ambitools Documentation

Pierre Lecomte
pierre.lecomte@gadz.org

August 22, 2016

Contents

1 Introduction

1.1 Goals of ambitools

ambitools is a collection of tools for sound field synthesis using Near-Field Compensated Higher Orders Ambisonics (NFC-HOA). For the rest of this document, the denomination Ambisonics will be used for convenience.

ambitools is developped in the context of my PhD on 3D sound field synthesis. The audio processing is coded in FAUST[1] (Functional AUdio Stream) which allows to produce efficent C++ code and exports in various DSP tools format: VST, standalone applications, LV2, etc. Thus, ambitools is multi-platform (although conceived under Linux/Jack).

The goal of ambitools is mainly to produces several modules to encode, decode and transform 3D synthesized sound field or 3D recordings in a context of physical sound field synthesis.

The project is open-source under GPL licence. ambitools is thought to be flexible to adapt to your configuration. The FAUST code for each tool has an header with several variable that you can change, such as Ambisonic order M, number of sources N, etc. Thus, if you need an encoder up to order $M = 4$, for $N = 5$ sources, you just has to change this two parameters in `hoa_encoder_N_sources.dsp` and produce de required plug-in using FAUSTLIVE[1] for example.

So far, these tools are used to drive the several spherical loudspeakers arrays at Université de Sherbrooke, Québec, Canada and Conservatoire National des Arts et Métiers, Paris, France.

Hopefully, since Ambisonics are flexible, you can use ambitools with binaural rendering and listen to 3D sound without the need of a spherical loudspeaker array!

Don't hesitate to contact me for any suggestions, requirements, critics or even just to tell me you're using ambitools !

Pierre Lecomte

1.2 Ambisonics overview

This section presents the basis of Ambisonics for 3D sound field synthesis. It is written with relatively few maths to focus on the concepts instead of the mathematical rigor, which would be out of scope of this document. If you want more insights to Ambisonics refer for example to [Daniel, 2000, Poletti, 2005, Ahrens, 2012].

1.2.1 Principles

Ambisonics are some mathematical approaches to represent a sound field in two or three dimensions over a basis of spherical harmonics. Without giving more mathematical details, Ambisonic main possibilities are displayed on Fig. 1.

Thus, in accordance to this Fig. 1, ambitools provides tools to:

- Encode a sound field captured by a spherical microphone.
- Encode a sound field from monophonic signals and explicit position in space.
- Transform and manipulate the encoded sound field in Ambisonic domain.
- Decode the sound field over spherical loudspeaker grids or over headphones with binaural convolution.

1.2.2 Spherical coordinate system

Ambisonics are usually described in spherical coordinate systems. In ambitools, the following spherical coordinate system is used and shown in Fig. 2:

$$x = r\cos(\theta)\cos(\delta),\ y = r\sin(\theta)\cos(\delta),\ z = r\sin(\delta). \tag{1}$$

The azimuth angle $\theta \in [0 \quad 360°]$. The elevation angle $\delta \in [-90° \quad 90°]$. The convention for rotation is counterclockwise direction (i.e. trigonometric direction).

1.2.3 Spherical harmonics

As mentionned before, Ambisonics describes a sound field over a spherical harmonics basis. Those functions are directional functions of the variable θ, δ. They are denoted $Y_{mn}(\theta, \delta)$ where subscript m represent the order of the spherical harmonic and n its degree. There is several definition for these functions. In ambitools, the N3D real spherical harmonic definition is used [Daniel, 2000]:

$$Y_{mn}(\theta,\delta) = \sqrt{(2m+1)\epsilon_n \frac{(m-|n|)!}{(m+|n|)!}} P_{m|n|}(\sin(\delta)) \times \begin{cases} \cos(|n|\theta) & \text{if} \quad n \geq 0 \\ \sin(|n|\theta) & \text{if} \quad n < 0 \end{cases}, \tag{2}$$

where $P_{m|n|}$ are the associated Legendre polynomials of order m and degree $|n|$, $(m, n) \in (\mathbb{N}, \mathbb{Z})$ with $|n| \leq m$, and $\epsilon_n = 1$ if $n = 0$, $\epsilon_n = 2$ if $|n| > 0$.

[1] `http://faust.grame.fr`

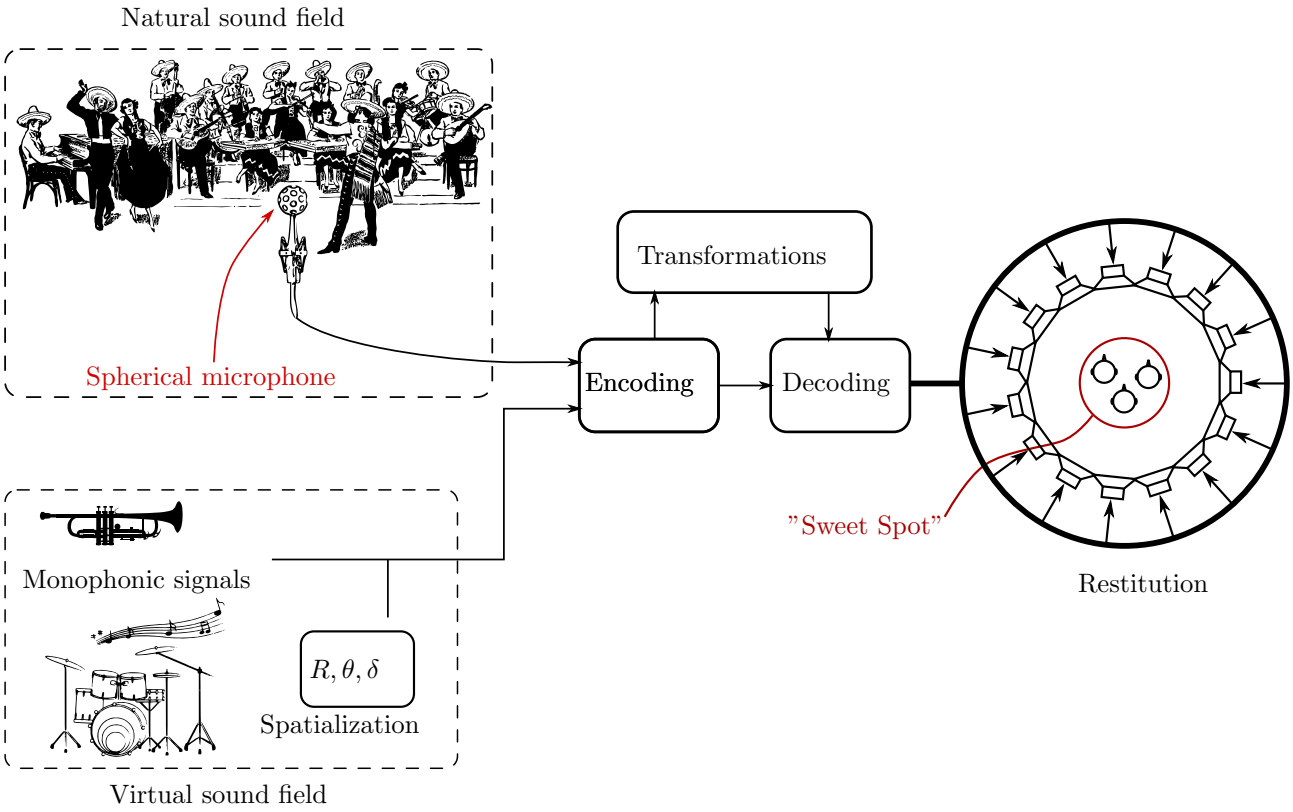


Figure 1: Ambisonic principles: A sound field can be captured in three dimensions using spherical microphone or it can be synthesized with monophonic signals and explicit positions in space. These two approaches result in an encoded sound field in Ambisonic domain up to order M. In this domain, transformations can be done such as: sound field mirroring, rotation, warping, directional filtering, etc. Finally, the decoding step computes the driving signals of the playback loudspeakers to recreate the encoded sound field in a sweet spot region of space. The decoding can be done over various configurations of speakers and even over headphones!

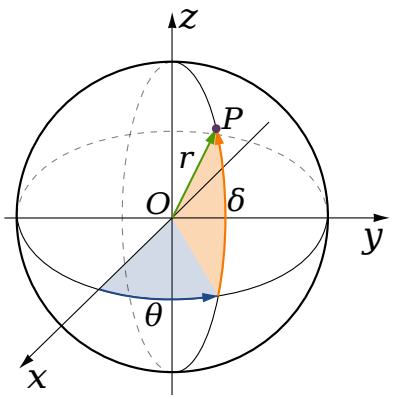


Figure 2: Spherical coordinate system in use. A point $P(x, y, z)$ is described by radius r, azimuth θ and elevation δ.

For each order m, there are $(2m + 1)$ spherical harmonics. Thus a basis truncated at order M contains $(M + 1)^2$ functions. For example, on Fig. 3 are displayed the spherical harmonics up to order $M = 5$ as balloon plots.

Thus, a sound field can be described as a linear combination of spherical harmonics. The weights of this linear combination are called the Ambisonics components or B-Format components. As an example, a sound field described up to order $M = 1$ is described with $(M + 1)^2 = 4$ Ambisonics components. At order $M = 5$ there will be 36 components, etc.

Without giving more details, the concept to retain is that the more components are available to describe the sound field (i.e. the higher the order), the larger the sweet-spot will be on Fig. 1.

At the moment, ambitools offers the possibility of 3D Ambisonics up to order $M = 5$. Thus for all the tools you need to compile, M SHOULD BE CHOSEN SUCH AS $M \leq 5$.

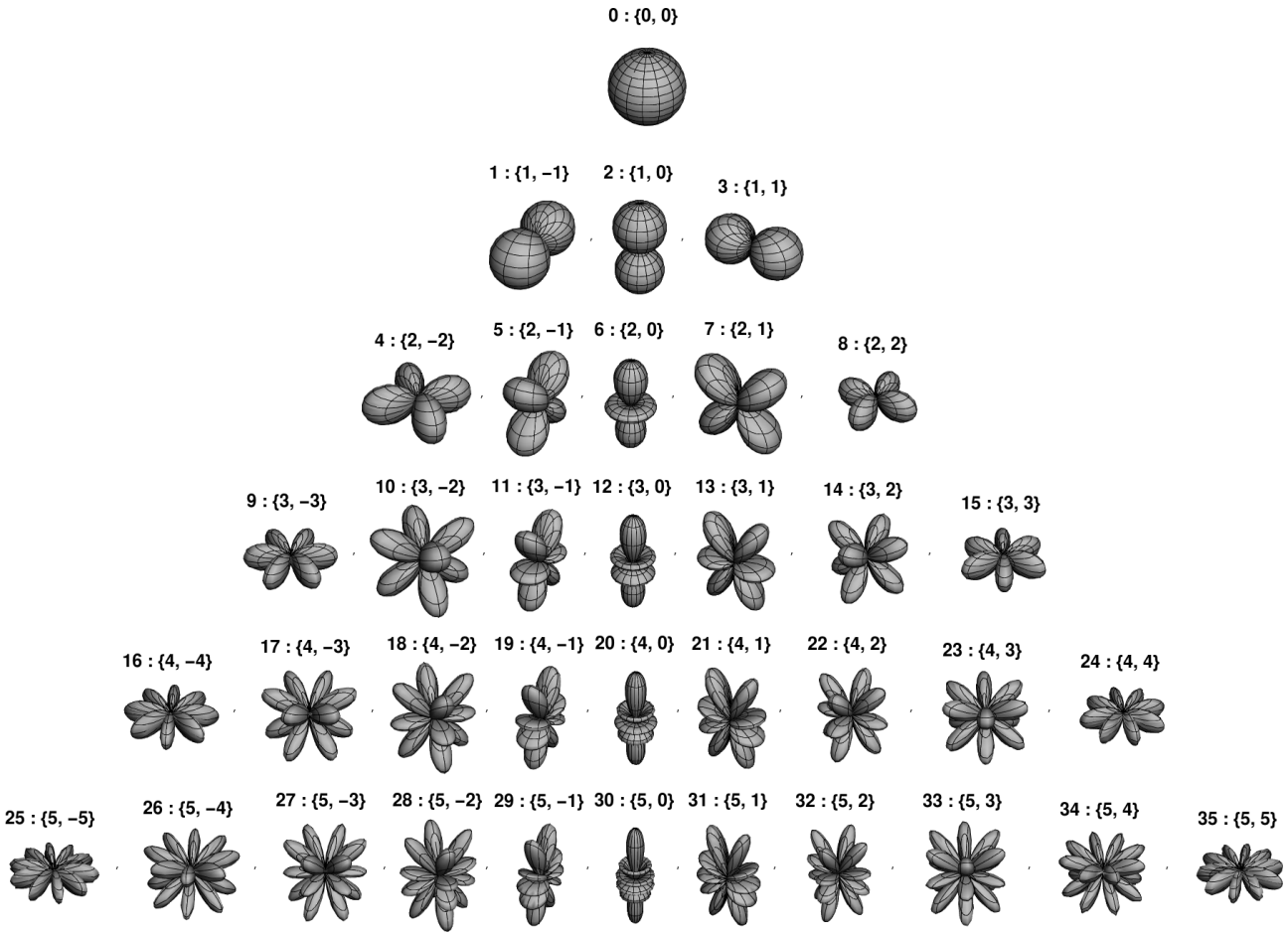


Figure 3: Balloon-plot of the spherical harmonics up to order $M = 5$.

2 Installing ambitools

2.1 Retrieve ambitools repository

To install ambitools, simply go on the github repository[2] and clone it. To do so, open a terminal in the directory you'd like to clone the repository and type the following command:

```
$ git clone https://github.com/sekisushai/ambitools
```

To keep the repository up to date, type the following command at the root of the directory `ambitools/`:

```
$ git pull
```

You can also download a `.zip` file from github[3]
The resulting `ambitools/` folder should have the following structure:

- `Documentation/` Everything concerning the documentation (pdfs, including some scientific papers).
- `Faust/` Everything written in FAUST language (all the ambitools plug-ins + some utilities).
  - `bin/` Compiled plug-ins in various formats.
  - `src/` Source code of the plug-ins.
  - `src/lib/` Shared libraries (spherical harmonics, gui, etc.).
- `FIR/` Finite Impulse Response (FIR) filters banks for binaural rendering and spherical microphone equalization filters, to use with Jconvoler[4], fast convolution software.
  - `spherical_microphones/` Equalization filters for rigid spherical microphone, such as mh acoustics EigenMike®[5].

[2] https://github.com/sekisushai/ambitools
[3] https://github.com/sekisushai/ambitools/archive/master.zip
[4] http://kokkinizita.linuxaudio.org/linuxaudio/
[5] http://www.mhacoustics.com

`hrir/` Head Related Impulses Responses (HRIR) of several people to use with binaural rendering over headphones.

- `Processing/` Everything written in PROCESSING language, namely the spherical VU-Meter ((see Sec. 3.2.1).

  `bin/` Compiled spherical VU-Meter in Java for various architectures.

  `src/` PROCESSING source code.

- `PureData/` Everything written in Pure Data (a few sounds generator patches, head-tracking patch and PlayStation-like remote patch to drive FAUST plug-ins with Open Sound Control protocol, OSC).

2.2 Dependencies

Depending on what you're looking for with the code, the following dependencies should be provided:

Dependency	Usage
OSC support	Drive the tools standalone tools with OSC.
JAVA	Run the Spherical VU-Meter (see Sec. 3.2.1).
JCONVOLVER	Real-time convolution for binaural rendering or spherical microphone signals filtering.
PROCESSING	Edit and compile the Processing code.
FAUST, FAUSTLIVE	Edit and compile the FAUST code.

Table 1: Dependencies of ambitools

2.3 Compile the Faust plug-ins

The FAUST plug-ins source codes are in the sub-folder `Faust/src/`.

2.4 Local Faust installation

If you have FAUST installed on your machine with the required dependencies, you can run the scripts collection `faust2*` to produce the plug-in of your choice in the desired format.

For example, to compile `hoa_encoder_N_sources.dsp` into a standalone Linux jack-qt application with OSC support, type the following command in a terminal in the folder `\Faust/src`

```
$ faust2jaqt hoa_encoder_N_sources.dsp -osc
```

2.5 FaustLive

To compile the plug-ins to your requirements, load the chosen plug-in in FAUSTLIVE[1] and choose `Window/Export As...` (see Fig. 4).

Figure 4: FAUSTLIVE Export Manager

2.6 Compile the Processing VU-Meter

The PROCESSING source code is in the folder `Processing/scr`. You should open the file `Spherical_VU_Meter.pde` in the PROCESSING editor and select "File/Export..." to produce a binary application.

3 The different tools

This section gives a quick presentation of each tool contained in ambitools.

3.1 Faust

The core of the sound processing is written in FAUST language. Note that the majority of the figures presented in the following section will be screen-shots of the tools compiled as standalone JACK applications for Linux, using `faust2jaqt` script. However, thanks to the versatility of FAUST language, note that each of these tools can be compiled for various architecture, using FAUSTLIVE[1] for example.

3.1.1 hoa_N_sources_encoder

- Inputs: N
- Outputs: $(M+1)^2$

This first tool allows to encode N signal inputs to an 3D Ambisonics scene described with Ambisonics components signals up to order M (the B-Format). You can choose N and M at the compilation time in file `hoa_encoder_N_sources.dsp`. The resulting graphical user interface when using `faust2jaqt` script is shown in Fig. 5 for $N = 3, M = 5$. Each input signal can be encoded in B-Format as a plane wave or a spherical wave.

For the plane wave case, the check-box `Spherical Wave` should be unchecked. Consequently, the knob `Radius` and input entry `Speakers Radius` are without effect in this case (a plane wave has no distance information).

For the spherical wave case, the knob `Radius` allows to choose the source radius to origin. Note that to represent a spherical wave in Ambisonic domain, near-field filters are activated [Daniel, 2003, Lecomte and Gauthier, 2015]. Those filters, unstable by nature, are stabilized using the near-field compensation filters from decoding step. Thus in this case, the radius of the spherical loudspeakers grid should be known prior to decoding and given in `Speakers Radius` input entry.

Finally, the outputs of this tool are B-Format signals up to order M. Each order can be muted or each individual Ambisonic components individually.

Figure 5: `hoa_encoder_N_sources` plug-in compiled using `faust2jaqt` script with $N = 2$ and $M = 5$. The `Gain` knob allows to adjust the input level of the input. `Radius` knob allows to choose the source distance to origin in case of spherical wave encoding. `Azimuth` and `Elevation` knobs allow to choose the source direction. The `Spherical Wave` check-box enables the encoding of spherical wave, using near-field filters. `Speakers Radius` sets the radius for the spherical arrays of loudspeakers at decoding stage in case of spherical wave encoding. Finally, the VU-Meters shows the level of 3D B-Format signals in dBFS up to order M. Each meter as a `Mute` check-box and each order as well.

3.1.2 hoa_mic_encoder*

- Inputs: 06, 26, 32, or 50 depending on the configuration.
- Outputs: $(M+1)^2$

It is possible to encode a natural sound field captured by a rigid spherical microphone, according to Fig. 1. To do so, the signals from each capsule on the spherical microphone are mixed to operate the Discrete Spherical Fourier Transform (DSFT) [Lecomte et al., 2015]. Then, the resulting $(M+1)^2$ components are filtered to obtain the Ambisonics components. In ambitools, the DSFT projection is done for several spherical rigid spherical microphone configurations using `hoa_mic_encoder*` tools. The filtering of the resulting components is done using Jconvolver, a real-time fast convolution software. See Sec. 3.3 for further details. Note that the filtering of the outputs signals of `hoa_mic_encoder*` tools is mandatory in order to obtain the Ambisonics components. The different microphone configuration available in ambitools are 06, 26 or 50 capsules spherical microphone arranged on a Lebedev grid [Lecomte et al., 2015] and a 32 capsules spherical microphone arranged on a pentakis-dodecahedron grid, i.e. the mh acoustics Eigenmike® microphone [Elko et al., 2009]. Those configuration can operate the Spherical Fourier Transform up to order $M = 1$, $M = 3$, $M = 5$ and $M = 4$ respectively [Lecomte et al., 2015, Moreau et al., 2006]. The corresponding tools are named `hoa_mic_encoder_lebedev06`, `hoa_mic_encoder_lebedev26`, `hoa_mic_encoder_lebedev50` and `hoa_mic_encoder_eigenmike32` respectively. The graphical user interface when using `faust2jaqt` script is shown in Fig. 6 for the `hoa_mic_encoder_lebedev26` tool compiled with $M = 3$.

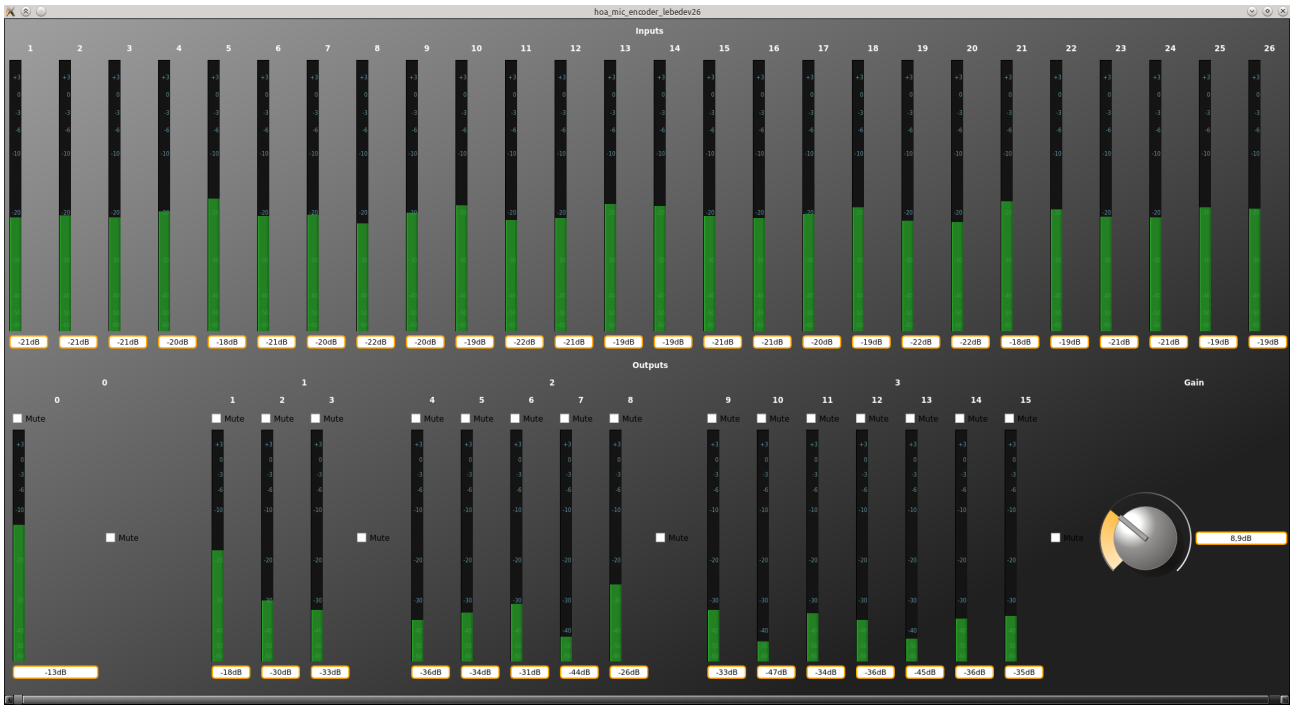


Figure 6: `hoa_mic_encoder_lebedev26` compiled using `faust2jaqt` script with $M = 3$. The VU-Meters at the top show the signal level in dBFS of each capsule on the spherical microphone. The VU-Meters at the bottom show the signal level of each components after the DSFT operation. The check-boxes `Mute` allow to mute some components or all components of an order. The knob `Gain` applies a global gain on the output signals.

Fig. 7 shows the connection in order to obtain the Ambisonics components from the signals of a rigid spherical microphone. The raw signals issued from the spherical microphone are played using a Digital Audio Workstation, Ardour 4 in this case. Those signals feed the tool `hoa_mic_encoder_lebedev50` in the middle of Fig. 7. This tool mix the signals to operate the DSFT. The resulting outputs signals are finally filtered using Jconvolver, on the right of Fig. 7. The outputs of Jconvolver are the Ambisonics components, here up to order $M = 5$.

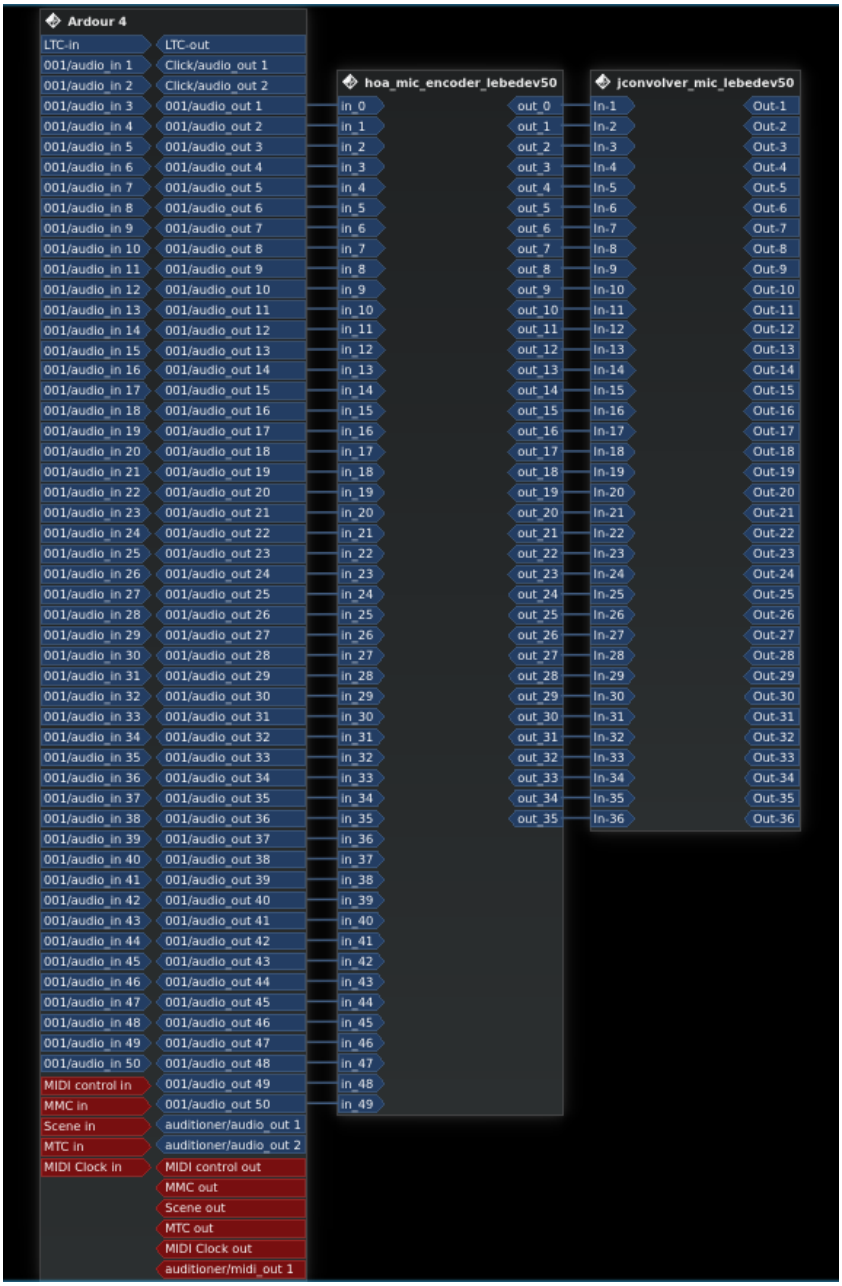


Figure 7: Connections in order to obtain the Ambisonics components from the signals of a rigid spherical microphone. On the left, the 50 outputs of Ardour 4: the 50 signals of the rigid spherical microphone with a 50-node Lebedev configuration. On the middle, the tool `hoa_mic_encoder_lebedev50` which operates the DSFT. On the right, the filtering of the signals with Jconvolver, the outputs of this box are the Ambisonics components up to order $M = 5$. Sceen-shot taken from Claudia under KXStudio Linux distribution.

3.1.3 hoa_decoder_*

- Inputs: $(M+1)^2$
- Outputs: 06, 26 or 50 depending on the decoder.

Three basic decoders by mode-matching [Daniel, 2000, Poletti, 2005] are available at the moment in ambitools: `hoa_decoder_lebedev06`, `hoa_decoder_lebedev26` and `hoa_decoder_lebedev50`. Those decoders allow to decode $(M+1)^2$ Ambisonics signals on Lebedev grids with respectively 6, 26 and 50 loudspeakers [Lebedev, 1975, Lecomte et al., 2015]. Those grids are able to reconstruct the sound field up to order $M = 1$, $M = 3$ and $M = 5$ respectively. If other decoders are required, you should have a look at the ambisonic decoder toolbox from Aaron Heller [Heller et al., 2012], or contact me. The graphical user interface when using `faust2jaqt` script is shown in Fig. 8 for the `hoa_decoder_lebedev50` decoder compiled with $M = 5$. The decoder can be with or without near-field compensation (NFC) filters [Daniel et al., 2003, Lecomte and Gauthier, 2015]. Those filters allow to take into account the finite distance of the loudspeakers: In other terms, if they are disabled, the loudspeakers are modeled as plane-wave generators. In case of spherical wave encoding using the `hoa_N_sources_encoder` plug-in, the `NFC` check-box should be unchecked as the near-field compensation filters are already used at encoding step (see Sec. 3.1.1).

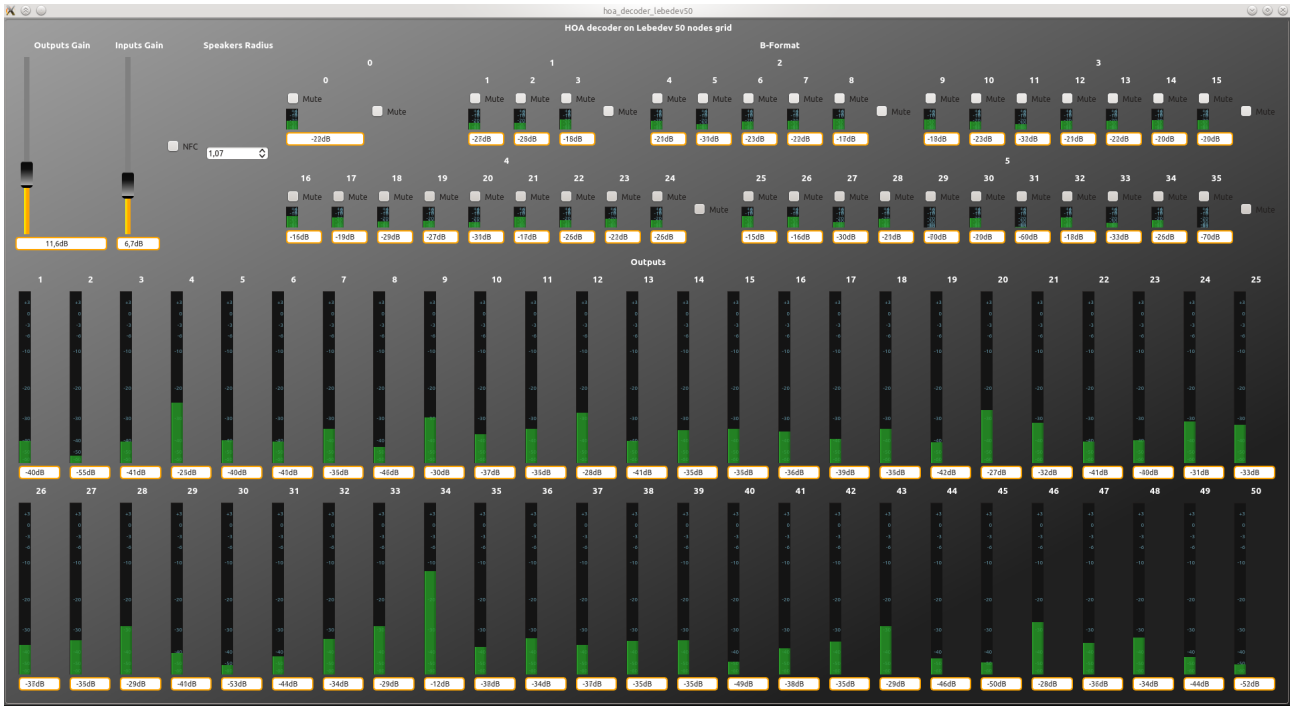


Figure 8: `hoa_decoder_lebedev50` compiled using `faust2jaqt` script with $M = 5$. The slider `Outputs Gain` applies a global gain on all outputs (loudspeaker signals). The slider `Inputs Gain` applies a global gain on all inputs (B-Format signals). The VU-Meters `Inputs` and `Outputs` give the signals level in dBFS. The check-box `NFC` activate or deactivate the near-field compensation filters. The input entry `Speakers Radius` allows to set the spherical grid radius. Finally, the check-boxes `Mute` above all inputs VU-meters allow to mute some specific B-Format signals. Or, all the signal from an order with `Mute` check-boxes on side of a VU-Meter group.

Usage with the Spherical VU-Meter Note that when using these tools to drive the Spherical VU-Meter (Sec. 3.2.1) with OSC, you need to activate OSC transmission mode 2. This is done by launching the tool with the argument `-xmit 2`. For example, with the decoder `hoa_decoder_lebedev50`, the command should be:

```
$ ./hoa_decoder_lebedev50 -xmit 2
```

3.1.4 hoa_decoder_lebedev50_binaural

- Inputs: $(M+1)^2$
- Outputs: 2

`hoa_decoder_lebedev50_binaural` is a decoder which computes directly the binaural signals for left and right hear from the Ambisonics components. It is construct using a basic decoder for 50-node Lebedev grid [Lecomte et al., 2015], and Head Related Impulse Response (HRIR) of a dummy head Neumann KU-100 from [Bernschütz, 2013]. There are $(M+1)^2 \times 2$ Finite Impulse Responses (FIR) filters involved in the process, with 256-taps each. Thus, this tool can be quite CPU consuming as it applies $(M+1)^2 \times 2$ linear convolution operations. The graphical user interface when using `faust2jaqt` script is shown in Fig. 9 for the `hoa_decoder_lebedev50_binaural` binaural compiled with $M = 2$.

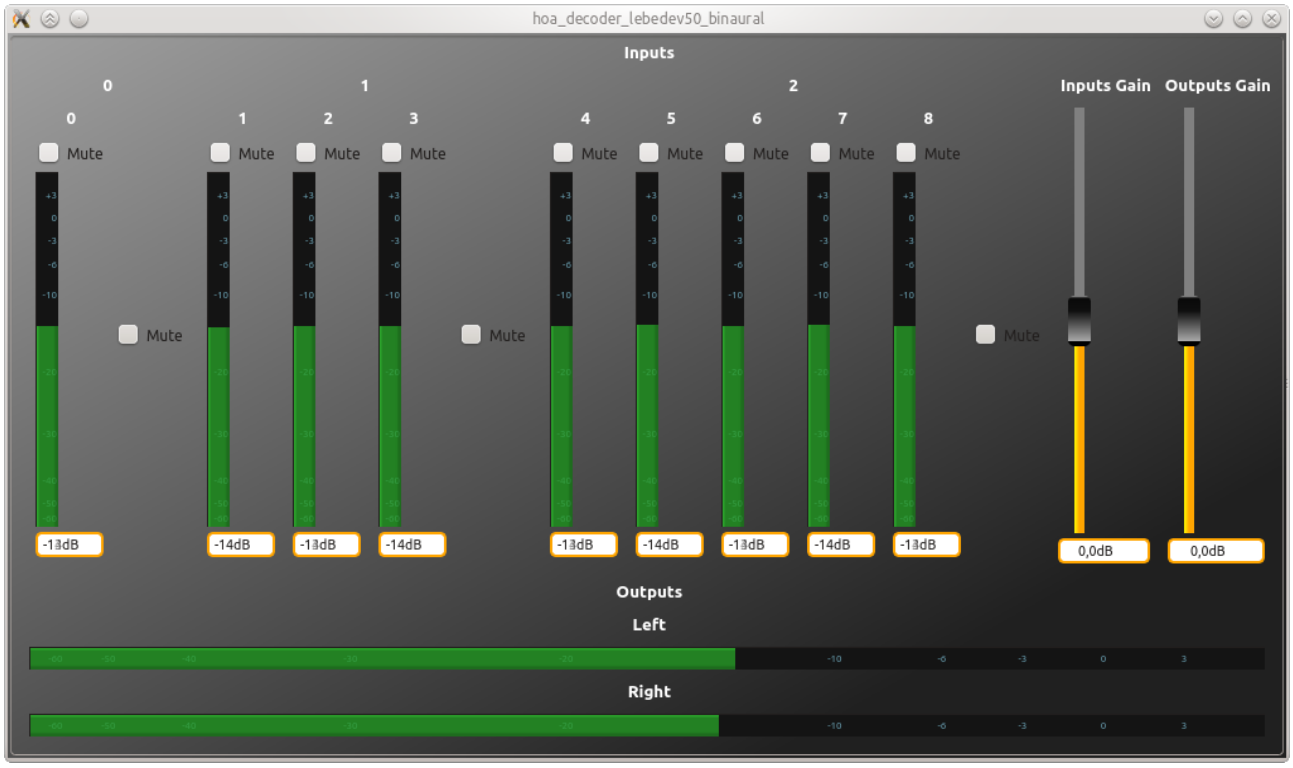


Figure 9: `hoa_decoder_lebedev50_binaural` compiled using `faust2jaqt` script with $M = 2$. The VU-Meters on top show the signal level of inputs Ambisonics components in dBFS. The check-boxes `Mute` allow to mute certain component or all components of an order. The slider `Inputs Gain` applies a global gain on all inputs. The slider `Outputs Gain` applies a global gain on all outputs. Finally, the horizontal VU-Meters at the bottom show the signal level in dBFS of Left and Right channels for headphone signal.

3.1.5 hoa_panning_*

- Inputs: N
- Outputs: 06, 26 or 50 depending on the configuration.

It is possible to compute directly the signals of the loudspeakers without passing by an encoding and decoding process [Lecomte et al., 2015]. This equivalent 3D-panning is implemented in the tools `hoa_panning_N_sources_lebedev06`, `hoa_panning_N_sources_lebedev26` and `hoa_panning_N_sources_lebedev50` for Lebedev grids with 6, 26 and 50 loudspeakers respectively [Lebedev, 1975, Lecomte et al., 2015]. Those grids allow to control the sound field up to order $M = 1$, $M = 3$ and $M = 5$ respectively [Lecomte et al., 2015]. The graphical user interface when using `faust2jaqt` script is shown in Fig. 10 for the `hoa_panning_N_sources_lebedev50` panning tool compiled with $M = 5$ and $N = 2$.

Figure 10: `hoa_panning_N_sources_lebedev50` compiled using `faust2jaqt` script with $M = 5$ and $N = 2$. The slider `Outputs Gain` applies a global gain on all outputs (loudspeakers signals). The check-boxes `Mute Order` allow to mute some Ambisonic order in the computation of the driving signals. The sliders `Gain Radius Azimuth Elevation` control the position and gain of the sources. The check-box `Spherical Wave` allows to switch between the synthesis of a spherical wave or a plane wave. In the case of spherical wave synthesis, the input entry `Speakers Radius` sets the spherical loudspeakers system radius.

Usage with the Spherical VU-Meter When using these tools to drive the Spherical VU-Meter (Sec. 3.2.1) with OSC, you need to activate OSC transmission mode 2. This is done by launching the tool with the argument `-xmit 2`. For example, with the panning tool `hoa_panning_N_sources_lebedev50`, the command should be:

```
$ ./hoa_panning_N_sources_lebedev50 -xmit 2
```

3.1.6 Note about sound field transformation tools in Ambisonic domain

The tools presented in the next sections perform sound field transformations in Ambisonic domain. Thus, they should be used on Ambisonic signals and inserted after encoding step and before decoding step. For instance, on Fig. 11 is shown the connections for the tool `hoa_mirroring`: It is inserted after `hoa_encoder_N_sources` and before `hoa_decoder_lebedev50`, acting on Ambisonic signals.

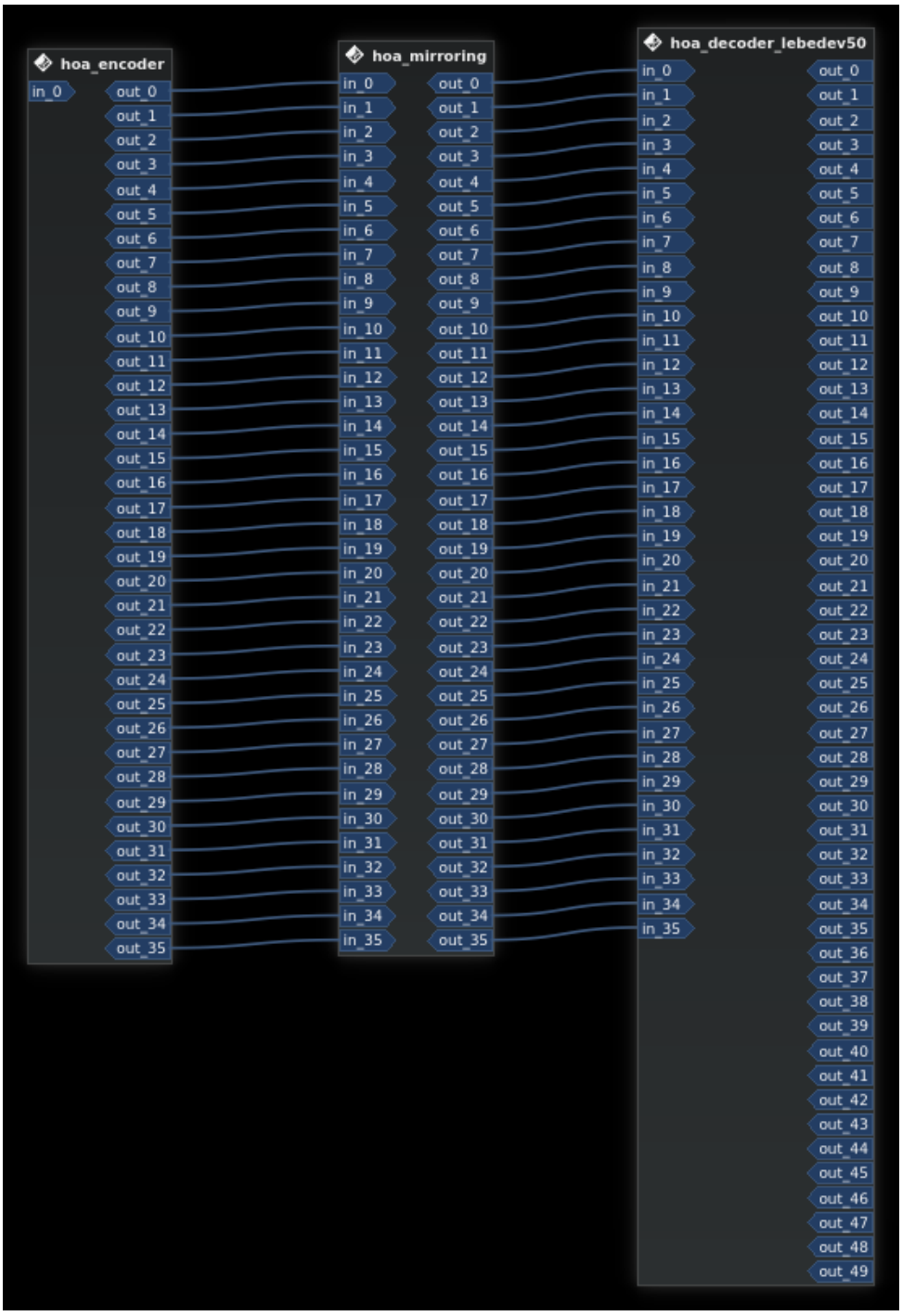


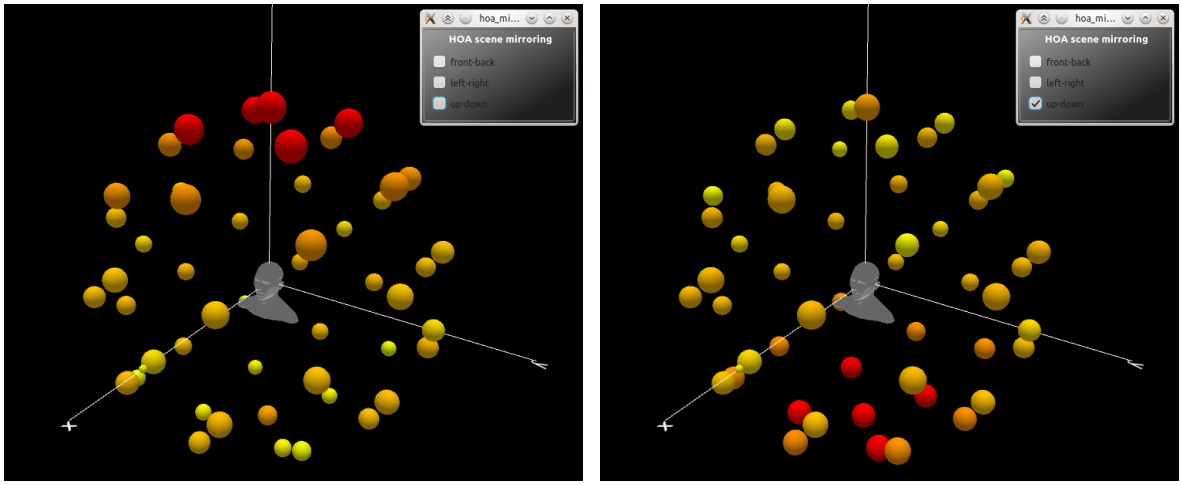
Figure 11: Connections between plug-ins `hoa_encoder_N_sources`, `hoa_mirroring` and `hoa_decoder_lebedev50` compiled with `faust2jaqt` script with $M = 5$. `hoa_mirroring` acts on Ambisonics signals. Screen-shot taken from Claudia under KXStudio Linux distribution.

3.1.7 hoa_mirroring

- Inputs: $(M+1)^2$
- Outputs: $(M+1)^2$

hoa_mirroring performs mirroring operations on the sound field. The directions front and back can be inverted, as well as left and right, up and down, or any combination of the above. In fact, the transformation changes the sign of particular Ambisonic components [Kronlachner and Zotter, 2014]. The graphical user interface when using faust2jaqt script is shown in Fig. 12. To "visualize" the transformation, Fig. 13 shows the Spherical VU-Meter (see Sec. 3.2.1) for a 50-node Lebedev grid on which a sound field is decoded. The sound field is composed of one source positioned at $(1.07\ \text{m}, 0°, 90°)$, i.e, above the head of the listener. One can observe the signal level of each loudspeaker after decoding step. Thus, on Fig. 13(a), when the hoa_mirroring tool is not active, the maximum sound energy is above. When the check-box up-down is checked, on Fig. 13(b) one observes that the sound field has been mirrored according to the up-down inversion. Thus, the sound energy is coming from under the listener at present.

Figure 12: hoa_mirroring compiled using faust2jaqt script. The check-boxes allows to select the mirror transformations among: left-right, front-back or up-down.



(a) (b)

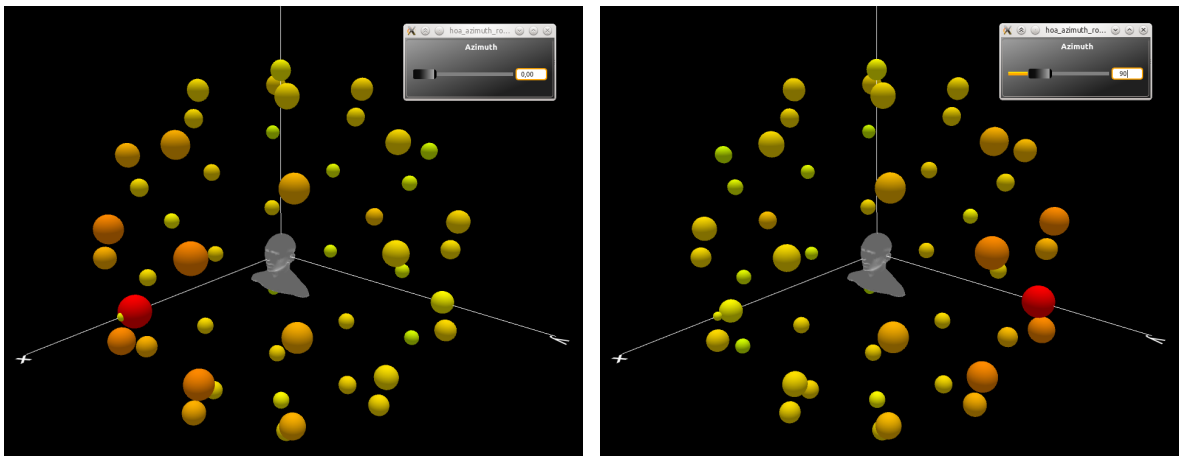
Figure 13: hoa_mirroring effects on a sound field. (a): original sound field decoded on a 50 loudspeaker Lebedev grid. (b) the sound field has been transformed according to a mirroring up-down with hoa_mirroring tool.

3.1.8 hoa_azimuth_rotator

- Inputs: $(M+1)^2$
- Outputs: $(M+1)^2$

hoa_azimuth_rotator performs a rotation of the sound field around $z-$axis (yaw angle). One recalls that the trigonometric sens is chosen for rotation convention (i.e. anti-clockwise, see Sec. 1.2.2). The transformation is done by a rotation matrix operation on the Ambisonic components [Daniel, 2000, Moreau, 2006, Kronlachner and Zotter, 2014]. The graphical user interface when using faust2jaqt script is shown in Fig. 14. It's basically just a slider to choose the azimuth angle of rotation. To "visualize" the transformation, Fig. 15 shows the Spherical VU-Meter (see Sec. 3.2.1) for a 50-node Lebedev grid on which a sound field is decoded. The sound field is composed of one source positioned at $(1.07\text{ m}, 0°, 0°)$, i.e, in front of the listener. One can observe the signal level of each loudspeaker after decoding step. Thus, on Fig. 13(a), when the hoa_azimuth_rotator slider Azimuth is set to 0°, i.e. no rotation, the maximum sound energy is in front of the listener. When the slider Azimuth is set to 90°, on Fig. 13(b) one observes that the sound field has been rotated 90° counter-clockwise around the $z-$axis. Thus, the sound energy is coming from the left of the listener at present.

Figure 14: hoa_azimuth_rotator compiled using faust2jaqt script. The slider Azimuth allows to choose the azimuth angle of rotation, around the $z-$axis.



(a) (b)

Figure 15: hoa_azimuth_rotator effects on a sound field. (a): original sound field decoded on a 50 loudspeaker Lebedev grid. (b) the sound field has been rotated according to a rotation around $z-$axis of $+90°$ using hoa_azimuth_rotator tool.

Head Tracking One possible use of the tool hoa_azimuth_rotator is to compensate the rotation of the head of the listener around $z-$axis when driven by a head-tracking system [Noisternig et al., 2003]. Thus, when using binaural rendering over headphone, the sound field is rotated of minus the angle of rotation of the head, to compensate the head rotation and leave the source at the same position is space for the listener. ambitools provides a Pure Data patch to drive hoa_azimuth_rotator with OSC and perform a head-tracking system with a smart-phone. See Sec. 3.4.1 for further informations.

3.1.9 hoa_beamforming_hypercardioid_to_mono

- Inputs: $(M+1)^2$
- Output: 1

hoa_beamforming_hypercardioid_to_mono is a tool to perform modal beamforming and extract a monophonic signal as if it was recorded by a directional microphone in the sound field. In fact, the Ambisonic signals are weighted and summed to give the monophonic output signal. The weights of the combination traduce the beampattern of the directional microphone which is used [Meyer and Elko, 2002]. Thus, this tool allows to listen to a chosen direction in the 3D sound field as if you were recording in the middle of the sound field with a directional microphone with a specific beampattern. hoa_beamforming_hypercardioid_to_mono provides the weights corresponding to regular hyper-cardioid microphone up to order 3 [Meyer and Elko, 2002, Lecomte et al., 2016]. On Fig. 17, the beampatterns available are shown: the higher the order of the hyper-cardioid is, the more directional the corresponding virtual microphone becomes. The graphical user interface when using faust2jaqt script is shown in Fig. 16 for $M = 3$.

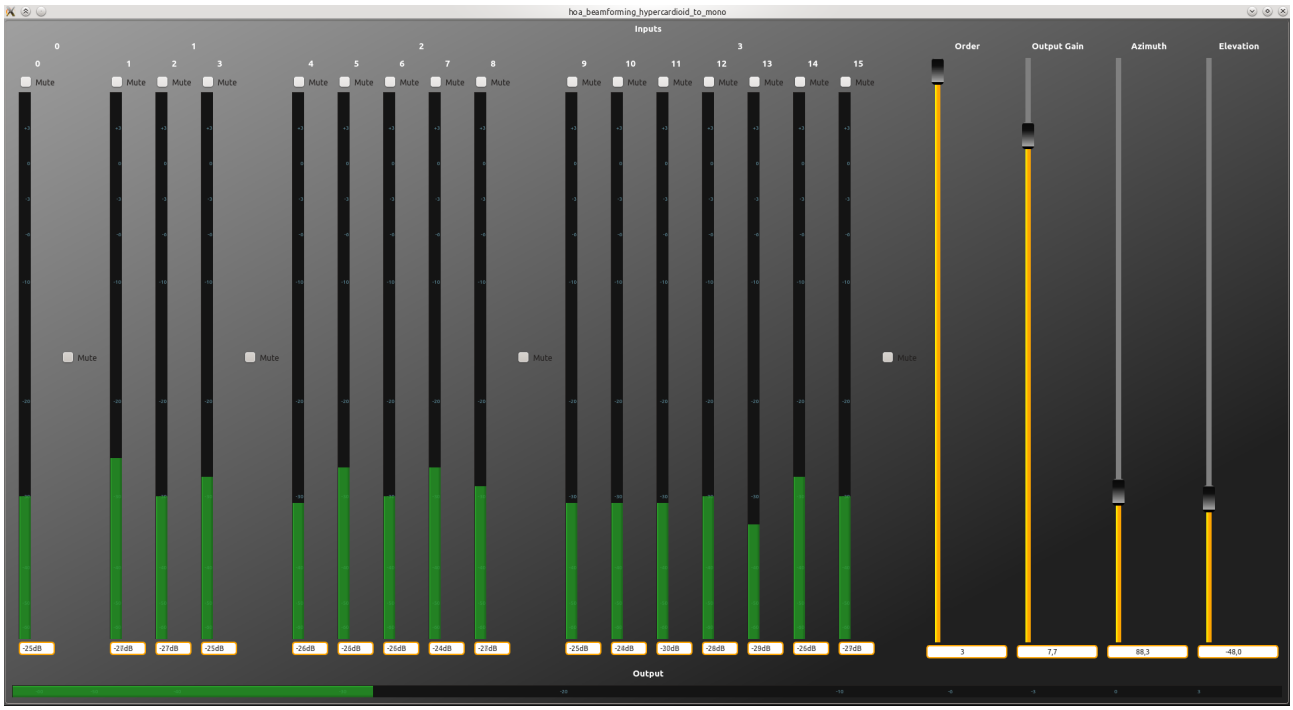
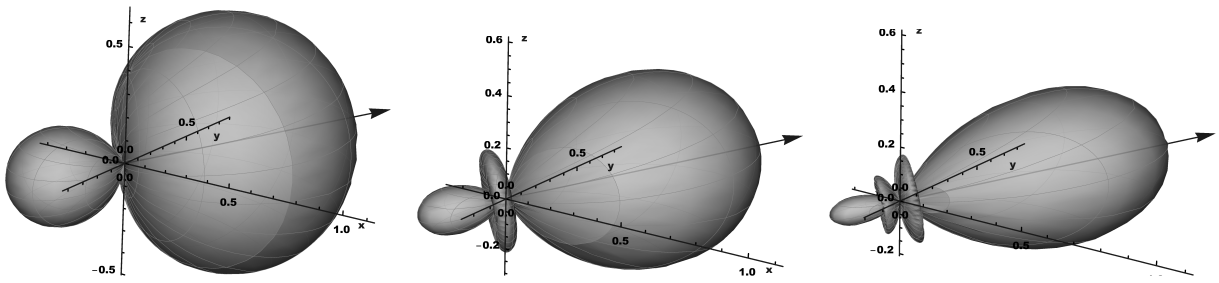


Figure 16: hoa_beamforming_hypercardioid_to_mono compiled using faust2jaqt script. The VU-Meters on the left give the signal level of each Ambisonic component. The check-boxes Mute allow to mute some components of all the components of an order. On the right, the slider Order allows to select the regular hyper-cardioid order. The slider Output Gain applies a gain on the output signal. The sliders Azimuth and Elevation allow to set the steering angles (θ_0, δ_0) of the hyper-cardioid beampattern. Finally, the horizontal VU-Meter on bottom shows the output signal level.



(a) Order 1 (b) Order 2 (c) Order 3

Figure 17: Regular hyper-cardioid beampattern up to order 3. The steering angles are $(\theta_0 = 45°, \delta_0 = 10°)$.

3.1.10 hoa_beamforming_hypercardioid_to_hoa

- Inputs: $(M+1)^2$
- Outputs: $(M_1+M+1)^2$

hoa_beamforming_hypercardioid_to_hoa performs a directional filtering with directional pattern described by regular hyper-cardioid beampatterns of order M_1, as shown on Fig. 17. Thus, unlike the tool hoa_beamforming_hypercardiod_to_mono where the output is a monophonic signal, the outputs here are Ambisonic signals [Lecomte et al., 2016]. The transformation applies a beampattern on the sound field to enhance some directions and reject other. The output HOA scene should be of higher order than the input HOA scene to avoid losing spatial information after the filtering [Lecomte et al., 2016]. In fact, the input scene has $(M+1)^2$ components, and the output scene should have $(M+M_1+1)^2$ components with a beampattern of order M_1. The graphical user interface when using faust2jaqt script is shown in Fig. 18 for $M=3$ and $M_1=2$. In this figure, one observes that the input scene is of order $M=3$ (16 Ambisonic signals), and the output scene is of order $M_1+M=4$ (25 Ambisonic components active). In fact, the slider Order is set to 1, thus $M_1=1$ and $M_1+M=4$. Note that when the slider Order is set to 0, the effect is bypassed as a beampattern of order $M_1=0$ is omnidirectional.

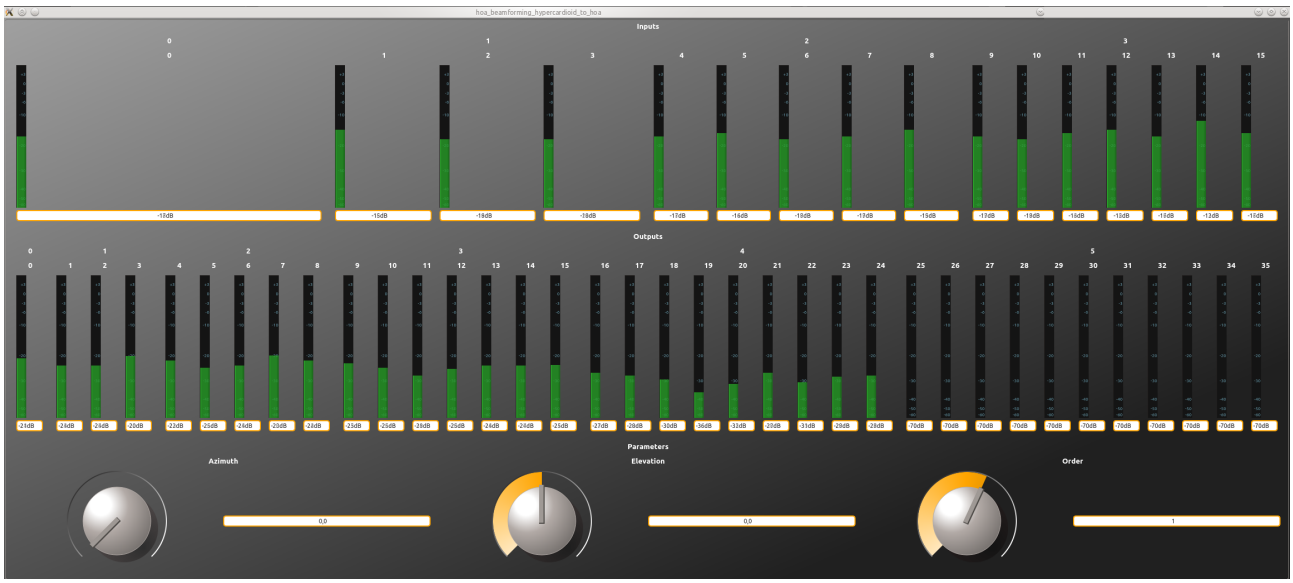
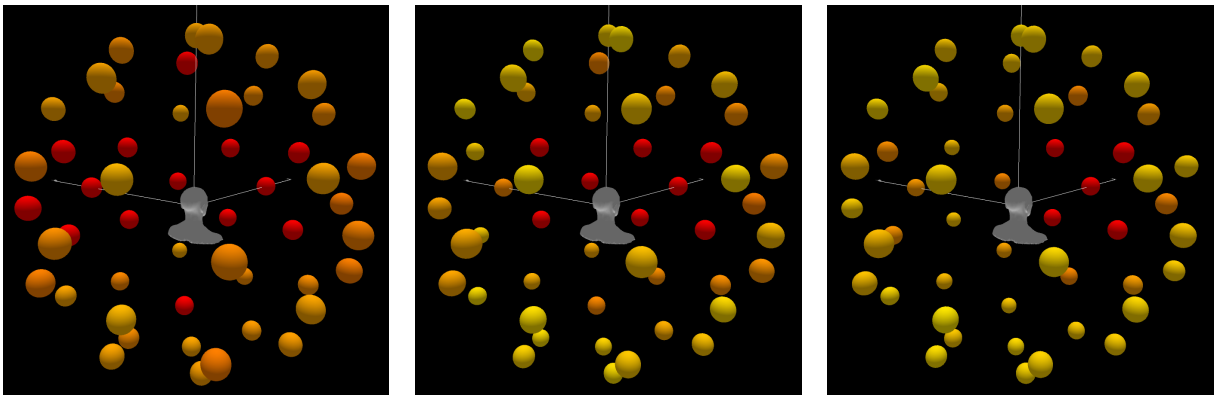


Figure 18: hoa_beamforming_hypercardioid_to_hoa compiled using faust2jaqt script with $M=3$ and $M_1=2$. The VU-Meters at the top show the inputs signal level in dBFS. The VU-Meters at the bottom show the outputs signal level in dBFS. The knob Azimuth and Elevation allow to choose the steering angles of the beampattern. Finally, the slider Order allows to choose the hyper-cardioid order, from $M_1=0$ to $M_1=2$ in this case.

To "visualize" the transformation, Fig. 19 shows the Spherical VU-Meter (see Sec. 3.2.1) for a 50-nodes Lebdev grid on which a sound field is decoded. The sound field is composed of two plane waves coming from direction $(0°, 0°)$ and $(90°, 0°)$ and was produced using tool hoa_encoder_N_sources (see Sec. 3.1.1). The Fig. 19(a) shows the meter when the knob Order in Fig. 18 is set to 0, i.e, the effect is by-passed. Fig. 19(b) shows the same meter when knob Order is set to 1 and knobs Azimuth and Elevation are set to 0° and 0° respectively, i.e. using a hypercardiod of order $M_1=1$ (Fig. 17(a)) with steering angles in front of the listener. One can observes that the sound energy comes now more from the front direction. This is confirmed when the slider Order is set to 2 (with same steering angle) on Fig. 19(c), i.e, with a hypercardioid of order $M_1=2$. In conclusion, t he tool hoa_beamforming_hypercardiod_to_hoa allows to explore the sound scene with more and more directional precision [Lecomte et al., 2016].



(a) `Order` = 0 (b) `Order` = 1 (c) `Order` = 2

Figure 19: `hoa_beamforming_hypercardioid_to_hoa` effects on a sound field. (a) original sound field decoded on a 50-loudspeaker Lebedev grid. (b) the sound field when the knob `Order` is set to 1. (c) the sound field when the knob `Order` is set to 2.

3.1.11 hoa_beamforming_dirac_to_hoa

- Inputs: $(M+1)^2$
- Outputs: $(M+1)^2$

hoa_beamforming_dirac_to_hoa performs a directional filtering on the sound field to listen to only one direction in space. The transformation is like in hoa_beamrforming_hypercardioid_to_hoa but this time, the beampattern is a truncated directional Dirac [Lecomte et al., 2016]. That is to say a function which is null everywhere except in the steering angles direction (θ_0, δ_0). Thus, this tool allows to listen to just one direction in the sound field. The graphical user interface when using faust2jaqt script is shown in Fig. 20 for $M = 3$. Note that this tool provides a crossfade between the situation without filtering and the situation with filtering on. This means that you can smoothly change between a sound field without any directional filtering and a sound field where just one direction is still on.

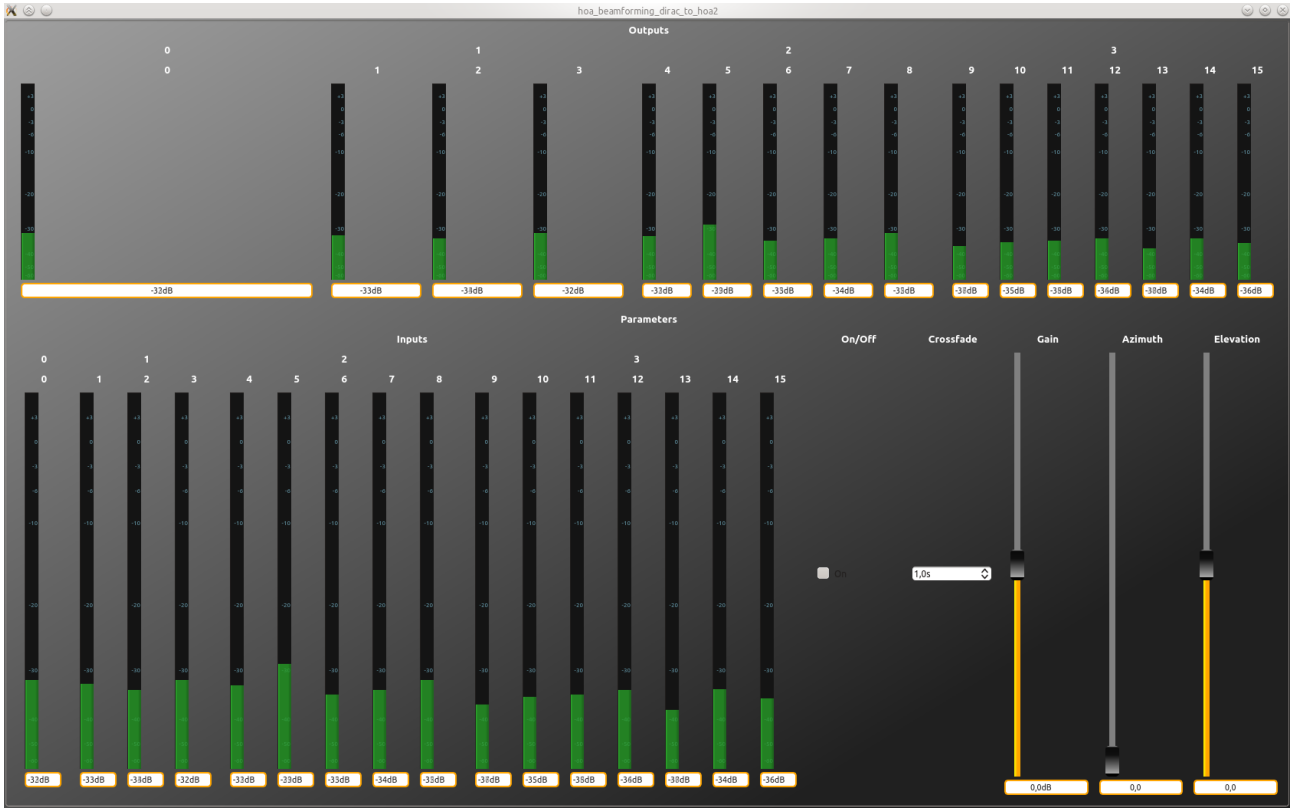
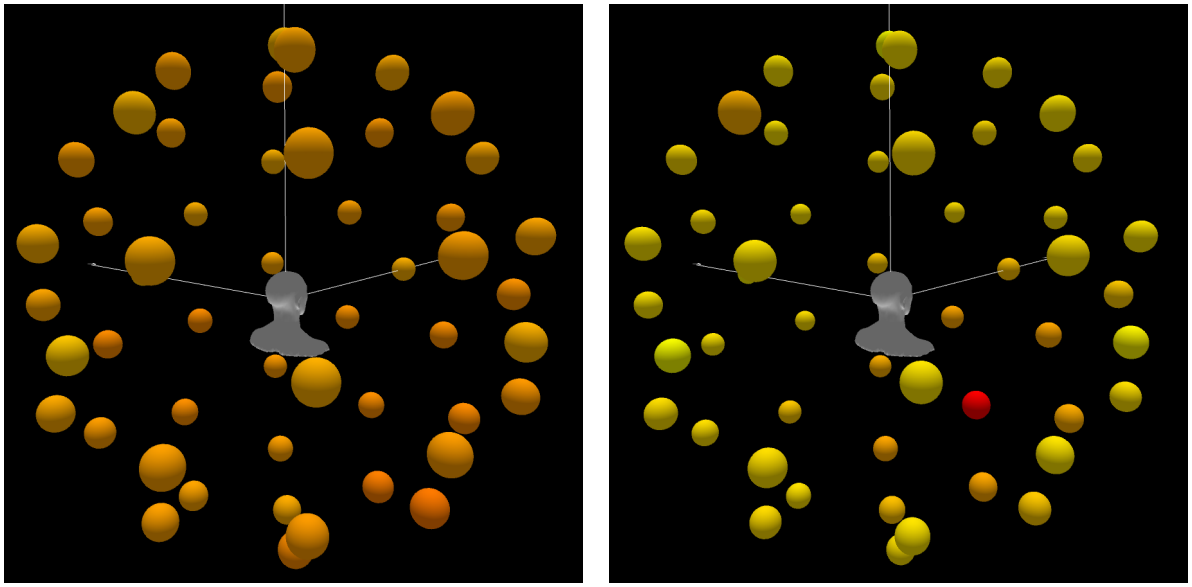


Figure 20: hoa_beamforming_dirac_to_hoa compiled using faust2jaqt script with $M = 3$. The VU-Meters at the top show the inputs signal level in dBFS. The VU-Meters at the bottom show the outputs signal level in dBFS. The check-box On/Off allows to bypass the effect with a crossfade. The input entry Crossfade allows to set the duration of the crossfade. The slider Gain applies a global gain on the output signals. The slider Azimuth and Elevation set the steering angles of the directional Dirac, i.e, the direction to listen to.

To "visualise" the transformation, Fig. 21 shows the Sperical VU-Meter (see Sec. 3.2.1) for a 50-loudspeaker Lebedev grid on which a sound field is decoded. The sound field is issued from a capture by a rigid spherical microphone using 50-nodes Lebedev grid. It is encoded using the tools hoa_mic_encoder_lebedev50 (see Sec.3.1.2) and Jconvolver filters (see Sec. 3.3). Fig. 21(a) shows the original sound field. No particular direction seem to predominant here. However, on Fig. 21(b), the filtering is activated (check-box On/Off checked in Fig. 20). The steering angles are set to $(\theta_0 = 0°, \delta_0 = -45°)$ and one observes that this is the main sound energy direction in the resulting decoded sound field.



(a) Effect Off

(b) Effect On

Figure 21: hoa_beamforming_dirac_to_hoa effects on a sound field. (a) original sound field. (a) the sound field after activation of the directional filtering.

3.2 Processing

3.2.1 Spherical_VU_Meter

ambitools provides a Spherical VU-Meter developped with PROCESSING language. A screen-shot is shown on Fig. 22 for a 50-nodes Lebedev grid [Lecomte et al., 2015]. This tool allows to "see" where the sound energy is in space, instead of using classical in-line VU-Meter. You can zoom, pan and rotate with the mouse the meter while running. The FAUST tools `hoa_panning*` and `hoa_decoder*` emit OSC messages on port UDP 5511. Those messages drive the spherical VU-Meter rendering in real-time: loudspeakers size and color for the meter. On the left of the meter is a color scale in dBFS to have an idea of the value of each loudspeaker signal level. Moreover, this tool allows to see the position of virtual sound sources in space: They are represented as small

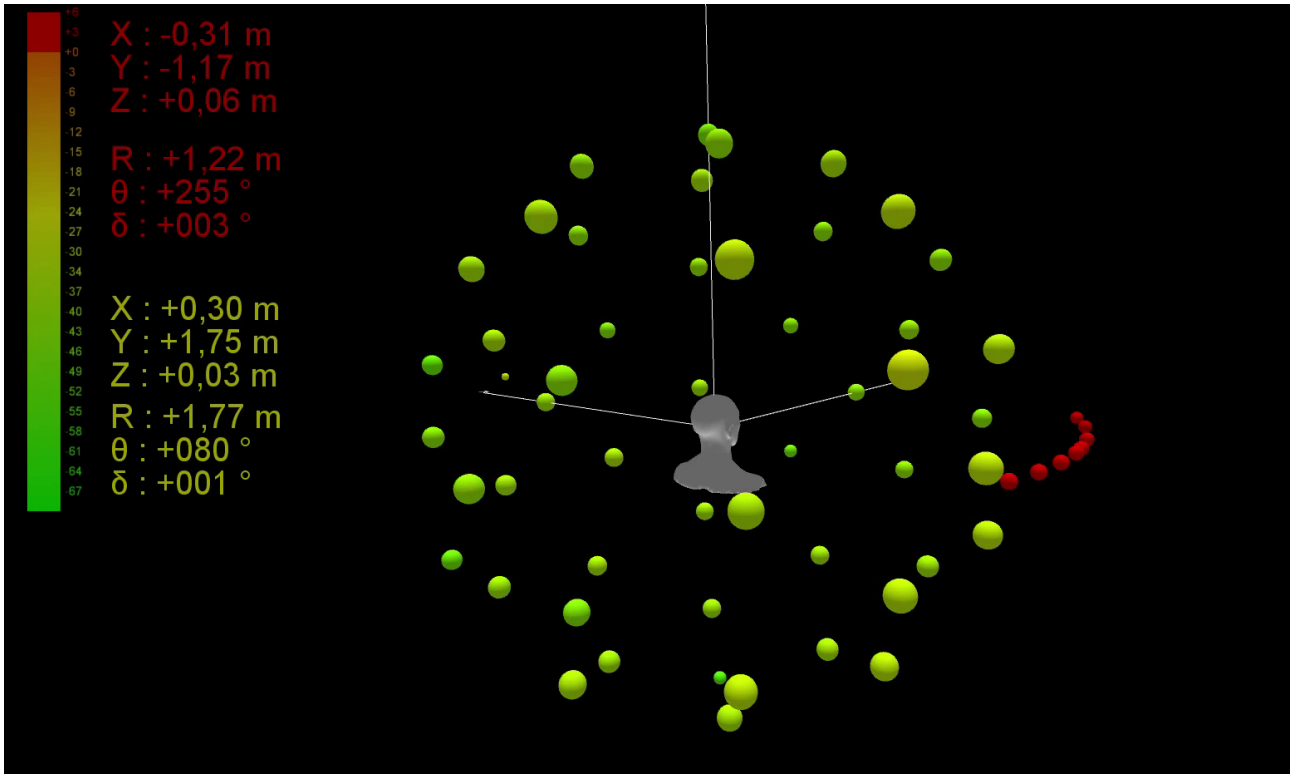


Figure 22: `Spherical_VU_Meter` for a 50 loudspeaker Lebedev array and two virtual sources. Each loudspeaker is represented as a color ball with size and color proportional to RMS Level in dBFS. A scale in dBFS is displayed on the left of the screen. The virtual sources are represented as red an yellow dots with fading color as they are moving. Their coordinates are displayed in Cartesian and spherical coordinates on the left. A grey manikin is standing in the middle of the array to indicates the front direction.

color balls with fading trajectories when moving. The coordinates of the sources are displayed on the left in Cartesian ans spherical. The tools `hoa_encoder*` and `hoa_panning*` emit OSC messages on port UDP 5511 with the values of source position to drive the spherical VU-Meter rendering.

Note that the meter is here presented for a 50-node Lebedev grid and 2 virtual sources. However, the code is flexible and a meter can be compiled for other loudspeaker configurations and number of virtual sources. Indeed, the loudspeaker Cartesian coordinates are provided as a `.cvs` file in folder `/Processing/src/Spherical_VU_Meter/data/`. Thus, to change the meter to an other loudspeaker configuration, just provide the Cartesian coordinates as a `.cvs` file and refer to this file in the code.

3.3 Jconvolver

Jconvolver[4] is a real-time fast convolution software written by Fons Adriaensen. It allows to do convolution operation with FIR filters. ambitools uses FIR filters for two main tasks:

- The binaural rendering with Head Related Impulse Responses (HRIRs).
- The radial equalization filters for rigid spherical microphone.

The filters are provided as `.wav` files containing the impulse responses. They are located in folder `/FIR` as well as the Jconvolver configuration files. Those files are text file containing defining the vector or matrix of filters.

3.3.1 jconvolver_mic*

The folders `FIR/spherical_microphones/jconvolver_mic*/` contain the configuration files and impulse responses of radial equalization filters for several rigid microphone. Those filters are unstable by nature and are stabilized by regularization, as in [Moreau et al., 2006]. They are applied after the Discrete Spherical Fourier Transform performed by `hoa_mic_encoder*` tools (see Sec. 3.1.2) and are mandatory to obtain the Ambisonic components in the context of sound field capture by rigid spherical microphone. Thus, the connections should be as in Fig. 7. Without giving more mathematical details (see [Daniel, 2000, Moreau et al., 2006, Lecomte et al., 2015] for instance), there is one filter by order. The filter at order m is replicated $(2m+1)$ times, for each component at order m issued from the DSFT. For example, as the rigid microphone mh acoustics EigenMike® works up to order $M = 4$ [Elko et al., 2009], there are 4 filters (i.e. 4 `.wav` files) in folder `/FIR/jconvolver_mic_eigenmike32`.

To start the convolution under JACK, the first thing to do is to edit the configuration file, for example `mic_eigenmike32.conf`. Edit and uncomment the line

```
#/cd /put/here/your/local/directory/adress/
```

with the current absolute path address on your local installation. Then, open a terminal in this folder and type the following command:

```
$jconvolver mic_eigenmike.conf
```

You should now have a JACK client named `Jconvolver` with $(M + 1)^2$ inputs and $(M + 1)^2$ outputs, as in Fig. 7.

3.3.2 hrir*

The folder `FIR/hrir/` contains several set of Head Related Impulses Responses (HRIRs) computed on various grids. Those HRIRs allow to do the binaural Ambisonics rendering [Noisternig et al., 2003]. In fact, the signals are decoded on a virtual grid of L loudspeakers, and each virtual loudspeaker signal is convolved with the corresponding HRIR on left and right ear. The summation of all speakers signals on left and right ears gives the binaural signals. Thus, this operation is a matrix of filters with L inputs and 2 outputs. The Jconvolver JACK client should be inserted after the decoder as in Fig. 23 for example. To start the convolution under JACK, the first thing to do is to edit the configuration file, for example `hrir_ku100_lebedev50.conf`. Edit and uncomment the line

```
#/cd /put/here/your/local/directory/adress/
```

with the current absolute path address on your local installation. Then, open a terminal in this folder and type the following command:

```
$jconvolver hrir_lebedev50.conf
```

You should now have a JACK client named `Jconvolver` with L inputs and 2 outputs, as in Fig. 23.

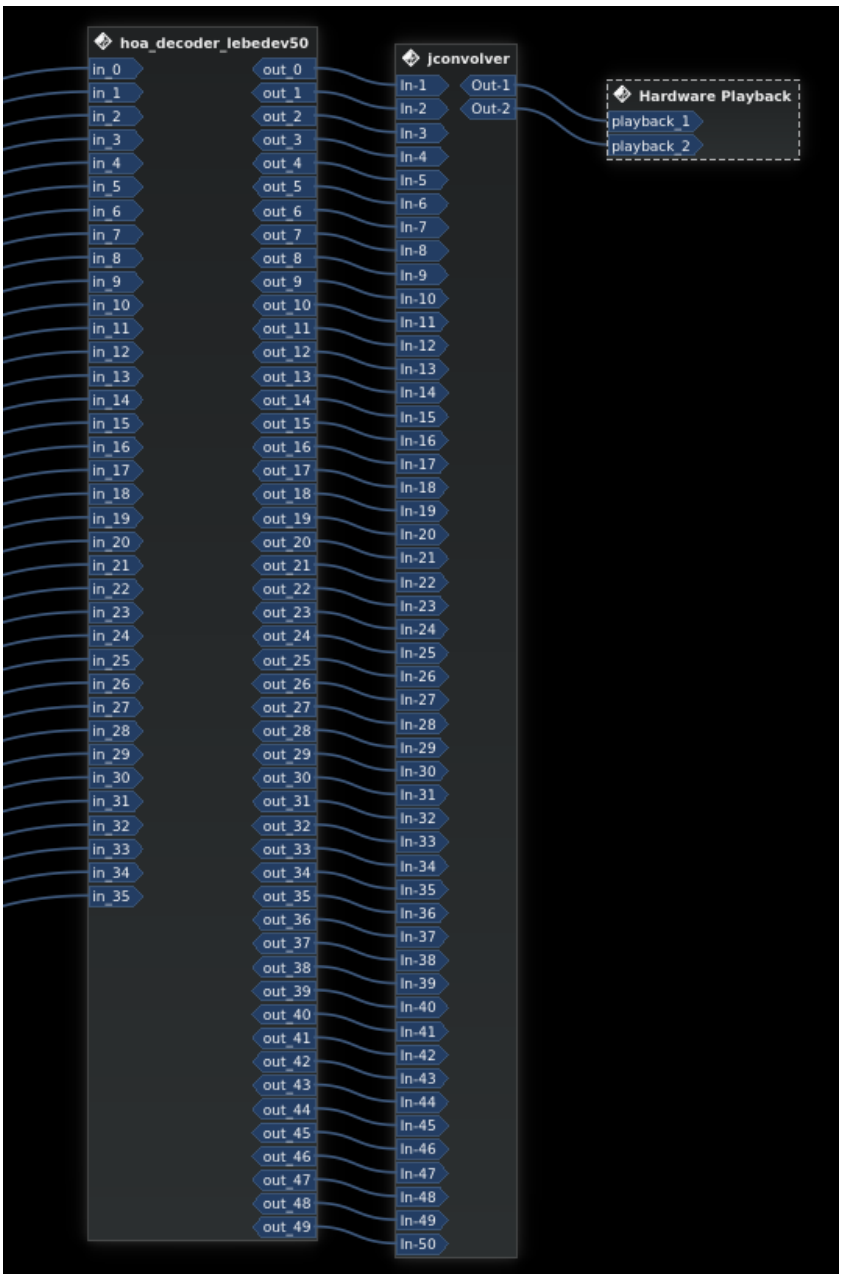


Figure 23: Connection between `hoa_decoder_lebedev50` tool and Jconvolver for binaural rendering. Screenshot taken from Claudia under KXStudio Linux distribution.

3.4 Pure Data

ambitools uses Pure Data to generate some sound but also to emit or convert OSC messages. Particulary the libraries `mrpeach`[6] and `hid`[7] are used to manipulate OSC instructions signals of a USB joystick respectively.

3.4.1 andOSC.pd for Head Tracking

The patch `andOSC.pd` is used to convert the OSC instruction send from the free Android application `andOSC` running on a smartphone. This application transmit on your local network OSC message with the values of accelerometer and gyroscope of your smartphone. On Fig. 24 is shown the graphical user interface of application `andOSC` on an Android smartphone. The value of interest on this figure is the first value of Orientation, which correspond to the yaw angle. Basically, the Pure Data patch `andOSC.pd` listen to all the information sent by application andOSC and extract the yaw angle value to drive the tool `hoa_azimuth_rotator` (see Sec.3.1.8). On Fig. 25 is shown the graphical user interface of the patch `andOSC.pd`. The yaw angle value is extracted and lowpass filtered to avoid brutal change of rotation value for `hoa_azimuth_rotator` tool and limits audible

[6] `https://puredata.info/downloads/mrpeach`
[7] `https://puredata.info/downloads/hid`

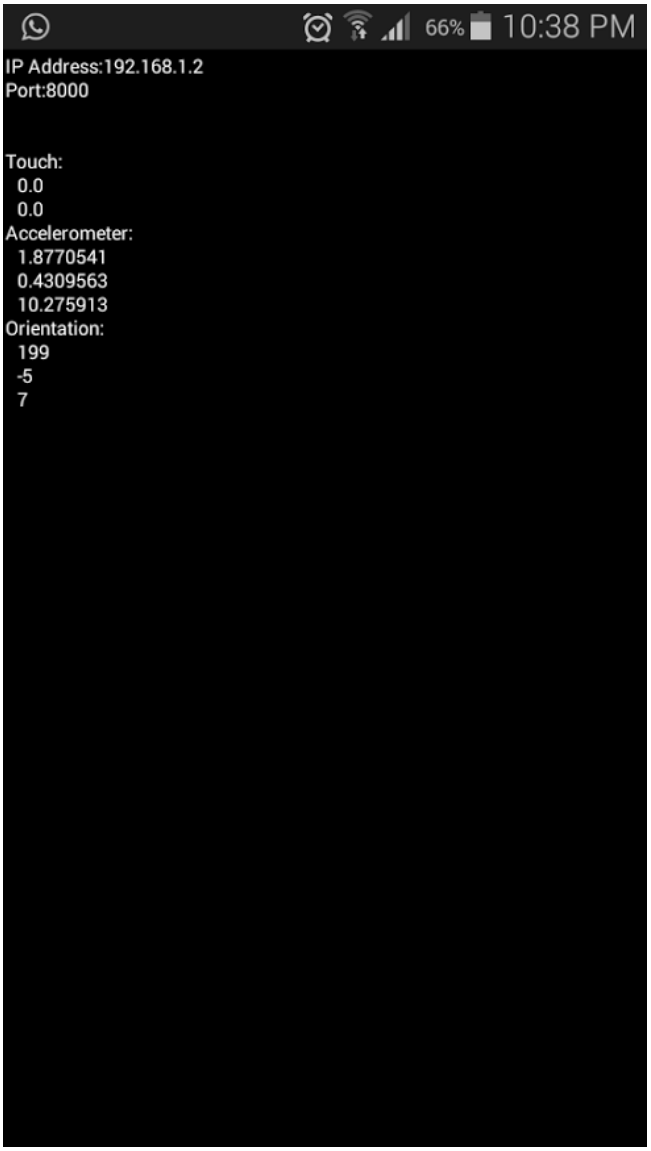


Figure 24: Sceen-shot of application `andOSC` on Android. The value of accelerometer an gyroscope are send by OSC on the local network.

click when brutal rotation of the sound-field is asked. The positive sens of rotation for `andOSC` is clockwise, and it is anti-clockwise for ambitools. Thus, the value of yaw angle transmitted by `andOSC` is directly the head rotation compensation angle needed for head-tracking with `hoa_azimuth_rotator`. To run the transmission of OSC message, DSP button should be on in PureData and tool `hoa_azimuth_rotator`, should listen to OSC message on port 6000. To do so, launch `hoa_azimuth_rotator` with the following command:

```
$./hoa_azimuth_rotator -port 6000
```

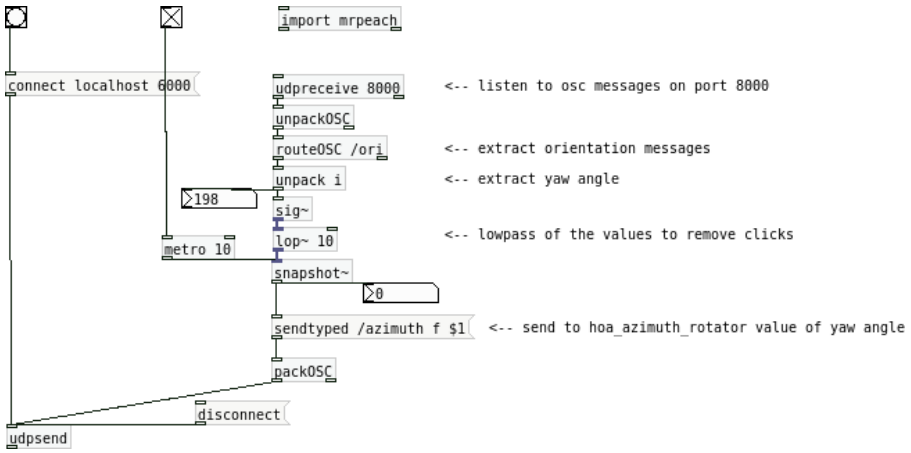


Figure 25: Graphical user interface of patch andOSC.pd. By default, the listening port on is UDP 8000 and transmission port is UDP 6000.

4 Examples of use

4.0.2 Listening to a 3D HOA recording through binaural rendering

Ardour $->$ hoa_mic_encoder_lebedev50 $->$ jconvolver_mic_lebedev50 $->$ hoa_azimuth_rotator $->$ hoa_decoder_lebedev50 $->$ jconvolver_hrir_ku100_lebedev50 $->$ outputs

4.0.3 Spatialize and control the trajectories of flies

hoa_encoder_N_sources $->$ hoa_decoder* $->$ outputs, or
hoa_panning_N_sources_* $->$ outputs

References

[Ahrens, 2012] Ahrens, J. (2012). *Analytic Methods of Sound Field Synthesis*. Springer.

[Bernschütz, 2013] Bernschütz, B. (2013). A spherical far field hrir/hrtf compilation of the neumann ku 100. In *Proceedings of the 40th Italian (AIA) Annual Conference on Acoustics and the 39th German Annual Conference on Acoustics (DAGA) Conference on Acoustics*, page 29.

[Daniel, 2000] Daniel, J. (2000). *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*. PhD thesis, Université Paris 6, Paris.

[Daniel, 2003] Daniel, J. (2003). Spatial sound encoding including near field effect: Introducing distance coding filters and a viable, new ambisonic format. In *Audio Engineering Society Conference: 23rd International Conference: Signal Processing in Audio Recording and Reproduction*, pages 1–15, Helsingør. Audio Engineering Society.

[Daniel et al., 2003] Daniel, J., Moreau, S., and Nicol, R. (2003). Further investigations of high-order ambisonics and wavefield synthesis for holophonic sound imaging. In *Audio Engineering Society Convention 114*, pages 1–18, Amsterdam. AES.

[Elko et al., 2009] Elko, G., Kubli, R. A., and Meyer, J. (2009). Audio system based on at least second-order eigenbeams.

[Heller et al., 2012] Heller, A. J., Benjamin, E. M., and Lee, R. (2012). A toolkit for the design of ambisonic decoders. *Linux Audio Conference*.

[Kronlachner and Zotter, 2014] Kronlachner, M. and Zotter, F. (2014). Spatial transformations for the enhancement of Ambisonic recordings. In *2nd International Conference on Spatial Audio*, Erlangen.

[Lebedev, 1975] Lebedev, V. (1975). Values of the nodes and weights of quadrature formulas of Gauss-Markov type for a sphere from the ninth to seventeenth order of accuracy that are invariant with respect to an octahedron. *USSR Computational Mathematics and Mathematical Physics*, 15(1):44–51.

[Lecomte and Gauthier, 2015] Lecomte, P. and Gauthier, P.-A. (2015). Real-Time 3D Ambisonics using Faust, Processing, Pure Data, And OSC. In *15th International Conference on Digital Audio Effects (DAFx-15)*, Trondheim.

[Lecomte et al., 2016] Lecomte, P., Gauthier, P.-A., Langrenne, C., Berry, A., and Garcia, A. (2016). Filtrage directionnel dans un scène sonore 3D par une utilisation conjointe de Beamforming et d'Ambisonie d'ordre elevés. In *CFA / VISHNO 2016*, pages 169–175, Le Mans.

[Lecomte et al., 2015] Lecomte, P., Gauthier, P.-A., Langrenne, C., Garcia, A., and Berry, A. (2015). On the use of a Lebedev grid for Ambisonics. In *Audio Engineering Society Convention 139*, New York.

[Meyer and Elko, 2002] Meyer, J. and Elko, G. (2002). A highly scalable spherical microphone array based on an orthonormal decomposition of the soundfield. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1781–1784.

[Moreau, 2006] Moreau, S. (2006). *Étude et réalisation d'outils avancés d'encodage spatial pour la technique de spatialisation sonore Higher Order Ambisonics: microphone 3D et contrôle de distance*. PhD thesis, Université du Maine, Le Mans.

[Moreau et al., 2006] Moreau, S., Daniel, J., and Bertet, S. (2006). 3d sound field recording with higher order ambisonics-objective measurements and validation of spherical microphone. In *Audio Engineering Society Convention 120*, pages 1–24, Paris. Audio Engineering Society.

[Noisternig et al., 2003] Noisternig, M., Sontacchi, A., Musil, T., and Holdrich, R. (2003). A 3d ambisonic based binaural sound reproduction system. In *Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*. AES.

[Poletti, 2005] Poletti, M. A. (2005). Three-dimensional surround sound systems based on spherical harmonics. *Journal of the Audio Engineering Society*, 53(11):1004–1025.