Installing CUBEmixer

This is a short tutorial how to install the CUBEmixer under your favorite OS. (as a first usage Linux, especially Debian is used)

Overview

About Requirements, Software Prerequisites, Downloads and Versions

CUBEmixer as Framework

The CUBEmixer is programmed in the graphical computer music language Pure Data (PD) and uses additional external libraries from the pure-data project. As a framework it is a collection of objects and functions for PD, separating the graphical User Interface (GUI) and Digital Signal Processing (DSP) Tasks. They can be run parallel on the same or on different computer as own programs and have to be configured for the specific application.

There have been a number of special PD libraries developed for the CUBEmixer, which over time are integrated into the mainstream PD external libraries, so the packaging will change with new versions of the CUBEmixer.

Since the CUBEmixer could also be used as a framework for different projects, there is the possibility to build your application around the CUBEmixer, without change it directly in an separate parallel folder.

Which Version of Pure Data (PD) is needed ?

Since PD is constantly changing with backward compatibility mostly guaranteed for the pd-vanilla version, CUBEmixer uses this version from Miller S. Puckette upstream to the subversion repository.

For this tutorial we use CUBEmixer V3.0 from subversion directory and PD Version 0.42, from the subversion directory installed locally in the CUBEmixer folder, but it should be compatible with 0.41 and future versions on your system, look at:

- http://crca.ucsd.edu/~msp/software.html
- http://puredata.info/downloads
- pd-vanilla from your favorite distribution

Suggestion: Do not use PD-extended or anything like this, since it can have incompatible libraries and patch-collections and also the PATHes to libraries can vary on different systems.

Which libraries ?

Because of the dependencies to various externals-libraries from the pd community project and the additional external libraries only for the CUBEmixer, there was the decision that CUBEmixer Packages includes these. In the subversion-tree they are included in the sources directory as external references. So you do not need to install additional external libraries at all, except the ones special for your project. Be careful which externals you use, because PD is lnown for name-space clashes and the CUBEmixer want to provide a smooth upgrade path.

Which Version of CUBEmixer ?

Because of historical reasons and dependencies on art projects, there are some versions for major steps in the development. The newest Version is mostly available as a package or at least as a Release Candidate and only on major Version architecture will change.

Minor releases are done for new extensions, new plugins which will be upwards compatible and for bug fixes.

If you want to stay on head of development, which I can recommend, use the subversion-trunk. There are some branches for different development experiments, which are mostly connected to art projects and where special additions will be merged into the trunk.

All of this versions can be found in the release listings of the project page:

- http://ambisonics.iem.at/xchange/products/cubemixer/

Downloads

This tutorial is done on and therfore covers the installation of Version 3.0, so we will download this either as Package or from svn.

Package:

- http://ambisonics.iem.at/xchange/products/cubemixer/releases/

Subversion:

Go to your project folder on your account and perform this on the command line or your favorite graphical tool to check out with subversion

```
svn co https://iem.svn.sourceforge.net/svnroot/iem/spatialization/CUBEmixer/trunk CUBEmixer
```

Installing from the source

How to install from the source. If you use a Package, you can skip this step.

We asume here you have already compiled some stuff with linux and know what is a "Makefile" and a compiler - if not please use a premade binary packages , unpack it to your working directory and skip this page, since then all externals and pd should have been compiled for your system.

1. get the source

Download the source distribution at releases in the project page:

- http://ambisonics.iem.at/xchange/products/cubemixer/releases

or check out from subversion repository (recommended):

```
svn co https://iem.svn.sourceforge.net/svnroot/iem/spatialization/CUBEmixer/trunk CUBEmixer
```

Now you should have following directory structure:

```
CUBEmixer /...
          /lib/libs ... here the externals goes in so it is empty now
          /...
          /src ... sources for pd and externals
          /...
```

2. Prerequisites

Prerequisites, besides the C-Compiler and basic C-libraries, and depencies for the external libraries and PD.

There are two options which to use PD, either your PD on the system or use the one in the CUBEmixer itself. For second option goto the src directory and read the README there. (I will not repeat it here since it may change over time.)

For the external libraries please see in their README.txt or INSTALL.txt, if you get a problem during compiling.

3.) compiling and installing

If you want PD inside the CUBEmixer, open a terminal and change into the src/ directory in the CUBEmixer folder:

```
> cd src
> make
```

If not you have can compile and install external locally. Change into the src/libs directory and type:

```
> cd src/libs
> make
please give the path to your PD-Sources with: make PDSOURCE=/path/to/pd/src
make: *** [checkpddir] Fehler 255
```

You are asked for the path to pd/src, so name your path here as example /usr/local/src/pd/src. Note: use an absolute path

```
> make PDSOURCE=/usr/local/src/pd/src/
... compiling ...
cp zexy/*.pd_linux ../../lib/libs/
cp zexy/abs/*.pd ../../lib/libs/
>
```

If there is no error your done, skip to next section.

If there is an error compiling one of the libraries you can try to compile them individual doing it should look like:

```
> make PDSOURCE=/usr/local/src/pd/src/ iemmatrix
```

to locate the error or go inside the external folder and follow the documentation there. If there is still errors please file a bug report.

If you want additional but optional libraries like oggcast execute

```
> make PDSOURCE=/usr/local/src/pd/src/ oggcast
```

Afterwards you should have ready to configure CUBEmixer see next step.

Configuration of system and start scripts

Before you can start the CUBEmixer you have to adjust the start script of your instance with the configuration values in the start scripts to adapt them to your hardware and requirements.

Now you have following directory structure filed with files:

```
CUBEmixer /bin ... starting scripts
          /doc ... documentations
          /etc ... configuration files
          /lib ... Patches and dataversion
          /lib/libs ... here the externals and abstraction of libraries
          /sounds ... test sounds
          /src ... sources for externals
          /tools ... helper tools
          some files like README.txt, License, ...
```

There are two possibilities, run CUBEmixer as is or use it as framework. For both you first have to configure and calibrate the CUBEmixer. For a first attempt we use the first option, the other will be discussed separately.

Configuration PD Startup Script

All configuration data are in the *etc* folder, so look in there and make a copy of *LocalVars.template.sh* and rename it to *LocalVars.sh* and open it in your favorite editor.

> **Explanation:** The main purpose is, that with starting CUBEmixer, the DSP-Engine and GUI-Application is started with a separate PD task, using the configuration in here. Therefore three configuration files are loaded in this order:
>
> ```
> "${ETC}/LocalVars.template.sh"
> "${ETC}/configured.sh"
> "${ETC}/LocalVars.sh"
> ```
>
> where ${ETC} can be changed in any of this for the next file. So with each file the other settings as variables are overwritten. In a standard setting *LocalVars.sh* is not really needed, but since systems differ, also settings differ. "configured.sh" will be generated later and these variables are used as a standard.

You can remove the commented introduction (comments starts with ##) in the first part, execpt the first line and write your description of the the system:

```
## configuration for the MKLave CUBEmixer
##
## DSP Engine: ADCs 40 channels over jack and no midi
## done w.ritsch may 2009
##
## Changes:
##
```

In the section "general settings regarding PD" and probably "GUI-part of the CUBEmixer" can all be commeted out (or deleted)

```
##
## general settings regarding PD
##
#PD_INSTALL=src/pd/bin
#PD_OPTIONS=
#PD_PATH="-path .:lib:lib/abs/:lib/libs/:lib/plugins/:lib/extensions/:lib/GUI/:lib/DSP/:lib/GUI/abs:lib/DSP/abs:lib/libs/iemabs:lib/libs/zexyabs"
#ETC_PATH="-path etc/CUBEmixer"
#PD_LIB="-lib zexy:iemlib1:iemlib2:iemmatrix:iem_ambi:iem_bin_ambi:iemgui:iem_spec2"

##
## settings regarding only the GUI-part of the CUBEmixer
##
#GUI_PATCH="lib/GUI/GUI16+OUT.pd"
#GUI_AUDIO="-nosound"
#GUI_MIDI="-nomidi"
#GUI_OPTIONS="-nrt"
# GUI_PATH=
# GUI_LIB=
#GUI_MESSAGE=""
```

In the " DSP-part of the CUBEmixer" you should only change youre audio setting and comment others out eg.:

```
##
## settings regarding only the DSP-part of the CUBEmixer
##
##DSP_PATCH="lib/DSP/DSP+NET+MIDI+CUE.pd"
DSP_AUDIO="-jack -channels 40 -r 44100 -audiobuf 10"
DSP_MIDI="-nomidi"
#DSP_OPTIONS="-rt"
#DSP_PATH=
#DSP_LIB=
#DSP_MESSAGE=""
```

For other section probably added in new versions, please see the comments in there.

Now we can test the start-script and start CUBEmixer with *CUBEmixer.sh* in the *bin* folder, just click on it or over a shell application:

```
bin/CUBEmixer.sh
```

Now The GUI should pop up in recent time and also a second patch with the comment DSP-Engine. After some seconds on the GUI the ON/OFF button should become red, a sign that the GUI connected to the DSP-Engine. Now you can click in the System Box of the GUI "really" and short afterwards "quit", to quit the CUBEmixer.

If not you can start CUBEmixerDSP.sh and CUBEmixerGUI.sh independently to see where the failures occur.

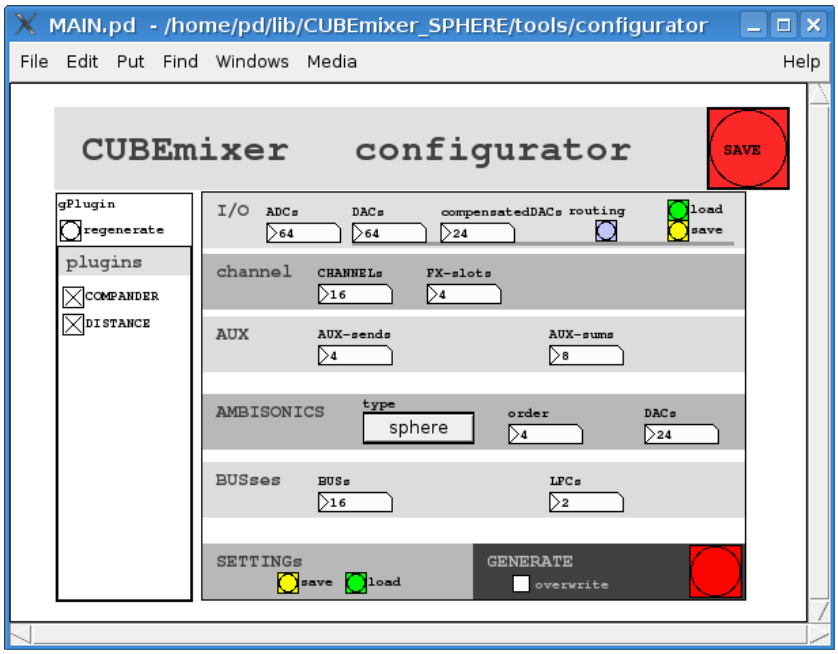
> **Explanation**: The scripts can also be used to write several own starting script, defining the variables, if it is needed for an application which uses the CUBEmixer. So Artwork can start the CUBEmixer as a part of them, either using the GUI or the own GUI.

Construct the CUBEmixer with the Configurator

With the usage of the Configurator you can construct your instance of the CCUBEmixer, scaling and setting of elements.

> **Explanation:** Since each instance of the CUBEmixer is on different Hardware it has to be configured, like different count of ADCs or DACs, different Ambisonics mode, number of Buses Aux and Channels, etc. A forst attempt was to make a dynamic patching, this means in PD, create objects dynamicly. I has shown, that this was error-prone, since if you edit one dynamicly generated patch, then it can be completely wasted next time loaded (see the PD-Community site for more information and a lot of discussion about this issue).
>
> We choose to use an additional PD-Patch to generate the needed other abstractions, costumized.When generated once, they can be edited or changed. At the time of writing there is only a limited support for the graphical GUI part, so better use not to exodic values.

The Configurator is located under the *tools* directory of the CUBEmixer folder and can be started with a script in the *bin* folder *configurator.sh*. It looks like this:



Configure your CUBEmixer

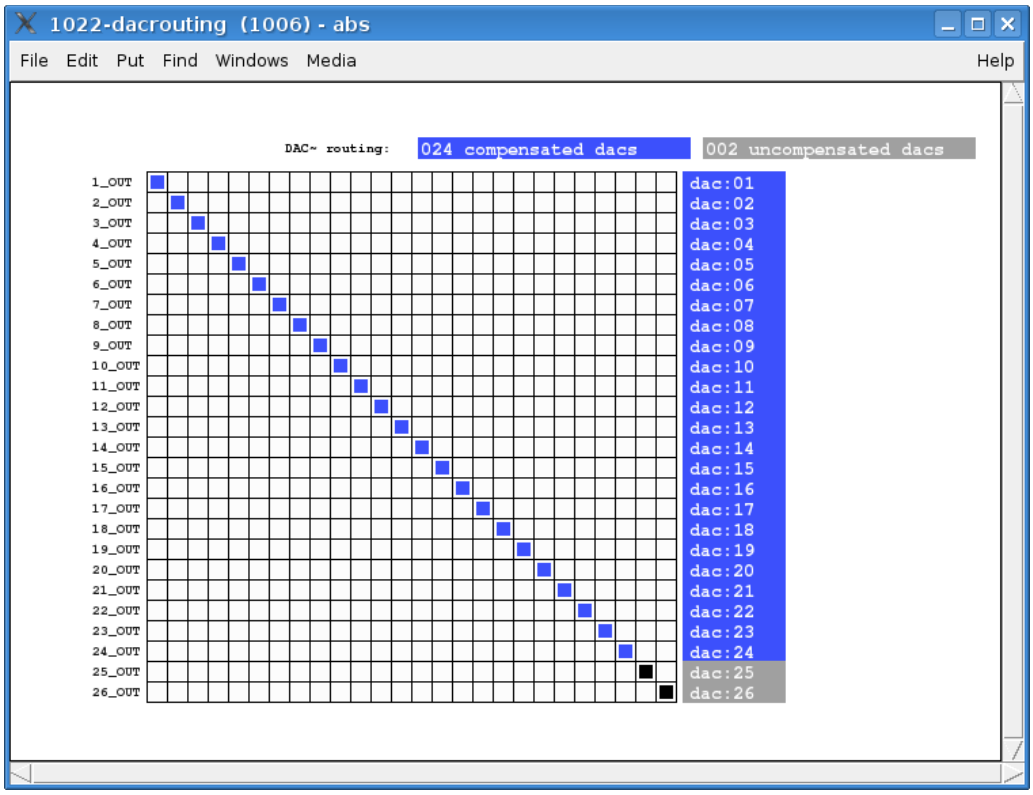
1. Plugin Structure

On the left side is a list of already available plugins for channel strips, check the one you want and press regenerate. To Save the generated plugins structure press the red *SAVE* on top of the patch

2. Input Outputs

On the right box the first row is that you tell the CUBEmixer how much ADC~ and DAC~ you want to use. This strongly depends on your hardware. In this case I have two RME cards with 26 channels and 18 channels so I choose 44 inchannels and 44 outchannels. This should correspond to your settings for PD, you have done before. *compensated* means, that a amount of channels can be calibrated by the Calibrator tool or by hand editing a configuration file. This must be less den DAC channels. I suggest use as much as you want to use for the Ambisonics Mixer, since this channels should be compensated, here I use 29.

All OUTs are numbered from 1..N, but on some system you want to renumber them because of routing. Also the first M channels can be compensated, so if you want channels with higher DAC-numbers for your Ambisonics bus, you can route OUTs to actual DACs in following dialog:



Note: Everytime you restart the dialog, all routing is reseted. Afterwards you can load and save routing in an seperate file.

3. AUX system

AUX sends means the amount of AUX controllers for each channel strip, *AUX sums* means how much AUX-Buses should be used, which have a seperate controller. This is because any *AUX send* controller can choose a *AUX-sum*.

4. Ambisonics

This row is the for choosing and adjusting the parameters in the Ambisonics domain.

type
: This is the Ambisonics type to use. For 2D Ambisonics choose *circ* (circular), for 3D Ambisonics either *hemi* for a hemisphere or *sphere* for a full sphere with speakers also below ceiling. See next section for more details on Ambisonics domain.

order
: The order which can be choosen also selects the amount of speakers which is minimal needed. If you have a rather regular arrangement you can come near this minimum else increase it in next step.

DACs
: The number of DACs used for this order, which can be bigger than the suggested amount and not more then available DACs. I should correspond to the number of compensated DACs, but have not.

5. bus system

BUSES are the additional buses used parallel to Ambisonics or other renderer. The can be mapped to DACs via a matrix mixer. *LFCs* are the number of Low-Frequency outputs, which normally drive sub-woofers.

6. Settings

Settings of the Configurator can be saved and loaded for future changes, but this is optional and not needed.

7. Generation

Now generate the the patches for the CUBEmixer, pressing the red button on bottom right. If you repeat this step, you have to choose *overwrite* and the previous generated patches are overwritten.

If you get some errors like this in the PD Console, ignore this, next time it should disappear later:

```
error: [folder_list] nothing found for "../../lib/DSP/DSPMAIN_ADC40_DAC40_IN16_FX4_AUX4_8_BUS16_circ2D11O24.pd"
error: [folder_list] nothing found for "../../lib/DSP/IN/ADC/44~.pd"
[...]
error: [folder_list] nothing found for "../../lib/../etc/CUBEmixer/configured.sh"
```

Your are done and you can start the *Cubemixer.sh* again, to see what has changed.

> **Explanation:** in the lib folders the needed patches are generated and the *etc/CUBEmixer/configured.sh* will have the needed arguments so the right GUIs and DSP patches will be loaded. Please be sure that the variables in *configured.sh* are not overwritten by your customized shell script.

Decoder matrix creation

How you can calculate your decoder matrix with the amdecalc tool. You need this to have a working Ambisonics system

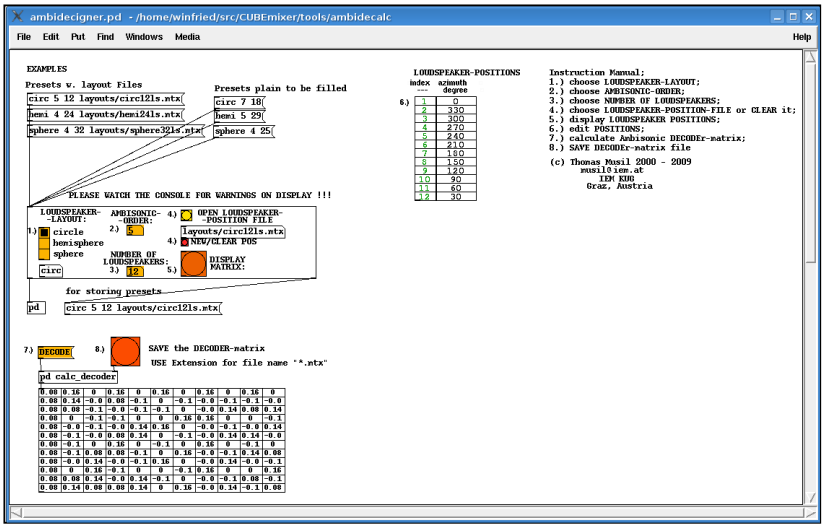
The calculation of a good decoder matrix for Ambisonics is not a standardized task, since there is a lot of discussion going on this topic and many ways proposed to solve it. Anyway a good decoder matrix should be done manually by an expert knowing the rehearsal situation, constrains and the target application for the Ambisonics system.

Anyhow we want to get our CUBEmixer work at least for a first start, afterwards later, we can go optimizing the matrix out of the experience we made. So if you need a rather perfect decoder matrix only few experts worldwide are known to do this right, among them Thomas Musil at IEM, who wrote this Ambidecalc tool.

It seems, in our experience, a 2D matrix can be calculated quite perfect with this tool, on an 3D-Ambisionics solution, a equal distribution of speakers helps a lot, and we mostly tested the hemisphere, since a complete speaker sphere is hard to find.

> **Explanation:** The patch uses a external of the Ambisonics PD library from IEM and tries to do a kind of inversion of the speaker matrix, which cannot be solved for every setting and sometimes leads to large or very low numbers (singularities).

The Ambidecalc patch is located under the *tools* directory of the CUBEmixer folder and can be started with a script in the *bin* folder *bin/ambidecalc.sh*. It looks like this:



In the dialog you can just follow the steps numbered. Here some hints.

Step 1-5 Choose your your Ambisonics domain

circle
: 2D Ambisonics means speakers are aligned on an circle (or small cylinder) so only one angle is relevant for the position of the speaker.

hemisphere
: 3D Ambisonics using only the upper side of the sphere, the two angles azimuth and elevation for each speaker has to be entered

sphere
: 3D Ambisonics using only the whole sphere, the two angles azimuth and elevation for each speaker has to be entered.

With the Ambisonics-Order you also choose the minimum number of speakers needed. You can afterwards increase the number of speaker your system uses, also this is known as room-oversampling. There is some discussion, sweet spot will increase, but some also that it also distort the Ambisonics behavior. Just

test it and hear the results and add a comment below.

NOTE: use the same order specified in the Configurator before !

Step 6 Enter your layout

Afterwards open one preset file or prepared file or click *New/Clear* and enter your values left

The text-file can be a preset (see top of dialog) or you choose one with angles like for 2D 12-speaker:

```
#matrix 12 1
0
330
300
...
```

or 3D 11 speaker:

```
#matrix 11 2
0 0
0 60
0 120
...
30 45
30 135
...
```

where the synthax is for 2D:

```
#matrix <rows N> 1
 <azimuth spk1>
 <azimuth spk2>
 ...
 <azimuth spkN>
```

Format for 3D:

```
#matrix <rows N> 2
 <elevation spk1> <azimuth spk1>
 <elevation spk2> <azimuth spk2>
 ...
 <elevation spkN> <azimuth spkN>
```

Step 7-8 Decode and Save

Here you have to look at the PD console, if there is:

```
ambi2_decode_hemi ERROR: matrix_inverse singular !!!!
```

then you should change the speaker layout or do some tricks with virtual speakers since there is no decoder matrix. Then press again and you should find a solution where you read:

```
ambi2_decode_hemi OK: matrix_inverse regular
```

Then have a look at the numbers, if there are some very high (greater than 100) or very low exept *-0.0* then you have a rather instable decoder matrix, so change the layout and try again. See Notes above about good decoder matrixes.

After this save the decoder matrix with a name with extension *.mtx* to youre working directory or I prefer in the *etc/CUBEmixer* folder. The matrix you want to use copy to the *etc/CUBEmixer/decoder.mtx*

So you are done and can test the matrix after doing the compensation next section.

Tricks

sphere instead of hemisphere

For a hemisphere with slighly negativ elevation, you can also use a sphere implementation. For speaker not present, since they would be under the floor, just put some of them as virtual speakers in the layout. After decoding just delete the lines for additional speakers in the *decoder.mtx* and set the dimension to real speakers.

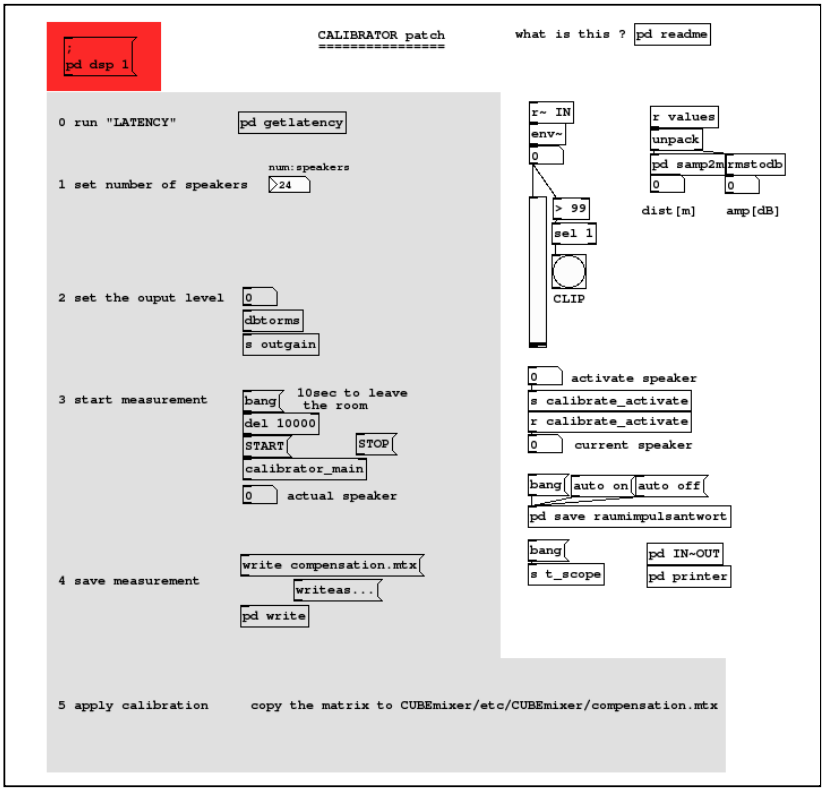
Calibration of Ambisonics Speakers

For a nice working system you should calibrate your speakers. Therefore the calibrator tool can be used.

A perfect calibration of the speaker system is a dream of most sound engineers, but there is no perfect world out there, so it is the CUBEmixer. One very important parameter is the sound level of each speaker, another the delay of sound towards the middle of the room. Other parameters like frequency response, impulse response and directive characteristics are also important for the quality of the whole system, but this is not implemented now, because there is no optimal solution and this differs very much on the application of the system. We asume all speakers are rather good equalized and produce a plane wavefront.

Therefore as a compensation has been integrated doing volume and delay adjusts on each dedicated DAC-Channels (see setup). Calibration is targeted for the center of the sweet spot. This can be done with the *Calibrator* tool using an autocorrelation analysis with noise

This step is discussed in the tool *calibration* there. Start it with executing *bin/Calibrator.sh*. Click on *pd readme* there for further instructions.



Calibration can also be done by hand, measuring the distcances between center and speaker, and with a level meter creating an calibration file.

Hint: the ear is always a good measurement tool.

DISCUSSION: In our experience a perfect calibrated system with distance compensation lead to phasing effects in the center. For a *sweeter* sweet spot, sometimes it is nice to add to this values random numbers to get a better distribution of this effects. Also suggested was, to use a burst to be folded with the signals, but this is not integrated by now in the CUBEmixer. Please add your experience with comments here.**

First start

Here a short preview for running the CUBEmixer is done.

Starting CUBEmixer

Like shown in previous section, the CUBEmixer can be run just starting *CUBEmixer.sh*. Afterwards wait for some seconds until the red ON/OFF button gets full red and turn on CUBEmixer. Look at the *cpu* and *peak* meter below, the should not become very high. They show a measured CPU-usage by the CUBEmixer.

After that choose a channel and unlock it in the channel view, select as input *test-tone* and turn on the channel. Afterwards there should be level indication some signal is on the input. Afterwards choose *Spatialisation* "Ambisonics" and increase the Volume on the channel and the Ambisonics System Master

Volume in the Master-Section. Also checking *24* and then gently increase the *Master* volume. Now you should hear some noise. You can move it around with the Ambisonics Pan View.

Alternatively you can load a test-setting in the System Dialog, which can be opened in the system view with the open button.

Congratulation, you successfully have install the CUBEmixer, have fun an please add some comment on sucess or failure in the comments.

Arguments and messages at CUBEmixer launch

You can Start the CUBEmixer with following command line args:

```
CUBEmixer.sh  [dsp_patch] [gui_patch] [dsp_path] [gui_path]

dsp_patch ... external DSP-patch
dsp_patch ... external DSP Patch
dsp_path  ... external DSP path
dsp_patch ... external DSP path
```

These can be used to have additional patches for GUI and DSP Application using CUBEmixer.

Also you can name a message to send to PD when started in yout *etc/CUBEmixer/LocalVars.sh*. This can used for example to load a specific paramter set or initial some variable, like the Mastervolume for automatic installations.

Further Extension possibilities

There will be another Tutorial for using CUBEmixer as framework and one for writing plugins and Extensions. Plugins are in the channelstrips and Extensions are Mastersections or Tools for the whole CUBEmixer. There will be hopefully an increasing number of plugins and Extension in future, so it is crucial to strictly use the convention documentated to stay relativly update save.

Anyhow, much fun with the CUBEmixer and do use the possibility for comments here.