ambitools Documentation

Pierre Lecomte
pierre.lecomte@gadz.org

May 29, 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 simplicity.

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. The Faust code is easily editable without beeing an C++ DSP engineer, so if a module should be adapted to a configuration, it'll be very easy to change a few lines in the code and produce quickly the required tool, using for example FaustLive[1].

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

Pierre Lecomte

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

1.2 Ambisonics

This section presents the basis of Ambisonics for 3D sound field synthesis. If you're familiar with Ambisonics you may skip this section.

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.
- `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]
  - `hrir/` Head Related Impulses Responses (HRIR) for various configurations to use with binaural rendering over headphones.
- `Processing/` Everything written in PROCESSING language, namely the spherical VU-Meter.
  - `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, and PlayStation-like remote patch to drive FAUST plug-ins with Open Sound Control protocol, OSC).

2.2 Compile the Faust plug-ins

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

2.3 Local Faust compiler

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.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.dsp -osc
```

2.4 FaustLive

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

[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

Figure 1: FaustLive Export Manager

2.5 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 here will be screenshots of the tools compiled as standalone Jack applications for Linux. However, thanks to the versatility of Faust compiler each of these tools can be compiled for various architecture, using FaustLive[1] for example.

3.1.1 hoa_encoder

This first tool allows to encode a monophonic signal as 3D B-format spatial audio signals. The graphical user interface using Faust Linux jack-qet compiler is shown in Fig. 2. Two types of sources encoding are available : plane waves sources and spherical wave source.

For the plane wave case, the check-box 'Spherical Wave Encoding' should be unselected. Consequently, the sliders 'Source Radius' and input entry 'Speaker Radius' are without effect on B-Format outputs signals.

For the spherical wave case, near-field filters are activated. [Daniel, 2003, Lecomte and Gauthier, 2015]. Those filters use the decoding near-field compensation filters to be stable. Thus in this case, the radius of the spherical loudspeakers layout should be known prior to decoding. Consequently, the slider 'Source Radius' allows to choose the source radius to origin and the 'Speaker Radius' input entry fixes the loudspeakers radius to origin. At the decoding stage, using `hoa_decoder` tools, the near-field compensation filters should be deactivated as they are already used at the encoding (see Sec.3.1.2).

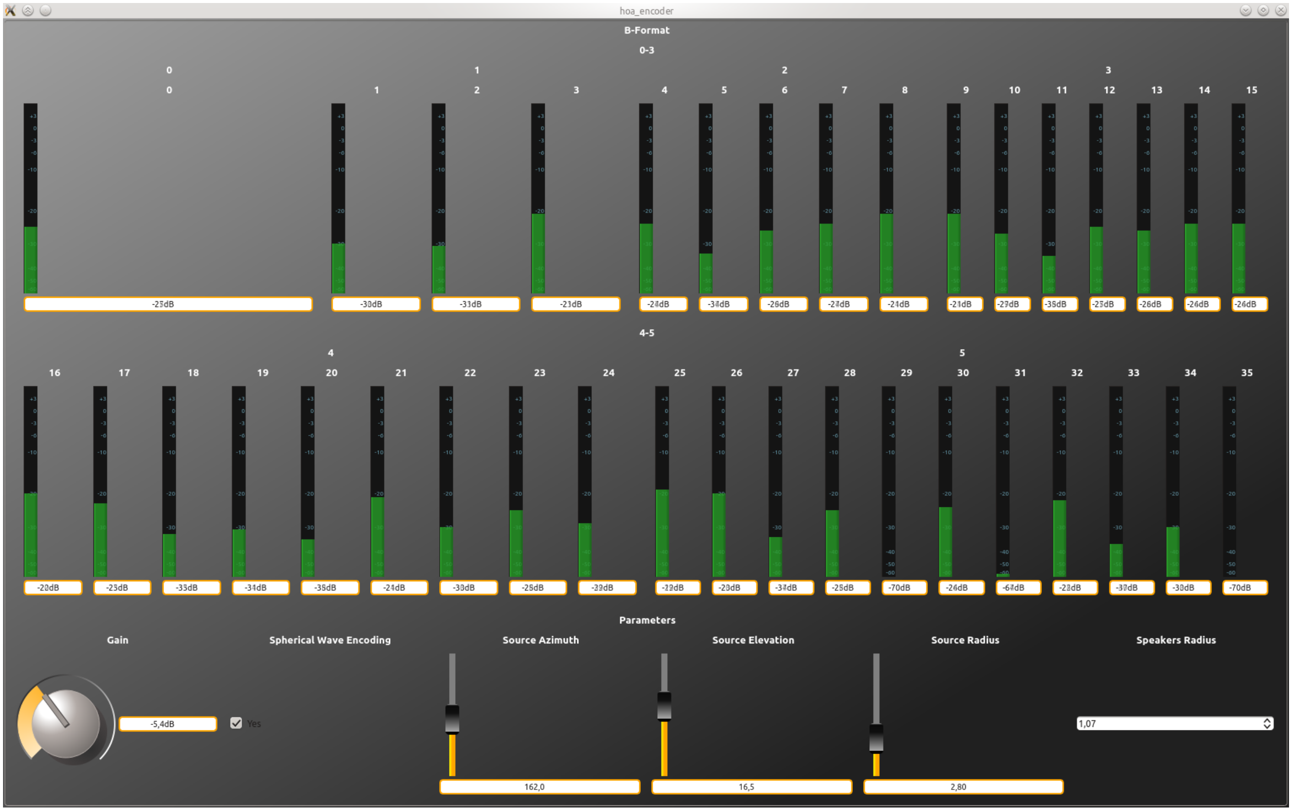

Figure 2: `hoa_encoder` plug-in under linux jack-qt4. The 'Gain' button allows to adjust the input level of monophonic signal. The 'Spherical Wave Encoding' check-box enables the encoding of spherical wave, using near-field filters. 'Source Azimuth' and 'Source Elevation' sliders allows to choose the source direction. 'Source Radius' allows to choose the source distance to origin in case of spherical wave encoding. 'Speakers Radius' sets the radius for the spherical arrays of loudspeakers at decoding stage in case of spherical wave encoding. Finally the B-Format VU-Meters shows the level of 3D B-Format signals in dBFS up to order 5 (36 signals).

3.1.2 hoa_decoder_*

Two basic decoder by mode matching [Daniel, 2000, Poletti, 2005] are available at the moment in ambitools: `hoa_decoder_lebedev26` and `hoa_decoder_lebedev50`. Those two decoder allows to decode a B-Format on Lebedev grids with respectively 26 and 50 loudspeakers [Lebedev, 1975, Lecomte et al., 2015]. Those grids are able to reconstruct the sound field up to order 3 and 5 respectively. If other decoder 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 using Faust Linux jack-qt compiler is shown in Fig. 3 for the `hoa_decoder_lebedev50` decoder. 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 : if they are disabled, the loudspeakers are seen as plane-wave generators. In case of spherical wave encoding using the `hoa_encoder` plug-in, the filters should be disabled as the near-field compensation filters are already used at encoding step (see Sec. 3.1.1).

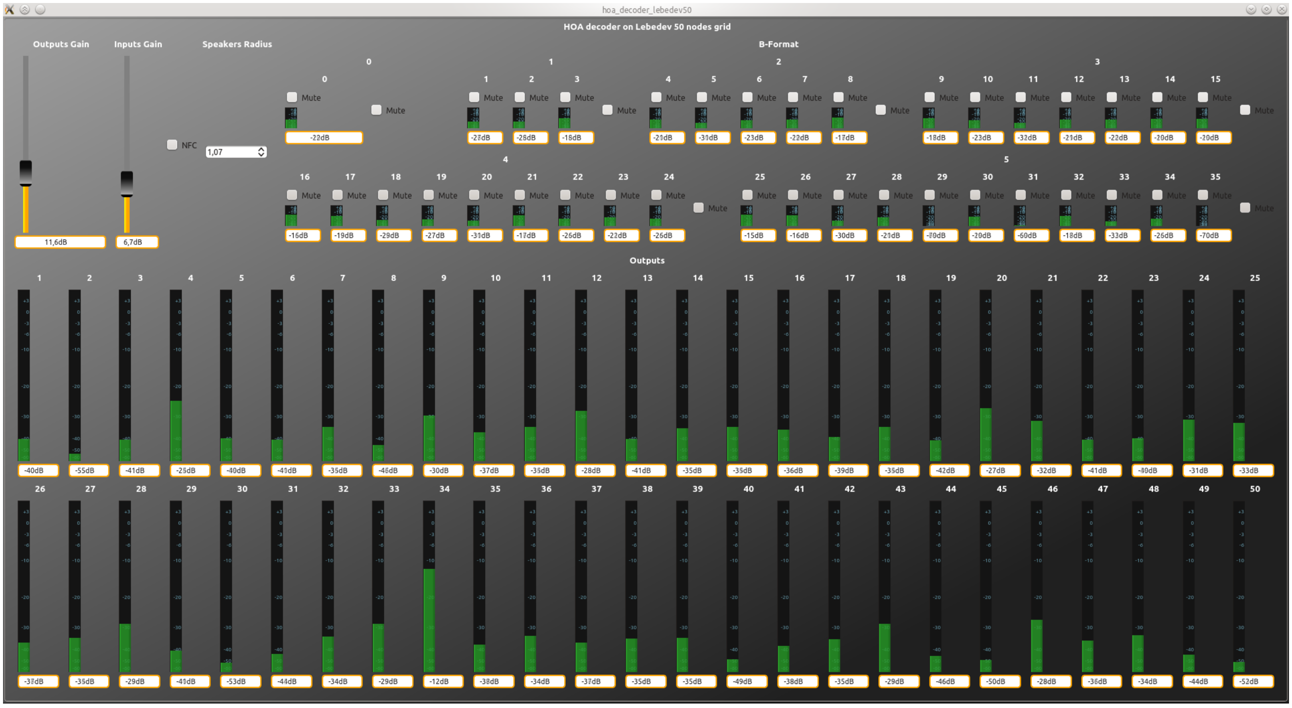

Figure 3: `hoa_decoder_lebedev50` plug-in under linux jack-qt4. The slider 'Outputs Gain' apply a global gain on all outputs (loudspeakers signals). The slider 'Inputs Gain' apply a global gain on all inputs (B-Format signals). The VU-Meters 'Inputs' and 'Outputs' give the signal level in dBFS. The check-box '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 an VU-Meter group.

3.1.3 hoa_panning_*

It is possible to compute directly the driving signals of the loudspeakers without passing by and encoding/decoding process [Lecomte et al., 2015]. This equivalent 3D panning is implemented in the tools `hoa_panning_lebedev26` and `hoa_panning_lebedev50` for 26 and 50 loudspeakers Lebedev grids respectively. The graphical user interface using Faust Linux jack-qt compiler is shown in Fig. 4 for the `hoa_panning_lebedev50` tool. For this version, two spherical waves are synthesized.

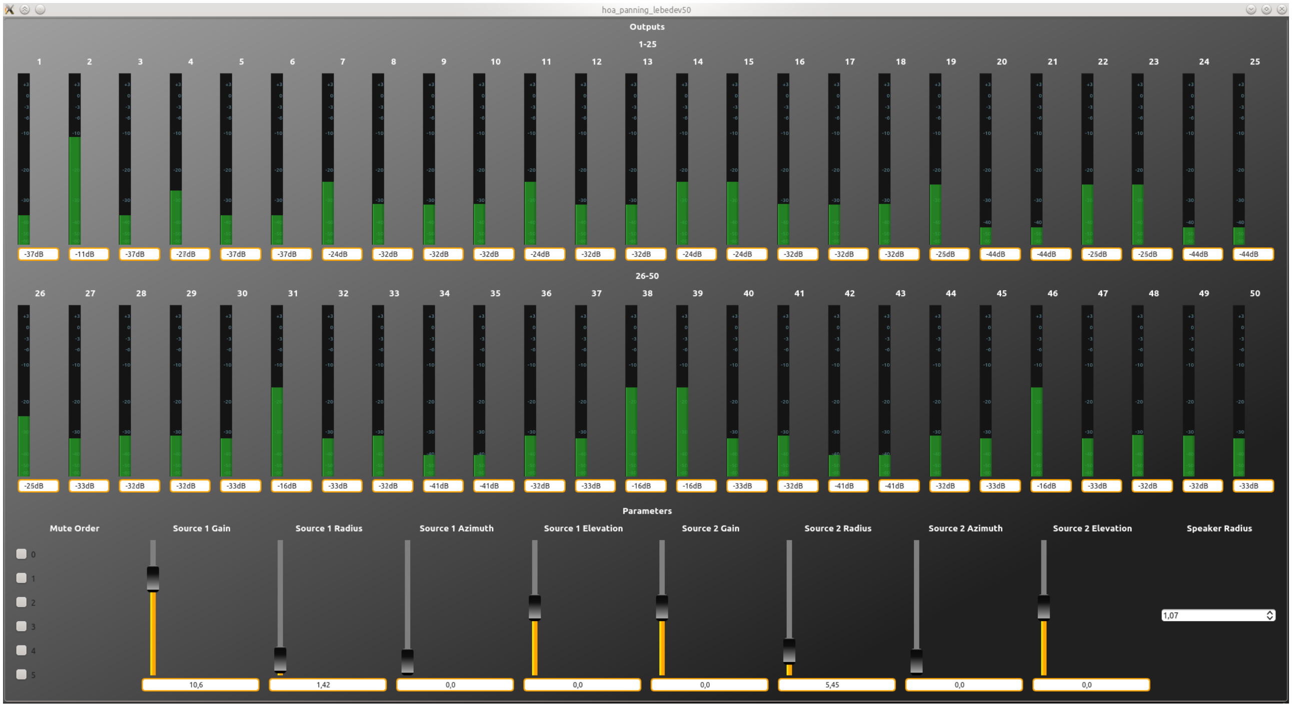

Figure 4: `hoa_panning_lebedev50` plug-in under Linux jack-qt4. The slider 'Outputs Gain' apply a global gain on all outputs (loudspeakers signals). The check-boxes 'Mute Order' allows to mute some Ambisonic orders in the computation of the driving signals. The sliders 'Gain' 'Radius' 'Azimuth' 'Elevation' controls the position and gain of two sources (spherical waves). In order to stabilize the near-field filters for spherical wave synthesis, the input entry 'Speaker Radius' fixes the loudspeaker array radius.

3.1.4 hoa_mirroring

The plug-in `hoa_mirroring` allows a sound field transformation in Ambisonic domain. Thus, it should be inserted somewhere in between encoding and decoding steps. The sound field is here flipped upside-down, left-right or front-back, or any combination of the above. The tool changes the sign of particular Ambisonic components to realize the transformation [Kronlachner and Zotter, 2014]. The graphical user interface using Faust Linux jack-qt compiler is shown in Fig. 5. An example of plug-in insertion under Linux jack is shown in Fig. 6.

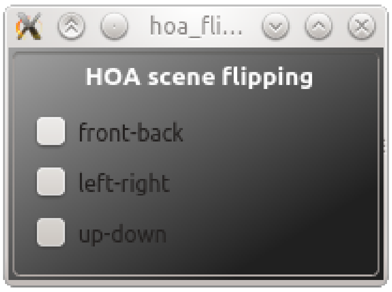

Figure 5: `hoa_mirroring` plug-in under Linux jack-qt4. The check-boxes allows to select the mirror transformation : 'left-right', 'front-back' or 'up-down'.

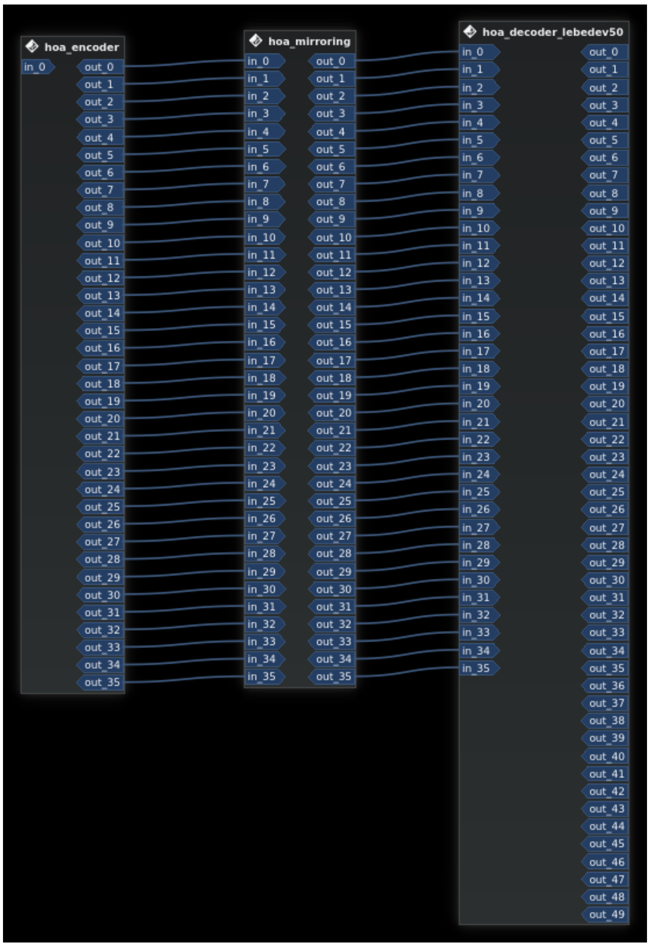

Figure 6: Example of `hoa_mirroring` insertion between encoding and decoding steps, i.e. in Ambisonic domain.

3.1.5 hoa_azimuth_rotator

3.1.6 hoa_beamforming_to_mono

3.1.7 hoa_beamforming_dirac_to_hoa

3.2 Processing

3.2.1 Spherical_VU_Meter

ambitools offers a Spherical VU-Meter developped with Processing language. A screen-shot is shown on Fig. 7. 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 the view of 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 : loudspeakers size and color for the meter, and source position.

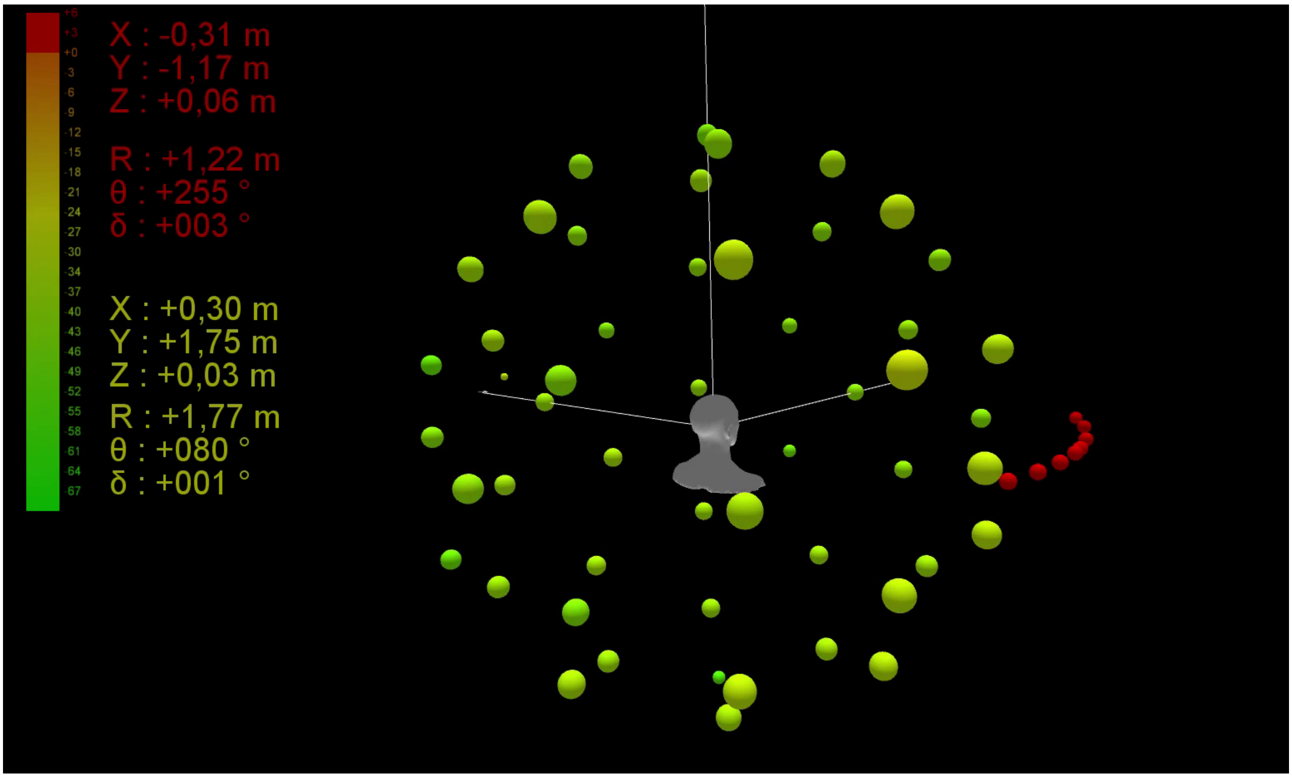

Figure 7: `Spherical_VU_Meter` for a 50 loudspeaker Lebedev array and two virtual sources. Each loudspeaker is represented as a color ball with size and color proportional to RMS Level in dBFS. A scale in dBFS is displayed on the left of the screen. The virtual sources are represented as red an yellow dots. 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.

3.3 Jconvolver

3.3.1 jconvolver_mic*

3.3.2 hrir_lebedev50

References

[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.

[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., 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.

[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.