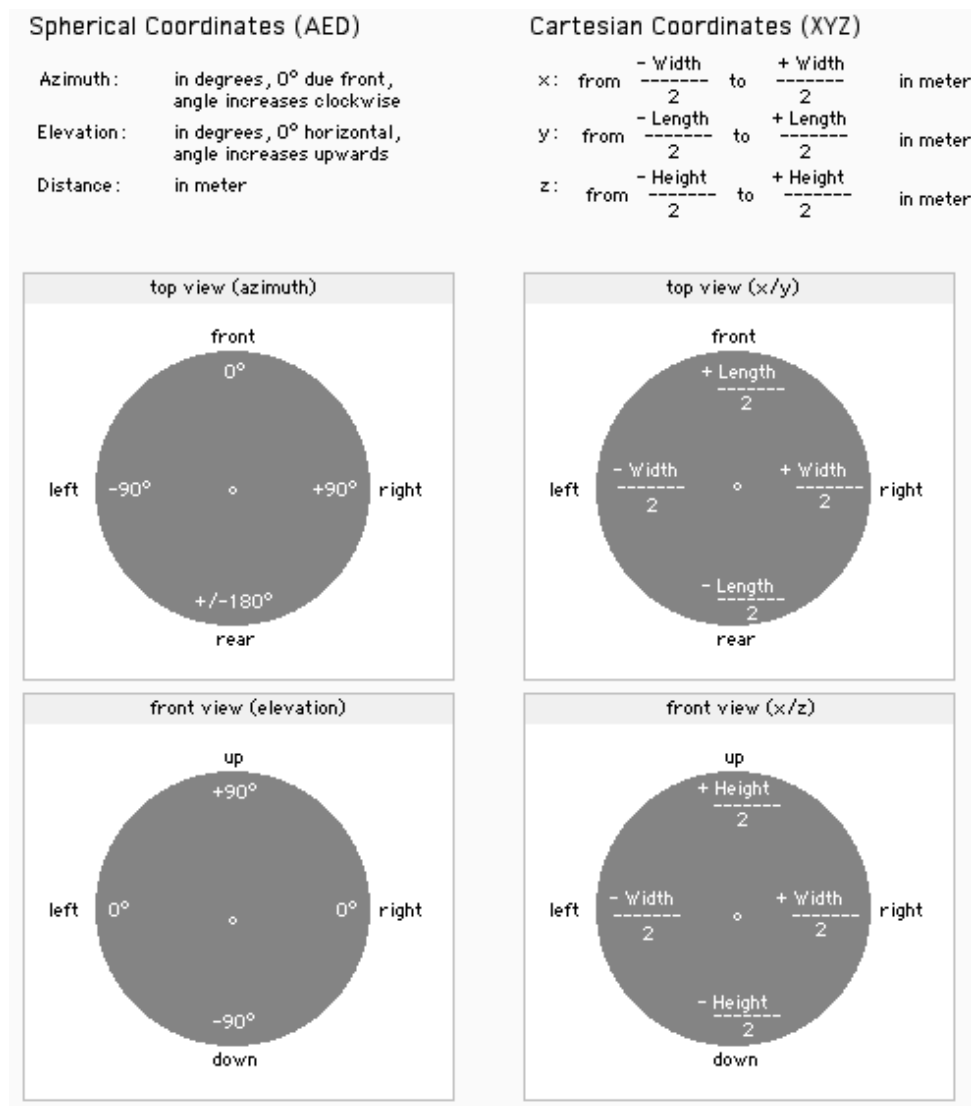


Figure 3.2: Coordinate Systems: Spherical (left) and Cartesian (right)

3.3 Known Issues

IF YOU SEE SOMETHING – SAY SOMETHING.

If you find something suspicious in this software, don't hesitate to inform the right person! Please send bug reports and suggestions to

nils_AT_music_DOT_mcgill_DOT_ca.

There is also a bugtracker at <http://redmine.jamoma.org/projects/vimic/issues>.

jmod.sur.vimic~

- ~~The first time the inspector opens takes quite long time.~~
- If the activation button of a microphone channel strip is turned off, it can cause a click, because the gain of this microphone is set to zero without being ramped.
- ~~Quick pick configuration for the x-coordinate and y-coordinate is not implemented yet.~~
- ~~The “follow” option for the source directivity is not implemented yet.~~
- After loading ViMiC, the “fading time” in X-Fade settings is disabled. It needs the DSP turned on to be manipulatable.
- “X_fade XL” produces nasty clicks when source moves.

jmod.sur.reverb~

- ~~After unmuting the module, the number of inputs are set to 16 and the number of outputs are set to 24 automatically.~~

Chapter 4

References

4.1 jmod.sur.vimic~

Configuration

Module Type:	audio
Algorithm Type:	default
Interface Size:	1U-half
Number of signal inlets:	1 MSP audio signal
Number of signal outlets:	1 Multicable with N audiochannels

Parameters

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
audio/gain	decimal	0.0 127.0	none	scheduler	linear	gain	midi	0	Set gain (as MIDI value by default).
audio/mute	boolean	0 1	none	none	none	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
microphone.N/active	boolean	0 1	both	none	none	none	none	1	if active is set to 0 then the Nth microphone is muted
microphone.N/directivity/ratio	decimal	0.0 1.0	both	scheduler	linear	none	none	0	Directivity value of the Nth microphone
microphone.N/gain	decimal	0.0 127.0	both	scheduler	linear	gain	midi	0	Gain of the Nth microphone
microphone.N/orientation	array	N/A	wrap	none	none	angle	deg	0	orientation [yaw pitch] of the Nth microphone
microphone.N/position	array	N/A	none	none	none	position	xyz	0	Position by default in xyz coordinate of the Nth microphone, add 'aed' or 'opengl' as a 4th argument to define spherical or openGL coordinates
microphones/amount	integer	1 24	both	none	linear	none	none	0	Number of rendered virtual microphones
microphones/directivity/power	integer	0 9	both	none	linear	none	none	0	power law for attenuation due to microphone directivity
microphones/polarity	boolean	0 1	none	none	linear	none	none	0	when unchecked, the polarity of all microphones is restricted - so all gain values are between 0 and 1, rather than between -1 and 1
rendermode	string	N/A	none	none	linear	none	none	0	Rendermode of ViMiC
rendermode/interval	integer	1 200	both	none	linear	none	none	0	tells ViMiC over how many signalblocks a change in position is interpolated - shorter time leads to faster transition and stronger doppler effect
rendermode/minimizeDelay/active	boolean	0 1	none	none	linear	none	none	0	minimizes the delay value to a minimum to maintain the interchannel delay differences
rendermode/normalizeGain/active	boolean	0 1	none	none	linear	none	none	0	when ticked, all calculated reflections are getting normalized to unity intensity
rendermode/threshold	decimal	0.000 1000.000	both	none	linear	none	none	0	renders only reflections above this amplitude value (full scale range)

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
rendermode/threshold/active	boolean	0 1	none	none	linear	none	none	0	when ticked, renderthreshold is active
rendermode/xfade/fade/function	string	N/A	none	none	linear	none	none	0	fading function of the crossfade for the x-fade mode
rendermode/xfade/fade/length	integer	0 9999	both	none	linear	none	none	0	length of the crossfade for the x-fade mode in samples
rendermode/xfade/threshold	integer	0 4096	low	none	linear	none	none	0	distance in samples before a crossfade in the x-fade mode is activated
report	integer	0 1	both	none	linear	none	none	0	report changes inside the vimic module
room/absorption.ceiling/high	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the high frequencies of the ceiling
room/absorption.ceiling/low	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the low frequencies of the ceiling
room/absorption.ceiling/low_mid.frequency	integer	10 9999	both	none	linear	time	Hz	1	low-mid crossover frequencies for the ceiling wall filter
room/absorption.ceiling/mid	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the mid frequencies of the ceiling
room/absorption.ceiling/mid_high.frequency	integer	10 9999	both	none	linear	time	Hz	1	mid-high crossover frequencies for the ceiling filter
room/absorption.floor/high	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the high frequencies of the floor wall
room/absorption.floor/low	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the low frequencies of the floor wall
room/absorption.floor/low_mid.frequency	integer	10 9999	both	none	linear	time	Hz	1	low-mid crossover frequencies for the floor filter
room/absorption.floor/mid	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the mid frequencies of the floor wall
room/absorption.floor/mid_high.frequency	integer	10 9999	both	none	linear	time	Hz	1	mid-high crossover frequencies for the floor filter
room/absorption.front/high	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the high frequencies of the front wall
room/absorption.front/low	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the low frequencies of the front wall
room/absorption.front/low_mid.frequency	integer	10 9999	both	none	linear	time	Hz	1	low-mid crossover frequencies for the front wall filter
room/absorption.front/mid	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the mid frequencies of the front wall
room/absorption.front/mid_high.frequency	integer	10 9999	both	none	linear	time	Hz	1	mid-high crossover frequencies for the front wall filter
room/absorption.left/high	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the high frequencies of the left wall
room/absorption.left/low	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the low frequencies of the left wall
room/absorption.left/low_mid.frequency	integer	10 9999	both	none	linear	time	Hz	1	low-mid crossover frequencies for the left wall filter
room/absorption.left/mid	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the mid frequencies of the left wall
room/absorption.left/mid_high.frequency	integer	10 9999	both	none	linear	time	Hz	1	mid-high crossover frequencies for the left wall filter
room/absorption.rear/high	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the high frequencies of the rear wall
room/absorption.rear/low	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the low frequencies of the rear wall
room/absorption.rear/low_mid.frequency	integer	10 9999	both	none	linear	time	Hz	1	low-mid crossover frequencies for the rear wall filter
room/absorption.rear/mid	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the mid frequencies of the rear wall
room/absorption.rear/mid_high.frequency	integer	10 9999	both	none	linear	time	Hz	1	absorption for the high frequencies of the rear wall
room/absorption.right/high	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the high frequencies of the right wall
room/absorption.right/low	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the low frequencies of the right wall
room/absorption.right/low_mid.frequency	integer	10 9999	both	none	linear	time	Hz	1	low-mid crossover frequencies for the right wall filter
room/absorption.right/mid	decimal	0.100 99.900	both	none	linear	none	none	1	absorption for the mid frequencies of the right wall
room/absorption.right/mid_high.frequency	integer	10 9999	both	none	linear	time	Hz	1	mid-high crossover frequencies for the right wall filter
room/distance/dbunit	decimal	0.000 60.000	both	none	linear	none	none	0	db-Unit for exponential attenuation function due to distance
room/distance/mode	string	N/A	none	none	linear	none	none	0	distance function: inverse or exponential
room/distance/power	decimal	0.000 9.000	both	none	linear	none	none	0	power law for inverse attenuation function due to distance
room/reflection/airfilter	integer	500 19000	both	none	linear	time	Hz	0	damping frequency of the early reflections due to air absorption

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
room/reflection/gain.0	decimal	0.000 1.000	none	scheduler	linear	gain	midi	0	gain of the direct sound component
room/reflection/gain.1	decimal	0.000 1.000	none	scheduler	linear	gain	midi	0	gain of the early reflections 1st order
room/reflection/gain.2	decimal	0.000 1.000	none	scheduler	linear	gain	midi	0	gain of the early reflections 2nd order
room/size/xyz	array	N/A	both	none	linear	none	none	0	Size of the virtual room in XYZ
room/temperature	decimal	-20.000 30.000	low	none	linear	temperature	C	1	Temperature in Celsius to calculate the speed of sound
source.1/orientation/active	boolean	0 1	none	none	linear	none	none	0	when ticked, source directivity is calculated
source.1/orientation/mode	string	N/A	none	none	linear	none	none	1	source orientation mode (center, static, or follow)
source.1/orientation/yaw	integer	-180 180	wrap	scheduler	linear	none	none	0	orientation of the sound source (yaw-angle)
update/rotate	integer	1 50	low	none	linear	time	Hz	0	vimic internal update rate in [Hz]- tells how often new parameter are getting updated
view/freeze	boolean	0 1	none	none	linear	none	none	0	Turn off the updating of user interface elements when parameters change. This may be done to conserve CPU resources.
view/highlight	string	N/A	none	none	linear	none	none	0	Highlight the module with a color tint such as red, green, or similar.
warning	integer	0 1	both	none	linear	none	none	0	turn on/off the printing of ViMiC's warning messages

Messages

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
documentation/generate	generic	0.000000 1.000000	none	none	none	none	none	1	Generate a html documentation page for this module and save it to disk. The argument 'tex' creates a LaTeX document.
getState	none	N/A	none	none	none	none	none	1	report inner state of the vimic module
init	none	N/A	none	none	none	none	none	1	Initialize a module completely to the original state.
microphone.N/directivity/preset	string	N/A	none	none	none	none	none	1	load a directivity preset for the Nth microphone
microphone.N/orientation/pitch	decimal	-180.0 180.0	wrap	scheduler	linear	angle	deg	0	Pitch angle of the Nth microphone
microphone.N/orientation/yaw	decimal	-180.0 180.0	wrap	scheduler	linear	angle	deg	0	Yaw angle for the orientation of the Nth microphone
microphone.N/position/x	decimal	-20.0 20.0	both	scheduler	linear	distance	m	0	X coordinate of the Nth microphone position
microphone.N/position/y	decimal	-20.0 20.0	both	scheduler	linear	distance	m	0	Y coordinate of the Nth microphone position
microphone.N/position/z	decimal	-20.0 20.0	both	scheduler	linear	distance	m	0	Z coordinate of the Nth microphone position
microphone.N/solo	boolean	0 1	none	none	none	none	none	0	activates solo mode for this microphone
preset/clear	none	N/A	none	none	linear	none	none	1	Clears all presets, providing a blank slate for saving new pre-sets.
preset/copy	array	N/A	none	none	linear	none	none	1	Create a new preset (2nd argument) by copying the contents of another preset (1st argument)
preset/default	none	N/A	none	none	linear	none	none	1	Open the default preset file and recall the first preset in that file.
preset/dump	none	N/A	none	none	linear	none	none	1	Dump all preset names.
preset/interpolate	array	N/A	none	none	linear	none	none	0	Interpolate between two named presets (argument 1 and 2) using a ratio (float in the range [0.0, 1.0]) specified as the third argument.
preset/mix	array	N/A	none	none	linear	none	none	0	Mix list of pairs of (preset name, mix value) using a ratio (float).
preset/post	none	N/A	none	none	linear	none	none	1	Post all presets to the Max window.
preset/read	generic	0.000 1.000	none	none	linear	none	none	1	Open an xml-preset file and recall the first preset in that file.
preset/recall	generic	0.000 1.000	none	none	linear	none	none	1	An optional argument defines the file to open. Recall a preset by number - you can also choose presets from the module menu.
preset/store	array	N/A	none	none	linear	none	none	1	Store a preset by number in memory. All presets present in memory will be written to disk when you send a /preset/write message to the module.
preset/storecurrent	none	N/A	none	none	linear	none	none	1	Store on the last recalled or stored preset
preset/storenext	none	N/A	none	none	linear	none	none	1	Store a preset in the next preset slot. Handy so that you do not need to specify a preset number manually.
preset/write	generic	0.000 1.000	none	none	linear	none	none	1	Write an xml-preset file to disk. An optional argument defines the file to open.
preset/writeagain	none	N/A	none	none	linear	none	none	1	Write on same xml-preset file.
room/absorption.ceiling/preset	string	N/A	none	none	linear	none	none	1	absorption presets for the floor
room/absorption.floor/preset	string	N/A	none	none	linear	none	none	1	absorption presets for the floor
room/absorption.front/preset	string	N/A	none	none	linear	none	none	1	absorption presets for the front wall
room/absorption.left/preset	string	N/A	none	none	linear	none	none	1	absorption presets for the left wall
room/absorption.rear/preset	string	N/A	none	none	linear	none	none	1	absorption presets for the rear wall
room/absorption.right/preset	string	N/A	none	none	linear	none	none	1	absorption presets for the right wall

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace /unit/native	/dataspace /allow	/repetitions	/description
source.1/directivity/loadFile	generic	0.000 1.000	none	none	linear	none	none	1	sound source directivity preset
source.1/directivity/openEditor	none	N/A	none	none	linear	none	none	0	opens the directivity editor
source.1/directivity/preset	string	N/A	none	none	linear	none	none	1	sound source directivity preset
source.1/position	array	N/A	none	scheduler	linear	position	xyz	0	Sourceposition in the virtual room - spatDIF compliant
update	none	N/A	none	none	linear	none	none	0	updates ViMiC manually (e.g. through an external clock), rather than automatically each time set with /updateRate
view/color/border	array	N/A	none	none	linear	none	none	0	The border color of the module in the format RGBA where values range [0.0, 1.0].
view/color/contentBackground	array	N/A	none	none	linear	none	none	0	The background color of the module in the format RGBA where values range [0.0, 1.0].
view/color/toolbarBackground	array	N/A	none	none	linear	none	none	0	The background color of the module's toolbar in the format RGBA where values range [0.0, 1.0].
view/color/toolbarText	array	N/A	none	none	linear	none	none	0	The color of the module's toolbar text in the format RGBA where values range [0.0, 1.0].
view/internals	none	N/A	none	none	linear	none	none	1	Attempts to open the internal algorithm for viewing. This works for most modules. Some modules may choose to cloak the algorithms - preventing this message from functioning.
view/panel	none	N/A	none	none	linear	none	none	0	Open an a module's control panel (inspector) if one is present.
view/refresh	none	N/A	none	none	linear	none	none	1	Update displayed values for module to reflect current state.
view/script	generic	0.000 1.000	none	none	linear	none	none	1	Low-level module hacking. Any arguments to this message will be interpreted as patcher scripting for the top-level patcher of the module.
view/size	array	N/A	none	none	linear	none	none	0	The size of the module's UI.

Returns

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
audio/amplitude.N	decimal	0.000000 1.000000	none		gain	linear	linear	0	instant amplitude of the signal number N

4.2 jmod.sur.reverb~

Configuration

Module Type: audio
Algorithm Type: default

Interface Size: 1U-half

Number of signal inlets: 2 : MSP signal + multicable
Number of signal outlets: 1 : multicable

Parameters

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
audio/gain	decimal	0.000 127.000	none	scheduler	linear	gain	midi	0	Set gain (as MIDI value by default).
audio/mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
fdn/delaylength	array	N/A	both	scheduler	linear	none	none	0	minimal and maximal delay/length of the feedback paths
fdn/filtergain	decimal	-24.000 0.000	both	scheduler	linear	none	none	1	Gain (dB)
fdn/filtertype	string	N/A	none	none	linear	none	none	0	What kind of filter to use. Possible values: lowpass — highpass — bandpass — bandstop — peaknotch — lowshelf — highshelf
fdn/frequency	decimal	30.000 11025.000	both	scheduler	linear	none	none	1	Center frequency (Hz)
fdn/modulation/active	integer	0 1	both	none	linear	none	none	0	activate modulation of the FDN delay path length
fdn/modulation/amplitude	array	N/A	low	scheduler	linear	none	none	0	minimal and maximal amplitude of the feedback path's length modulation
fdn/modulation/frequency	array	N/A	low	scheduler	linear	none	none	0	minimal and maximal modulation frequency of the feedback path's length
fdn/q	decimal	0.000 100.000	both	scheduler	linear	none	none	1	Resonance (Q)
input/filtergain	decimal	-24.000 24.000	both	scheduler	linear	none	none	1	Gain (dB)
input/filtertype	string	N/A	none	none	linear	none	none	0	What kind of filter to use. Possible values: lowpass — highpass — bandpass — bandstop — peaknotch — lowshelf — highshelf
input/frequency	decimal	30.000 11025.000	both	scheduler	linear	none	none	1	Center frequency (Hz)
input/q	decimal	0.000 100.000	both	scheduler	linear	none	none	1	Resonance (Q)

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
interpolationtime	decimal	0.000 1000.000	low	scheduler	linear	none	none	0	Interpolation time for changes of feedback path length and path modulation.
numInputs	integer	1 16	both	none	linear	none	none	0	number of input signals
numOutputs	integer	1 24	both	none	linear	none	none	0	number of outputs
output.N/gain	decimal	0.0 127.0	both	scheduler	linear	gain	midi	0	output gain for each output
predelay	decimal	0.000 1000.000	both	scheduler	linear	none	none	0	Predelay for incoming signals
t60	decimal	0.000 60.000	both	scheduler	linear	none	none	0	t60 reverb time
view/freeze	boolean	0 1	none	none	linear	none	none	0	Turn off the updating of user interface elements when parameters change. This may be done to conserve CPU resources.
view/highlight	string	N/A	none	none	linear	none	none	0	Highlight the modul with a color tint such as red, green, or similar.

Messages

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace	/dataspace /unit/native	/repetitions /allow	/description
documentation/generate	generic	0.000000 1.000000	none	none	none	none	none	1	Generate a html documentation page for this module and save it to disk. The argument 'tex' creates a LaTeX document.
init	none	N/A	none	none	none	none	none	1	Initialize a module completely to the original state.
output.N/active	boolean	0 1	none	none	none	none	none	0	enables and disables a specific output
preset/clear	none	N/A	none	none	linear	none	none	1	Clears all presets, providing a blank slate for saving new pre-sets.
preset/copy	array	N/A	none	none	linear	none	none	1	Create a new preset (2nd argument) by copying the contents of another preset (1st argument)
preset/default	none	N/A	none	none	linear	none	none	1	Open the default preset file and recall the first preset in that file.
preset/dump	none	N/A	none	none	linear	none	none	1	Dump all preset names.
preset/interpolate	array	N/A	none	none	linear	none	none	0	Interpolate between two named presets (argument 1 and 2) using a ratio (float in the range [0.0, 1.0]) specified as the third argument.
preset/mix	array	N/A	none	none	linear	none	none	0	Mix list of pairs of (preset name, mix value) using a ratio (float).
preset/post	none	N/A	none	none	linear	none	none	1	Post all presets to the Max window.
preset/read	generic	0.000 1.000	none	none	linear	none	none	1	Open an xml-preset file and recall the first preset in that file. An optional argument defines the file to open.
preset/recall	generic	0.000 1.000	none	none	linear	none	none	1	Recall a preset by number - you can also choose presets from the module menu.
preset/store	array	N/A	none	none	linear	none	none	1	Store a preset by number in memory. All presets present in memory will be written to disk when you send a /preset/write message to the module.

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace /unit/native	/repetitions /allow	/description
preset/storecurrent	none	N/A	none	none	linear	none	1	Store on the last recalled or stored preset
preset/storenext	none	N/A	none	none	linear	none	1	Store a preset in the next preset slot. Handy so that you do not need to specify a preset number manually.
preset/write	generic	0.000 1.000	none	none	linear	none	1	Write an xml-preset file to disk. An optional argument defines the file to open.
preset/writeagain	none	N/A	none	none	linear	none	1	Write on same xml-preset file.
view/color/border	array	N/A	none	none	linear	none	0	The border color of the module in the format RGBA where values range [0.0, 1.0].
view/color/contentBackground	array	N/A	none	none	linear	none	0	The background color of the module in the format RGBA where values range [0.0, 1.0].
view/color/toolbarBackground	array	N/A	none	none	linear	none	0	The background color of the module's toolbar in the format RGBA where values range [0.0, 1.0].
view/color/toolbarText	array	N/A	none	none	linear	none	0	The color of the module's toolbar text in the format RGBA where values range [0.0, 1.0].
view/internals	none	N/A	none	none	linear	none	1	Attempts to open the internal algorithm for viewing. This works for most modules. Some modules may choose to cloak the algorithms - preventing this message from functioning.
view/panel	none	N/A	none	none	linear	none	0	Open an a module's control panel (inspector) if one is present.
view/refresh	none	N/A	none	none	linear	none	1	Update displayed values for module to reflect current state.
view/script	generic	0.000 1.000	none	none	linear	none	1	Low-level module hacking. Any arguments to this message will be interpreted as patcher scripting for the top-level patcher of the module.
view/size	array	N/A	none	none	linear	none	0	The size of the module's UI.

Returns

/name	/type	/range /bounds	/range /clipmode	/ramp /drive	/ramp /function	/dataspace /unit/native	/repetitions /allow	/description
fdn/length	decimal	0.000000 1.000000	none		time	ms	0	Sum of Delays inside the FDN

Bibliography

- [1] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 943–950, 1979.
- [2] S. Bech, "Spatial aspects of reproduced sound in small rooms," *J. Acoust. Soc. Am.*, vol. 103, pp. 434–445, 1998.
- [3] J. M. Berman, "Behaviour of sound in a bounded space," *J. Acoust. Soc. Am.*, vol. 57, pp. 1275–1291, 1975.
- [4] J. Braasch, N. Peters, and D. L. Valente, "A loudspeaker-based projection technique for spatial music applications using virtual microphone control," *Computer Music Journal*, vol. 32, no. 3, pp. 55–71, 2008.
- [5] L. Cremer and H. A. Müller, "Principles and Applications of Room Acoustics, (translated by T. J. Schultz)," *Applied Science Publishers*, vol. 1, pp. 17–19, 1982.
- [6] J. Dattorro, "Effect design," *J. Audio Eng. Soc.*, vol. 45, pp. 660–684, 1997.
- [7] J. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. of the 90th AES Convention, Preprint 3030*, Paris, France, 1991.
- [8] T. Lossius, P. Baltazar, and T. de la Hogue, "DBAP - Distance-Based Amplitude Panning," in *Proc. of the International Computer Music Conference*, Montreal, Canada, 2009.
- [9] G. Martin, W. Woszczyk, J. Corey, and R. Quesnel, "Controlling phantom image focus in a multichannel reproduction system," in *107th AES Convention, Preprint 4996*, New York, US, 1999.
- [10] R. Pellegrini, "Perception-based design of virtual rooms for sound reproduction," in *22nd AES International Conference, Preprint 000245*, Espoo, Finland, 2002.
- [11] N. Peters, J. Braasch, and S. McAdams, "Sound spatialization across disciplines using virtual microphone control (ViMiC)," *Journal of Interdisciplinary Music Studies (JIMS)*, vol. 5, no. 2, 2011.
- [12] N. Peters, S. Ferguson, and S. McAdams, "Towards a Spatial Sound Description Interchange Format (SpatDIF)," *Canadian Acoustics*, vol. 35, no. 3, pp. 64 – 65, September 2007.
- [13] N. Peters, T. Lossius, and J. C. Schacher, "The Spatial Sound Description Interchange Format – principles, specification, and examples," *Computer Music Journal*, vol. 37, no. 1, pp. 11–22, 2013.
- [14] N. Peters, T. Matthews, J. Braasch, and S. McAdams, "Spatial Sound Rendering in Max/MSP with ViMiC," in *Proc. of the International Computer Music Conference*, Belfast, UK, 2008, pp. 755–758.
- [15] T. Place and T. Lossius, "Jamoma: A modular standard for structuring patches in Max," in *Proc. of the International Computer Music Conference 2006*, New Orleans, US, 2006, pp. 143–146. [Online]. Available: www.jamoma.org
- [16] T. Place, T. Lossius, and N. Peters, "The Jamoma Audio Graph Layer," in *Proc. of the 13th Int. Conference on Digital Audio Effects*, Graz, Austria, 2010.
- [17] T. D. Rossing, Ed., *Springer Handbook of Acoustics*. Berlin, Germany: Springer-Verlag, 2007.
- [18] M. Wright and A. Freed, "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers," in *Proc. of the International Computer Music Conference*, Thessaloniki, Greece, 1997, pp. 101–104.