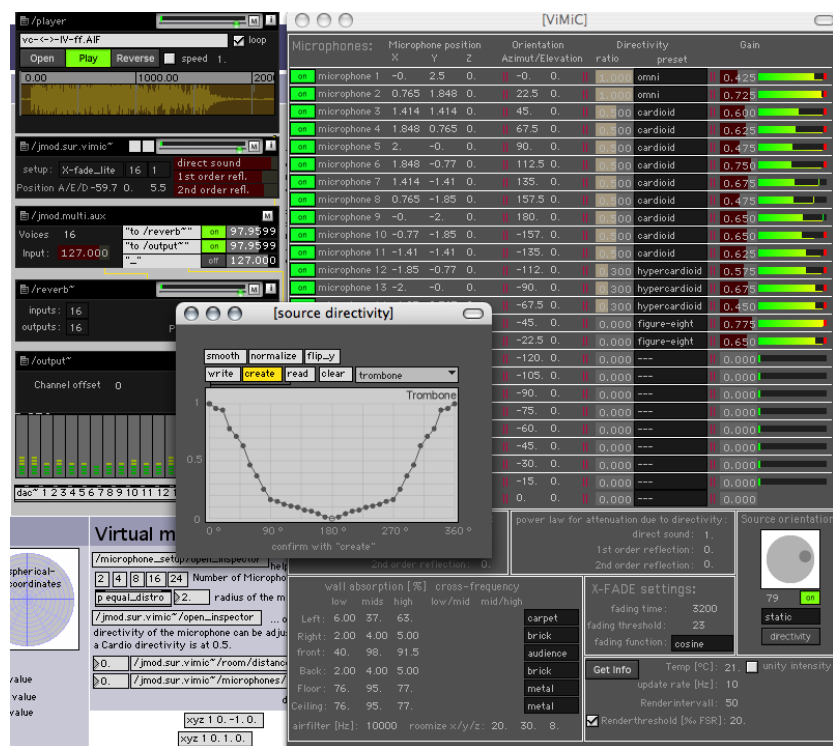


## DOCUMENTATION

# ViMiC - Virtual Microphone Control for Max/MSP



*Documentation  
written by:*  
Nils PETERS

*Implemented by:*  
Jonas BRAASCH  
Tristan MATTHEWS  
Nils PETERS

Draft Version, February 5, 2008

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Requirements . . . . .	2
1.2	Installation . . . . .	2
<b>2</b>	<b>ViMiC</b>	<b>3</b>
2.1	Principles . . . . .	3
2.1.1	Source - Microphone Relation . . . . .	3
2.1.2	Late Reverb . . . . .	5
2.2	Moving Sources . . . . .	6
2.2.1	Rendering with Doppler effect . . . . .	6
2.2.2	Rendering without Doppler effect . . . . .	6
<b>3</b>	<b>jmod.sur.ViMiC</b>	<b>7</b>
3.1	Introduction . . . . .	7
3.1.1	Connecting ViMiC . . . . .	8
3.2	Parameter & Controlling . . . . .	10
3.2.1	Front-end . . . . .	10
3.2.2	Inspector Interface . . . . .	14
3.2.3	I - Microphone configuration . . . . .	15
3.2.4	II - Room Model Parameter . . . . .	18
3.2.5	III - Source Directivity . . . . .	20
3.2.6	IV - X-fade setting . . . . .	22
3.2.7	V - Miscellaneous parameters . . . . .	23
3.3	Rendermodes . . . . .	25
3.3.1	ViMiC_XL . . . . .	25
3.3.2	X_fade_XL . . . . .	25
3.3.3	ViMiC_lite & X_fade_lite . . . . .	25
3.3.4	Panning . . . . .	25
3.3.5	Comparison . . . . .	25
<b>4</b>	<b>Reverb</b>	<b>27</b>
4.1	Introduction . . . . .	27
4.2	Parameter & Controlling . . . . .	28
4.2.1	Front-end . . . . .	28

---

4.2.2	Inspector window . . . . .	30
<b>5</b>	<b>Speakermatcher</b>	<b>32</b>
<b>A</b>	<b>Coordinate System</b>	<b>34</b>
<b>B</b>	<b>OSC namespace</b>	<b>35</b>
<b>C</b>	<b>Known Issues</b>	<b>36</b>

# Chapter 1

## Introduction

### 1.1 Requirements

ViMiC requires:

- Max/MSP 4.6.3 or newer
- MacOS 10.4.11 or newer
- Ambimonitor, part of the Jamoma distribution  
downloadable as part of the *ICST Ambisonics Tools for MaxMSP*:  
[www.icst.net/downloads](http://www.icst.net/downloads)
- ej.function, part of *ejies*,  
downloadable from [www.e--j.com/](http://www.e--j.com/)

### 1.2 Installation

ViMiC is available in the subversion repository of Jamoma:

```
https://jamoma.svn.sourceforge.net/svnroot/jamoma/  
branches/active
```

A tutorial to accessed the Jamoma SVN is posted at: [www.jamoma.org/wiki/FrontPage](http://www.jamoma.org/wiki/FrontPage)

For installing Jamoma after downloading, following the advises in the included *Readme.html*.

Unzip ViMiCMax .mxo.zip, located in */UserLib/ViMiC/jmod.sur.vimic/*.

The ViMiC toolbox is located in the folder */UserLib/ViMiC/*. To test if everything is installed correctly, just load the help-patch *jmod.sur.vimic .help*, located at */UserLib/ViMiC/jmod.sur.vimic/*.

# Chapter 2

## ViMiC

### 2.1 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.

#### 2.1.1 Source - Microphone Relation

Sound sources and microphones can be placed and moved in 3D as desired. Figure 2.2 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. 2.1 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.

$$g_i = \frac{1}{d_i^q} \quad d \geq 1 \quad (2.1)$$

Further attenuation happens through the chosen microphone characteristic and source directivity (see Fig. 2.3). For all common microphone characteristics, the directivity for a certain angle of incidence  $\delta$  can be imitated by calculating Eq. 2.2 and applying a set of microphone coefficients from Table 2.1.1. 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.

$$\Gamma = (a + b \cdot \cos \delta)^w \quad 0 \leq a, b \leq 1 \quad (2.2)$$

Characteristic	a	b	w
Omnidirectional	1	0	1
Subcardioid	0.7	0.3	1
Cardioid	0.5	0.5	1
Supercardioid	0.33	0.67	1
Hypercardioid	0.3	0.7	1
Figure-of-8	0	1	1

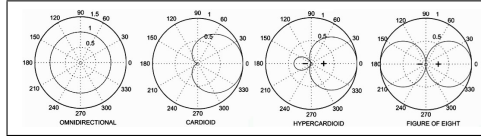


Figure 2.1: Common microphone characteristics

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  $1^\circ$  accuracy.

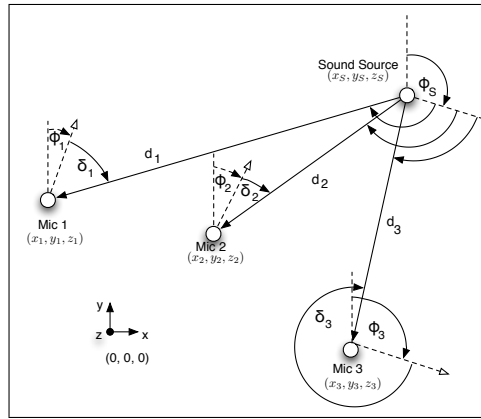


Figure 2.2: Geometric example

### 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 Pellegrini [2002]. Early reflections are strong auditory cues in encoding the sound

source distance. According to virtual room size and position of the microphones, adequate early reflections are rendered in 3D through the well-known image method Allen & Berkley [1979]. Each image source is rendered according to the time of arrival, the distance attenuation, microphone characteristic and source directivity, as described in section 2.1.1. 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 ( $8 \text{ microphones} \cdot [6 \text{ early reflections} + 1 \text{ direct sound path}]$ ). Although time delays are efficiently implemented through a shared multi-tap delay line, this processing can be computationally intensive.

### 2.1.2 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 Jot & Chaigne [1991] 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. 2.3), 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 4 on page 27.

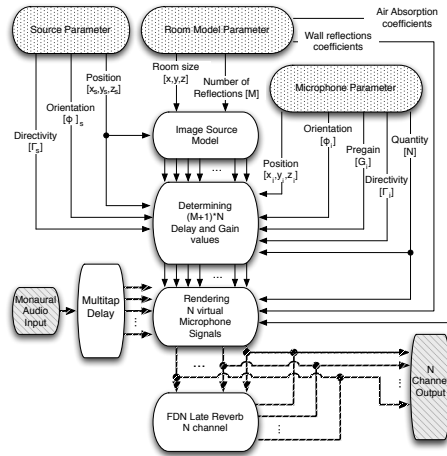


Figure 2.3: Flowchart of the Max/MSP processing

## 2.2 Moving Sources

In Figure 2.4 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.

### 2.2.1 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  $f_s = 44.1$  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.

### 2.2.2 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.

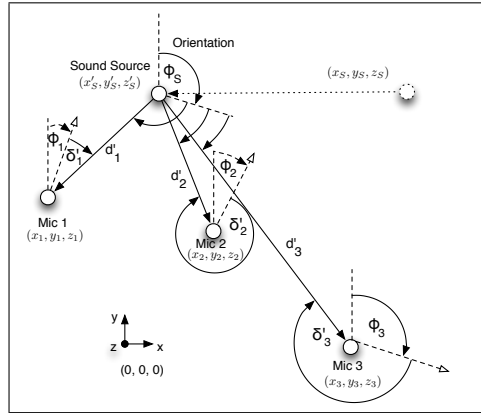


Figure 2.4: Moving sources, geometric example



## Chapter 3

# The Jamoma module

**jmod.sur.vimic~.mxt**

### 3.1 Introduction

For easier and more flexible controllability, ViMiC was structured as high-level modules using the Jamoma<sup>1</sup> framework for Max/MSP/Jitter Place & Lossius [2006]. Jamoma offers a clear advantage in its standardization of presets and parameter handling. Each ViMiC parameter has been sorted into three primary namespaces: *source*, *microphone* and *room*; and has a specified data range. ViMiC is fully controlled through a GUI and through external OSC-messages. Wright & Freed [1997]. Jamoma modules can be loaded in two ways.

#### Loading ViMiC as an ordinary object

1. Create a new empty Max patch.
2. Create a new object box, and type in  
`jmod.sur.vimic~.mxt \test`  
This should create an object with two inlets and two outlets, similar to Figure 3.1. The module's OSC name assigned to this object is `\test`.
3. Double-click the object, to open a small window, offering an interface for the object.
4. Click on the upper left circle to close the interface.

#### Loading ViMiC in a bpatcher

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

---

<sup>1</sup><http://www.jamoma.org>




Figure 3.1: Loading ViMiC as an ordinary object

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~.mxt`
4. Assign a unique OSC name to the module by giving an argument, e.g. `/test`, see Figure 3.2.
5. Close the bpatcher inspector.
6. The created bpatcher has two inlets and two outlets.

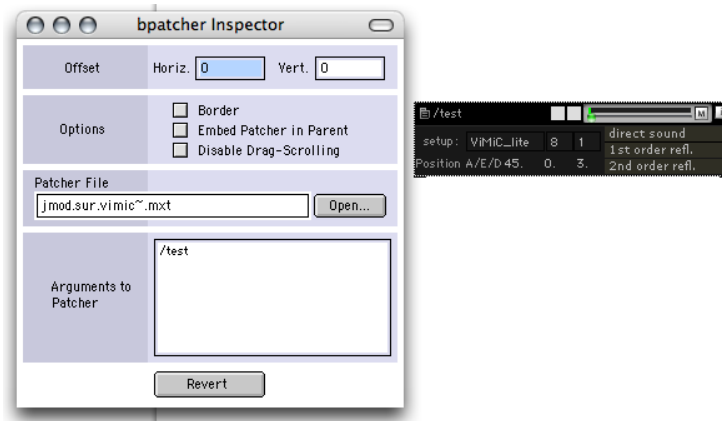


Figure 3.2: Loading ViMiC in a bpatcher

In terms of functionality, both display options provide the same kind of functionality, it is just a matter of taste or space in the Max patch what display mode is preferred.

### 3.1.1 Connecting ViMiC

#### Audio connection

The ViMiC module has two inlets and two outlets. The right inlet takes the audio signal to be processed in ViMiC, and the right outlet delivers the processed multichannel audio in form of a multicable. 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 3.3 a simple noise signal is spatialized with ViMiC, and the processed multichannel audio signals are connected to the Jamoma module `jmod.sur.output~.mxt`. The latter distributes of a multicable signal to multiple `dac~` channels.

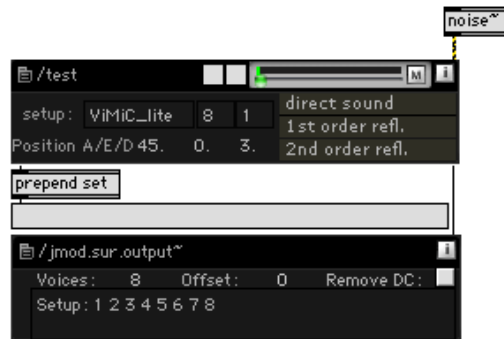


Figure 3.3: Here the multicable output is connected to `jmod.sur.output~.mxt`

### OSC-message connection

As usual in Jamoma modules, the left inlet and the left outlet are assigned 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 at the left outlet. In Figure 3.3 the left outlet is connected to a message box in order to display the last parameter change.

## 3.2 Parameter & Controlling

### 3.2.1 Front-end

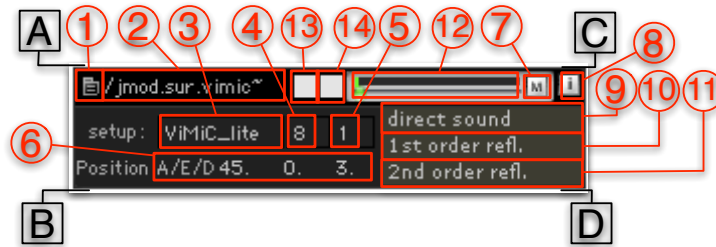


Figure 3.4: ViMiC main control interface

Identifier	Name	Description
A	Message inlet for OSC-commands	p.9
B	Message outlet for OSC-commands	p.9
C	Signal inlet	p.9
D	Multichannel signal outlet	p.9
1	Module pop-up menu	p.10
2	Modules OSC name	p.11
3	Rendermodes pop up menu	p. 11
4	Number of pre-configured microphones	p.11
5	Order of rendered early reflections	p.11
6	Sound Source Position	p.12
7	Mute button	p.12
8	Inspector button	p.12, p.14
9	Gain control for the direct sound component	p.13
10	Gain control for the 1st-order early reflections	p.13
11	Gain control for the 2nd-order early reflections	p.13
12	Gain control for the output volume	p.13
13	Warning button	p.13
14	Report button	p.14

In Figure 3.4 the front end of the module is shown. Here the most often used parameter are shown. The following section gives information about each controllable parameter.

#### (1) - The module pop-up menu

The module pop-up menu provides several Jamoma specific option. Reader are pointed to the Jamoma documentation.  
related OSC-messages:

```
ui/refresh  
ui/freeze <1,0>
```

### (2) - The modules OSC name

Here the given modules OSC-name appears. It identifies the module to its namespace. This name becomes necessary if a OSC-message is not send directly to the module's left inlet, but the module is controlled remotely e.g. through the `jmod.cuelist.mxt` module. The module OSC name has to be unique. In case of multiple used module name, Jamoma will rename the modules to a unique identification 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
- ViMiC\_lite
- X\_fade\_XL
- X\_fade\_lite
- Panning
- Static

Detailed information to the different render methods refer to section 3.3 on page 25. equivalent OSC-message:

```
/rendermode <string>
```

### (4) - Number of pre-configured microphones

In a drop down menu the maximum number of virtual microphones can be selected. A change is only performed 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 the rendered early reflections can be determined. The higher the chosen number, the more early reflections are rendered. A change is only performed if the DSP is turned off. Per microphone the following number of reflections are processed:

Order	Reflections
0	0
1	6
2	18

equivalent OSC-message:

```
/room/reflection/order <int> [0,1,2]
```

#### (6) - Sound Source Position

The coordinate triplet defines the position of the sound source in the virtual room. Although the position is displayed in spherical coordinates (azimuth, elevation, distance), according to the SpatDIF standardization Peters et al. [2007] a position can be defined either in spherical or cartesian coordinates through external OSC-messages. Convention for the coordinate system can be found in appendix A.

related OSC-messages:

```
/source/position/absolute/aed <azimuth elevation distance>
/source/position/absolute/a <azimuth-coordinate>
/source/position/absolute/e <elevation-coordinate>
/source/position/absolute/d <distance-coordinate>

/source/position/absolute/xyz <x y z>
/source/position/absolute/x <x-coordinate>
/source/position/absolute/y <y-coordinate>
/source/position/absolute/z <z-coordinate>
```

#### (7) - The mute button

If mute is active, the ViMiC stops rendering, but still calculates the geometric model, so if the module is unmuted, the virtual sound source appears at the most updated position.

equivalent OSC-message:

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

#### (8) - The inspector button

By clicking on the [i] button in the upper right corner, ViMiC's inspector window opens to give control over other parameter described in 3.2.2.

equivalent OSC-message:

```
/open_inspector
```

**(9) - Gain control for the direct sound component**

This horizontal fader controls the gain of the direct sound component at each virtual microphone. The gain controller were programmed by using Jamoma's `jcom.parameter.gain` object. Therefore there are different ways to control the gain value. related OSC-message:

```
/room/reflection/0/gain/midi <midi value> [0.0 .. 127.0]
/room/reflection/0/gain <db value>        [-77.0 .. 0.0]
```

At the moment the midi value 127.0 is similar to 0.0 dB which refers to a linear gain of 1.0.

**(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. related OSC-message:

```
/room/reflection/1/gain/midi <midi value> [0.0 .. 127.0]
/room/reflection/1/gain <db value>        [-77.0 .. 0.0]
```

**(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. related OSC-message:

```
/room/reflection/2/gain/midi <midi value> [0.0 .. 127.0]
/room/reflection/2/gain <db value>        [-77.0 .. 0.0]
```

**(12) - Gain control for the output volume**

The horizontal slider controls the output volume of the module. parameter handling is equivalent to gain controller Nr. (9), (10) and (11). In order to boost the output, gain values can be bigger than 0 dB.

related OSC-message:

```
/audio/gain/midi <midi value> [0.0 .. 157.0]
/audio/gain/ <db value>        [-77.0 .. 18.0]
```

**(13) - 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]
```

**(14) - the report button**

If this button is ticked, every parameter change will cause postings to the Max Window.

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

**3.2.2 Inspector Interface**

Through clicking on the inspector button, or by sending the OSC message `/open_inspector` to the module the Inspector interface appears (see Figure 3.5). The biggest part of the inspector window is filled with the channel strip for each virtual microphone (I), which functionality is explained in section 3.2.3 on the following page.

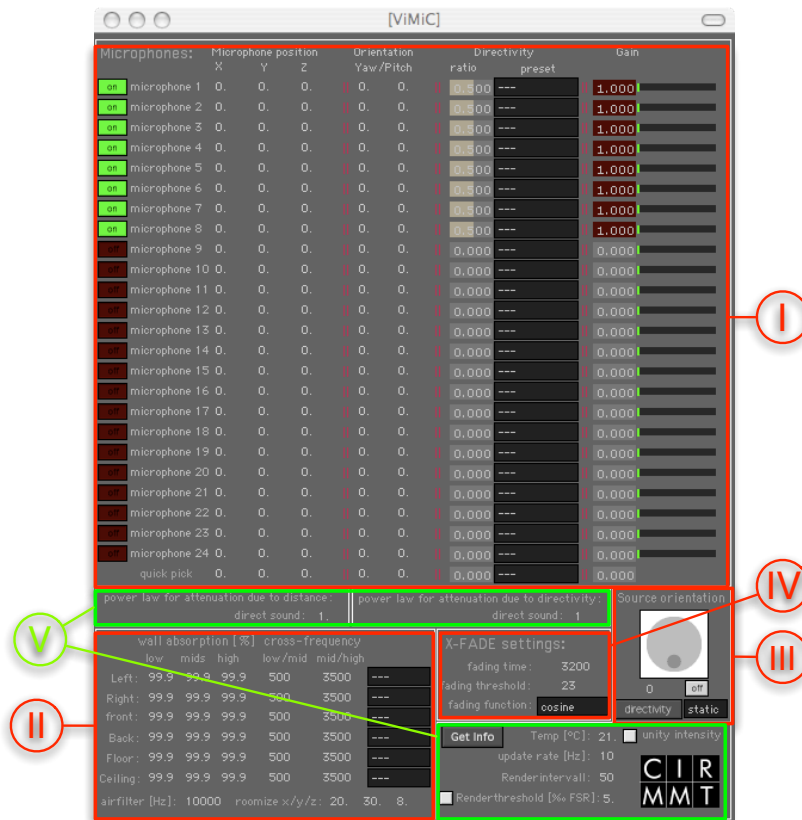


Figure 3.5: ViMiC inspector window



Identifier	Name	Description
I	Channel strips for each microphone	p.15
II	Room model parameters	p.18
III	Source directivity settings	p.20
IV	Settings for X-Fade Render-method	p.22
V	Miscellaneous parameters	p.23

### 3.2.3 I - Microphone configuration

Figure 3.6 shows a part of the microphone configuration area. Like a channel strip in a mixing board, each strip has the same functionality. Therefore only one channels strip example is discussed here. In order to access a specific microphone channel with an OSC-message, displayed # in the OSC-namespace, has to be replaced by the number of the virtual microphone (1-24). The setting of (4) in the front-end (see page 11) determines the highest number a virtual microphone can have. So If (4) is set to eight, only the first eight microphones are used. It is possible to change setting of higher numbered microphones. These settings don't get lost! If (4) is changed later in order to render more virtual microphones, this settings will be activated.

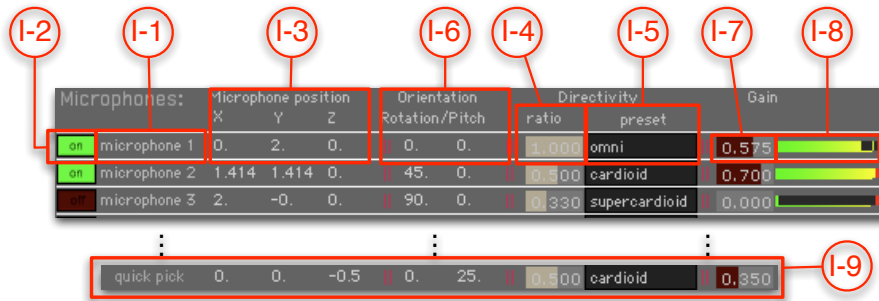


Figure 3.6: Channel Strip of Virtual Microphone

Identifier	Name	Description
I-1	Channel number	p.16
I-2	Active button	p.16
I-3	Microphone position	p.16
I-4	Microphone directivity ratio	p.16
I-5	Microphone directivity presets	p.17
I-6	Microphone orientation	p.17
I-7	Microphone pre-gain	p.17
I-8	VU-meter	p.17
I-9	Quick pick configuration	p.18

**(I-1) - Channel number**

This displays the number of the related virtual microphone. There is nothing to edit here. The number of the virtual microphone is directly related to the output channel number. So 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 off (it will turn red), the pre-gain volume of this virtual microphone to zero. If the active button is reset to on (it turns green), the pre-gain is ramped from zero to the previous gain value in order to avoid clicks.  
related OSC-messages<sup>1</sup>:

```
/microphones/#!/mute <int> [0,1]
```

**(I-3) - Microphone position**

The position of a microphone is set in cartesian coordinates. However, the module `jmod.speakermatcher.mxt` can be used to place the microphones with the help of a graphical user interface or in form of cartesian as well as spherical coordinate triplets (see section 5, page 32).  
related OSC-messages:

```
/microphones/#!/position/xyz <x-coordinate y-coordinate z-coordinate>
/microphones/#!/position/z <z-coordinate>
```

**(I-4) - Microphone directivity ratio**

As described in section 2.1.1, p. 3 the microphone directivity is defined by the equation:

$$\Gamma = (a + b \cdot \cos \delta)^w \quad 0 \leq a, b \leq 1 \quad (3.1)$$

$$1 = a + b \quad (3.2)$$

The horizontal slider controls the ratio of the parameter  $a$  and  $b$  of this equation. So if the horizontal slider is set to 1.0, the variable  $a$  has the value of 1.0 and the variable  $b$  a value of 0.0.

equivalent OSC-message:

```
/microphones/#!/directivity/ratio <float> [0..1]
```

---

<sup>1</sup>the # in the OSC-namespace has to be replaced with the number of the virtual microphone (1-24) e.g. for the microphone 5: `/microphone/5/mute 1`

**(I-5) - Microphone directivity presets**

Rather than manually setting the directivity of a virtual microphone, this drop down menu serves several predefined microphone directivity pattern. The following pattern are available:

- omni
- subcardioid
- cardioid
- supercardioid
- hypercardioid
- figure-eight

equivalent OSC-message:

```
/microphones/##/directivity/preset <string>
```

**(I-6) - Microphone orientation**

A microphone can point towards a certain direction. This feature becomes important when a virtual microphone is designed with a specified directivity. Yaw-angle (-180.0° .. 180.0°) and pitch-angle (-90.0° .. 90.0°) can be set.

```
/microphones/##/orientation <yaw-angle pitch-angle>
/microphones/##/orientation/yaw <yaw-angle> [-180.0 .. 180.0]
/microphones/##/orientation/pitch <pitch-angle> [-90.0 .. 90.0]
```

**(I-7) - Microphone pre-gain**

This option can be used to modify the sensitivity of the microphones separately from each other.

related OSC-message:

```
/microphones/##/gain/midi <midi value> [0.0..127.0]
/microphones/##/gain <db value> [-77.0..0.0]
```

**(I-8) - VU-meter**

The VU-meter shows the signal energy after the output volume control. In other words, it is a classical AFL-VU-meter. The integration time is around 300 ms. To save some DSP, the VU-meter can be switched off either due to ticking the entry *disable Signal meters* in the modules pop-up menu (see (1), on page 10).

related OSC-messages:

```
/audio/meters/freeze <int> [0,1]
audio/meters/clear
```

**(I-9) - Quick pick configuration**

Sometimes it is not necessary to configure certain microphone parameter differently for each microphones. In this case the quick pick configuration eases the microphone set up: Each modification of one of these quick pick parameter affects all microphones in the same way. For example, if all microphones should have a cardioid directivity, rather than configuring each microphone separately, just select the cardioid directivity from the quick pick area.

**3.2.4 II - Room Model Parameter**

If early reflections were selected in the front-end menu (see (5) on p.11), the room model becomes active. Early reflections are calculated according the the position of the sound source, the microphones and the room model (see 2.1.1 on page 4). Here the controllable room parameter are explained briefly.

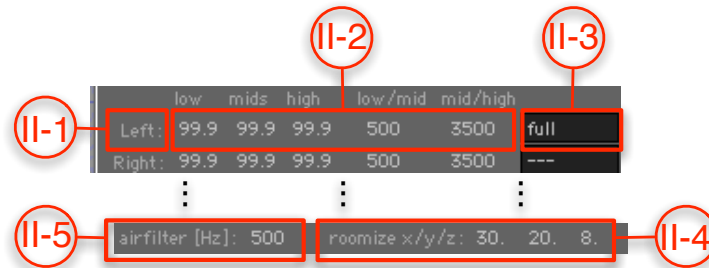


Figure 3.7: Room model parameter

Identifier	Name	Description
II-1	Wall Identifier	p.18
II-2	Absorption parameter	p.18
II-3	Absorption preset	p.19
II-4	Room size	p.19
II-5	Air absorption filter	p.19

**(II-1) - Wall Identifier**

Tells the name of the wall where the parameter on the right belong to.

**(II-2) - Absorption parameter**

The absorptive behavior of the room boundaries is modeled through a high-mid-low shelf filter. In each band the wall absorption is defined in %. The bigger the number, the higher the absorptive character. The crossover frequencies between the low mid

and high band can be set as well.

```
/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 wildcards ## in the OSC-messages has to be replaced with the identifier of the wall:

- left
- right
- front
- rear
- floor
- ceiling

#### (II-3) - Absorption preset

Several surface presets are available to set the Absorption parameter, as explained above.

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

#### (II-4) - Room size

Here the size of the virtual room is set in width, length and height. The unit is meter. Each dimension is limited to 40 meter.  
related OSC-messages:

```
/room/size/xyz <width length height>
```

#### (II-5) - Air absorption filter

As explained in section 2.1.1 on page 4, the early reflections are filtered to simulate the damping of the air. This 2nd-order lowpass filters all early reflections in the same way. The cut-off frequency can be set through:

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

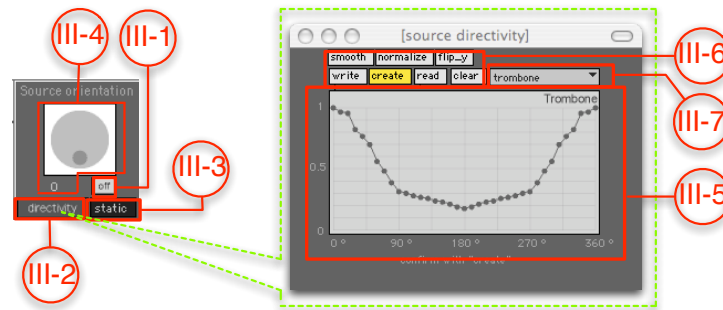


Figure 3.8: Source Directivity options

### 3.2.5 III - Source Directivity

Identifier	Name	Description
III-1	Source directivity activator	p.20
III-2	Source Directivity Inspector button	p.20
III-3	Directivity mode	p.20
III-4	Orientation wheel	p.21
III-5	Directivity editor	p.21
III-6	Editor tools	p.21
III-7	Directivity presets	p.21

#### (III-1) - Source directivity activator

With that toggle, the source directivity calculation can be switched on/off.

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

#### (III-2) - Source Directivity Inspector button

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

```
/ open_directivity_editor
```

#### (III-3) - Directivity mode

There are three directivity modes implemented:

1. static
2. center
3. follow

**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 (III-4)).

**center:** During a movement, the source is oriented automatically towards the center.

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

```
/source/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 (means the 0° direction) of the sound source is pointing. The Orientation is defined in the range of [-180° .. 180°].

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

#### (III-5) - Directivity editor

Here, the directivity of a sound source can be freely designed with the help of the mouse. It works like a break point function editor. A line can be transformed into a curve by pressing the "alt" key while dragging the line with the mouse.

To confirm an edited source directivity, the yellow "create" button (see III-6) has to be clicked.

#### (III-6) - Editor tools

Some 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\_x:** horizontally flip of the directivity curve at the 0.5 line.

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

**create:** has to be clicked in order to confirm the edited curve.

**write:** to save a curve in a separate text file.

**read:** to load a directivity from a extern file.

```
/source/directivity/load_file <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.

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

### 3.2.6 IV - X-fade setting

The following parameter are specific for the "x\_fade" render methods. More information for the "x\_fade" render methods can be found in section 2.2.2 on page 6 and in section 3.3 on page 25. These parameter can only be changed if the sound source is not moving.

#### (IV-1) Fading Threshold

This parameter sets the length in samples, one of the calculated propagation paths has to change, in order to trigger a cross-fade from the current position to a new position.

```
/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 which is set in IV-2. The unit of this parameter is samples.

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

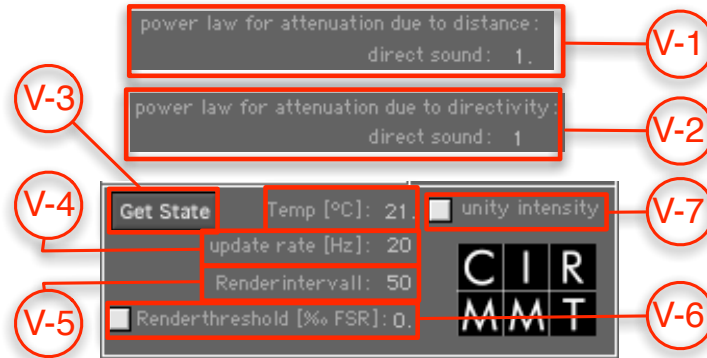
#### (IV-3) Fading Function

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

```
/rendermode/xfade/fadefunction <string>
```





### 3.2.7 V - Miscellaneous parameters

Identifier	Name	Description
V-1	Distance law exponent	p.23
V-2	Microphone Directivity exponent	p.23
V-3	"Get State" button	p.24
V-4	ViMiC update rate	p.24
V-5	ViMiC Renderinterval	p.24
V-6	Renderthreshold	p.24
V-7	Intensity normalization	p.24

#### (V-1) - Distance law exponent

This parameter sets the exponent  $q$  in the attenuation function described in equation 2.1 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.

/room/distance/power <float> [0.0 .. 8.0]

#### (V-2) - Microphone Directivity exponent

This parameter sets the exponent  $w$  in the attenuation function described in equation 2.2 on page 4. By increasing the exponent  $w$  to a value greater than 1 will produce an artificially sharper directivity pattern. Furthermore, by applying even numbered parameter, anti-phased gain values are avoided. Note, that only integer values are allowed for this exponent.

/microphones/directivity/power <int> [0 .. 8]

**(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 in order to find bugs.

```
\report_all
```

**(V-4) - ViMiC update rate**

To save resources, not every parameter change is causing ViMiC to update (re-render) its virtual microphone signals. This parameter sets the frequency, how often ViMiC is forced to re-render its virtual microphone signals. All messages arrive the ViMiC-external as fast as possible, independently from this update frequency.

```
/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:

```
/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.

```
/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.

```
/rendermode / threshold / active <int> [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 3.3. This is an experimental option.

$$I = \sum_{i=1}^n g_i^2 = 1 \quad (3.3)$$

```
/rendermode/normalization/active <int> [0,1]
```

### 3.3 Rendermodes

ViMiC supports several different Rendermodes which can be chosen from the drop-down menu in the front-end (see (3) on page 11). The following section describes each of them:

#### 3.3.1 ViMiC\_XL

If a sound moves, ViMiC\_XL renders the sound rays with through delay-line interpolation, which causes an audible doppler effect (see section 2.2.1 on page 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 3.2.4 on page 18.

#### 3.3.2 X\_fade\_XL

X\_fade\_XL renders moving sound sources with the cross-fade method, described in section 2.2.2 on page 6. The special X-Fade settings in the ViMiC inspector (see section 3.2.6 on page 22) come into play. 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.

#### 3.3.3 ViMiC\_lite & X\_fade\_lite

The render methods ViMiC\_lite and X\_fade\_lite differs from their bigger brothers ViMiC\_XL and X\_fade\_XL in the quality of the room model: Instead of having the choice of a individual wall absorption parameter for each wall, the entire virtual room has the same kind of wall absorption. To edit the wall absorption, the parameters of the "left" wall in the ViMiC inspector (the first wall in the ViMiC Room settings see 3.2.4 on page 18) have to be manipulated.

#### 3.3.4 Panning

The Panning 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. Regarding of what order of reflections was selected in (5), no early reflections are calculated, because the reflections would appear undelayed. If the gain normalization option is active (V-7), and microphone characteristic is set to omni, DBAP-like (distance based amplitude panning) Lossius [2007] panning is created.

#### 3.3.5 Comparison

Rendermode	Source is...	No Reflections		1st-order Reflections		2nd-order Reflections	
		8 mics	24 mics	8 mics	24 mics	8 mics	24 mics
ViMiC_XL	stationary	7.4%	14.6%	15.5%	40.7%	30.6%	+++
	moving	8.6%	18.3%	23.4%	63.4%	50.6%	+++
X_fade_XL	stationary	7.6%	15.6%	17.3%	45.5%	35.8%	+++
	moving	8.2%	16.4%	20.9%	54.6%	43.4%	+++
ViMiC_lite	stationary	7.2%	214.4%	10.8%	25.5%	15.4%	40.0%
	moving	8.3%	17.6%	17.4%	44.9%	33.2%	+++
X_fade_lite	stationary	8%	16.5%	13.5%	34.0%	22.5%	60.9%
	moving	8.2%	17.0%	14.0%	34.6%	22.2%	58.3%
Panning	stationary	6.4%	12.2%	-	-	-	-
	moving	6.5%	12.4	-	-	-	-

Table 3.1: Max/MSP Processor load for one sound source at different render settings, (measured at mac mini 1.66 GHz intel core due)

## Chapter 4

# The Jamoma module

**jmod.sur.reverb~.mxt**

### 4.1 Introduction

The late reverb is implemented as a feedback delay network (FDN) with 16 delay lines, displayed in Figure 4.1. The use of FDN for artificial reverberation can be justified fundamentally by a stochastic model of late reverberation decays assuming that sufficient density of acoustic model in the frequency domain and of reflections in the time domain are achieved. Under these assumptions, later reflections can be modeled as a Gaussian exponentially decaying random process, characterized by a spectral envelope. This approach cannot guarantee the same degree of accuracy a convolution with a measured impulse response, but provide a more efficient parametrization for dynamic control of the synthetic reverberation effect like reverberation time, modal density or reverb coloration.

Any FDN can be represented as a set of digital delay lines whose inputs and outputs are connected by a feedback/mixing matrix which defines the FDN structure. 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 (see Figure 4.2). The advantage of this matrix is that it can be efficiently implemented with a butterfly algorithm using  $N \log 2N = 64$  additions. As proposed in Dattorro [1997] the delay-lines in the FDN can be modulated by low frequency oscillators in terms of delay-length and attenuation coefficient.

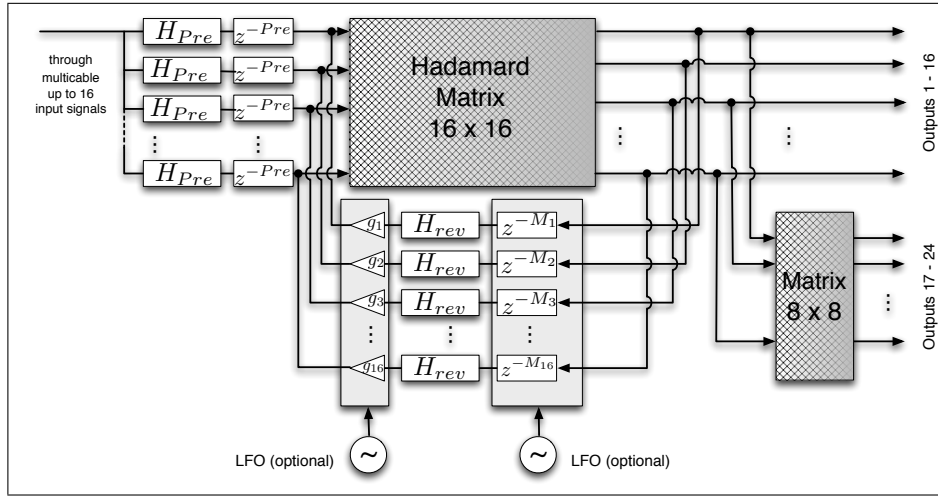


Figure 4.1: FDN-reverb flowchart

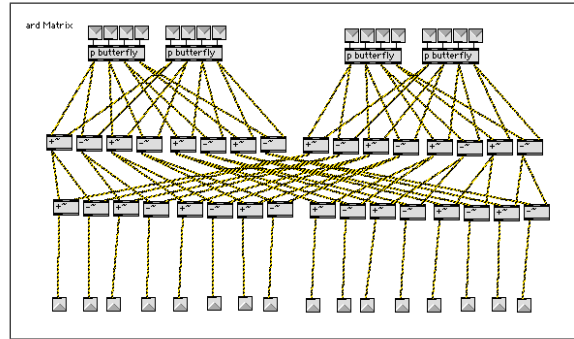
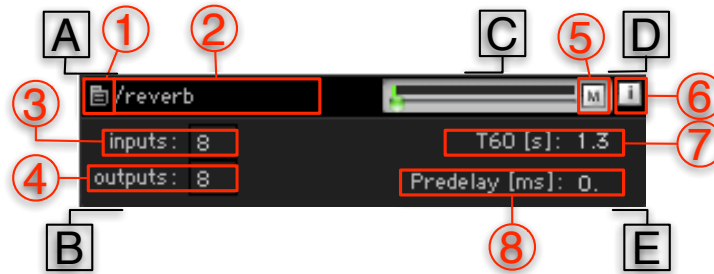


Figure 4.2: FDN Hadamard matrix

## 4.2 Parameter & Controlling

### 4.2.1 Front-end



Identifier	Name	Description
A	Message inlet for OSC-commands	p.29
B	Message outlet for OSC-commands	p.29
C	Multichannel signal inlet	p.29
D	Signal inlet	p.29
E	Multichannel signal outlet	p.29
1	Module pop-up menu	p.29
2	Modules OSC name	p.29
3	Number of input channels	p.29
4	Number of output channels	p.29
5	Mute button	p.29
6	Inspector button	p.29
7	Reverb time $T_{60}$	p.29
8	Pre-Delay	p.30

**Reverb Connections**

NOT WRITTEN YET

**Module pop-up menu**

NOT WRITTEN YET

**Modules OSC name**

NOT WRITTEN YET

**Number of input channels**

NOT WRITTEN YET

**Number of output channels**

NOT WRITTEN YET

**Mute button**

NOT WRITTEN YET

**Inspector button**

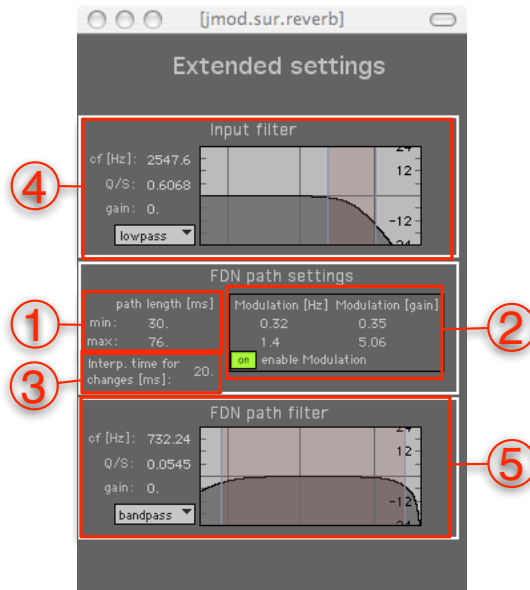
NOT WRITTEN YET

**Reverb time  $T_{60}$** 

NOT WRITTEN YET

**Pre-Delay**

NOT WRITTEN YET

**4.2.2 Inspector window**

Identifier	Name	Description
1	Range of the FDN delay lengths	p.30
2	FDN Modulation options	p.30
3	Interpolation time	p.30
4	Pre-filter settings	p.31
5	FDN Filter settings	p.31

**Range of the FDN delay lengths**

NOT WRITTEN YET

**FDN Modulation options**

NOT WRITTEN YET

**Interpolation time**

NOT WRITTEN YET



**Pre-filter settings**

NOT WRITTEN YET

**FDN Filter settings**

NOT WRITTEN YET

## **Chapter 5**

### **The Jamoma module**

**`jmod.speakermatcher.mxt`**

NOT WRITTEN YET

## Bibliography

- J. B. Allen & D. A. Berkley (1979). ‘Image method for efficiently simulating small-room acoustics’. *J. Acoust. Soc. Am.* **65**(4):943 – 950.
- J. Dattorro (1997). ‘Effect design’. *J. Audio Eng. Soc.* **45**:660–684.
- J. Jot & A. Chaigne (1991). ‘Digital delay networks for designing artificial reverberators’. In *90th AES Convention, Preprint 3030*, Paris, France.
- T. Lossius (2007). *Sound Space Body: Reflections on Artistic Practice*. Ph.D. thesis, Bergen National Academy of the Arts.
- R. Pellegrini (2002). ‘Perception-Based Design of Virtual Rooms for Sound Reproduction’. In *22nd AES International Conference, Preprint 000245*.
- N. Peters, et al. (2007). ‘Towards a Spatial Sound Description Interchange Format (SpatDIF)’. *Canadian Acoustics* **35**(3):64 – 65.
- T. Place & T. Lossius (2006). ‘Jamoma: A modular standard for structuring patches in Max’. In *Proceedings of the International Computer Music Conference 2006*, New Orleans, US.
- M. Wright & A. Freed (1997). ‘Open Sound Control: A New Protocol for Communicating with Sound Synthesizers’. *Proceedings of the 1997 International Computer Music Conference* pp. 101–104.

# Appendix A

## Coordinate System

ViMiC uses a right-handed cartesian coordinate system, which limits are defined by the virtual room size (see. 3.2.4).

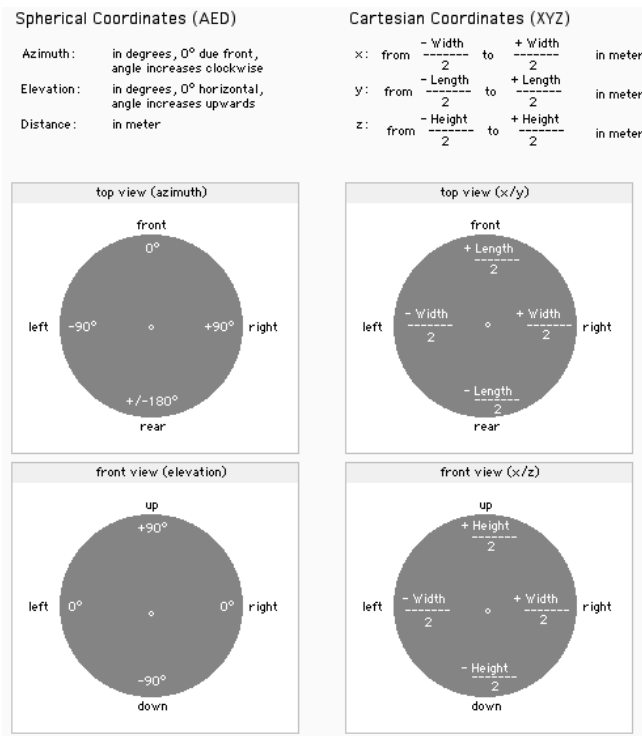


Figure A.1: Coordinate Systems: Spherical (left) and Cartesian (right)

## **Appendix B**

### **OSC namespace**

On this page I want to list the entire OSC namespace for ViMiC. As long this is not written, please refer to the file `jmod.sur.vimic~.html` in the ViMiC directory. If anybody know a working `html2tex` converter, please let me know.

## Appendix C

### 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 bugs and suggestions to

nils\_DOT\_peters\_AT\_mcgill\_DOT\_ca.

#### **jmod.sur.vimic .mxt**

- The first time the inspector opens takes quiet 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 beeing ramped.
- Quick pick configuration (see p.18) for the x-coordinate and y-coordinate is not implemented yet.
- The "follow" option for the source directivity is not implemented yet.

#### **jmod.speakermatcher.mxt**

- The first time the inspector opens takes quiet long time

#### **jmod.sur.reverb .mxt**

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