HuFo System

None

None

None

Table of contents

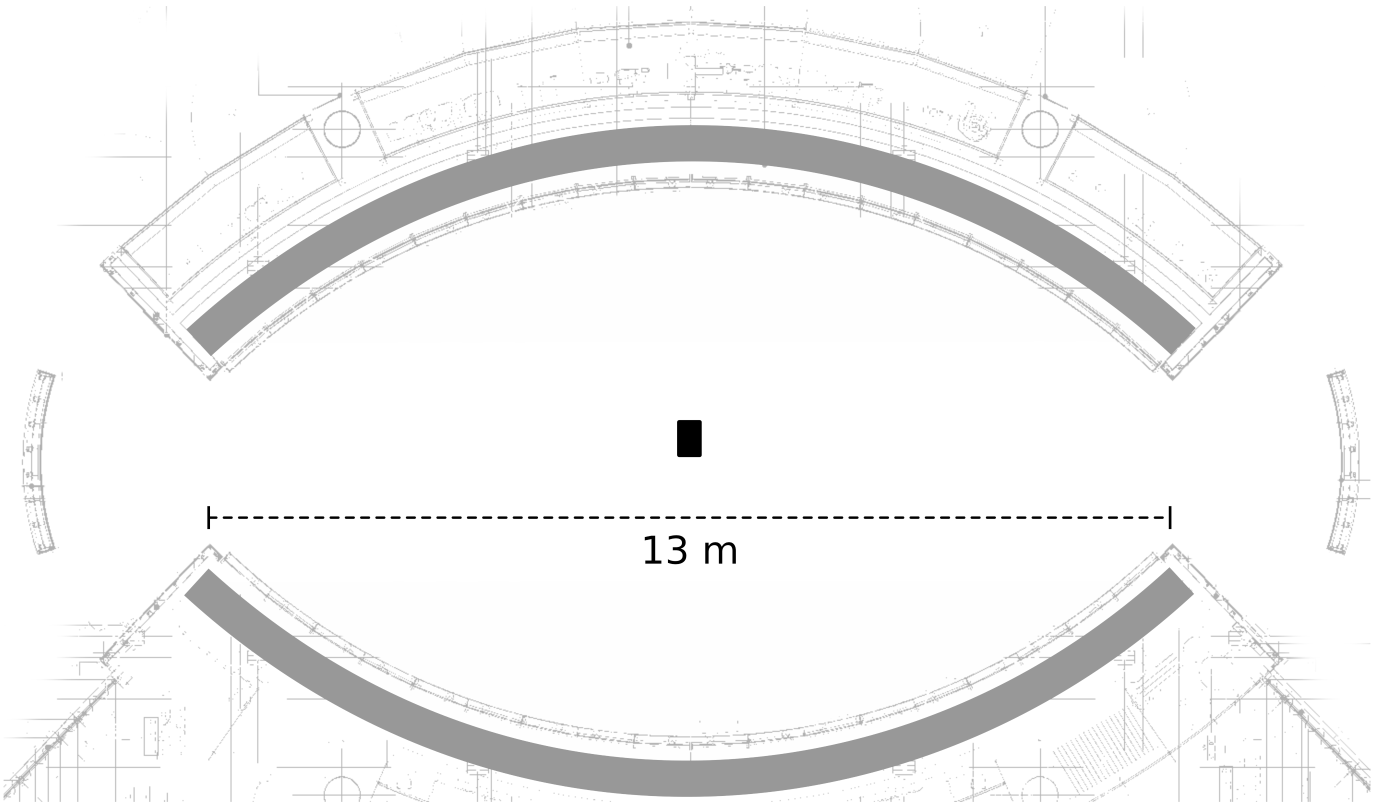
1. About the System

1.1 Concept

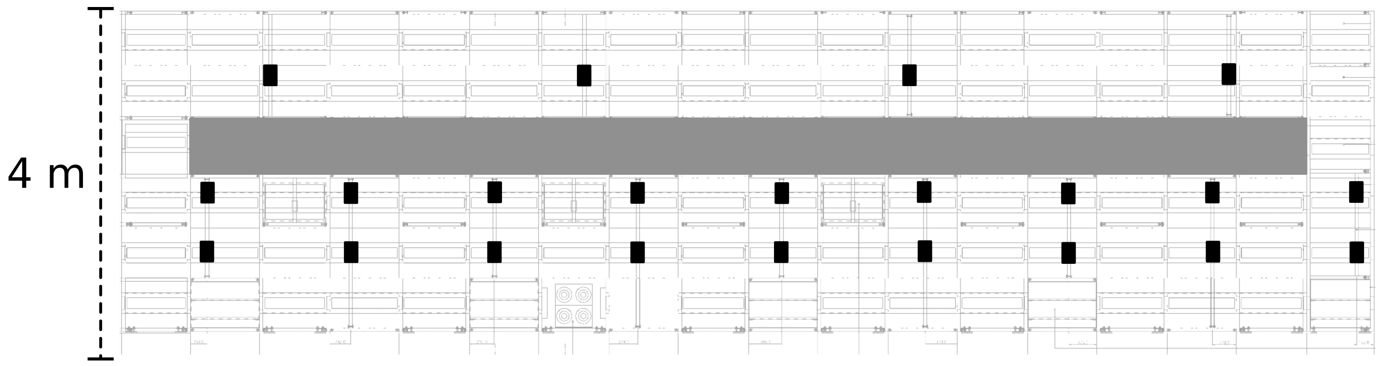
The Ethnomusicology Museum at the newly built Humboldt Forum features a listening room for immersive illustration of relevant auditory content. It is unique in that it combines multiple methods for sound field synthesis, mainly Wave Field Synthesis (WFS) and Higher Order Ambisonics (HOA), thus providing an enhanced spatial experience. A dedicated software system, developed by the Audiocommunication Group at TU Berlin, allows a seamless integration of these methods.

1.2 Loudspeaker Configuration

The listening room features 32 WFS panels by Four Audio. This results in a total of 256 WFS channels, using 768 tweeters and 64 woofers. Both arcs hold 16 panels each, mounted as a continuous ribbon above head height with a horizontal speaker distance of 10 cm. 45 Genelec 8020C speakers are arranged in three levels on each arc, used for Ambisonics rendering.



Top view of the listening room with the entrances (left,right) and two arcs, each with a ribbon of WFS panels (gray).

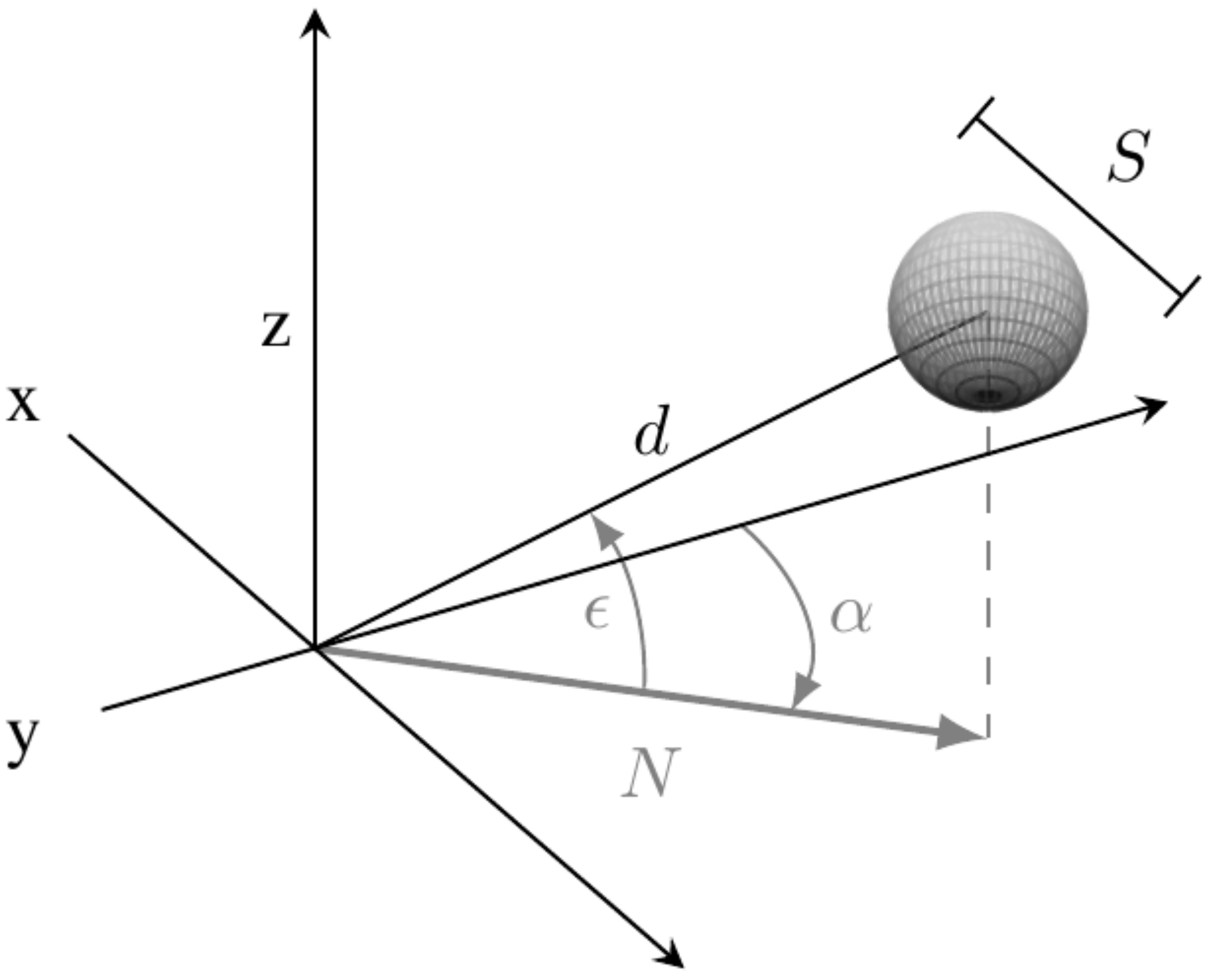


Side view of a single arc with the ribbon of WFS panels (gray) and the Ambisonics speakers (black).

1.3 Fundamentals of Spatialization

The Virtual Sound Source

The combined system works with so called *virtual sound sources*, also referred to as the object oriented approach of sound spatialization. The figure below shows the model of such such a virtual sound source, which is defined by its position, using two angles, the distance and a source width.



Every sound event in a project or composition can be connected to such a virtual sound source, allowing to place it in the listening space. Depending on the preferred workflow, different tools can be used for a dynamic arrangement of the individual sound sources. The provided tools in Reaper grant a quick entry by providing control through simple automation trajectories in a DAW project.

The Send-System

In addition to the position of the virtual sound source, the proposed approach also allows the continuous fade of sound events between the Ambisonics and the WFS rendering through a send system.

2. Working with Reaper

The most accessible way of using the HuFo system is a single DAW session from a laptop, just as working with any other multi-track audio project. Reaper is used as the standard software in most projects and recommended. Other solutions are possible but not fully tested. Each audio track in a Reaper project is directly routed to the system with DANTE and connected to a virtual sound source. Source position and additional properties can be controlled and automated as any other parameter. This workflow ensures that content can be prepared in advance and ported to the system afterwards.

2.1 Audio-Sources in general

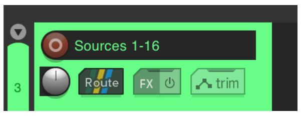
An audio-source in a spatial audio-system is defined by the audio content and the source attributes, like the position of the source. Depending on the used rendering-module in the HuFo-System (WFS, Ambisonics, Reverb) additional attributes can be defined, e.g. *plane wave* and *Doppler effect* for the WFS module.

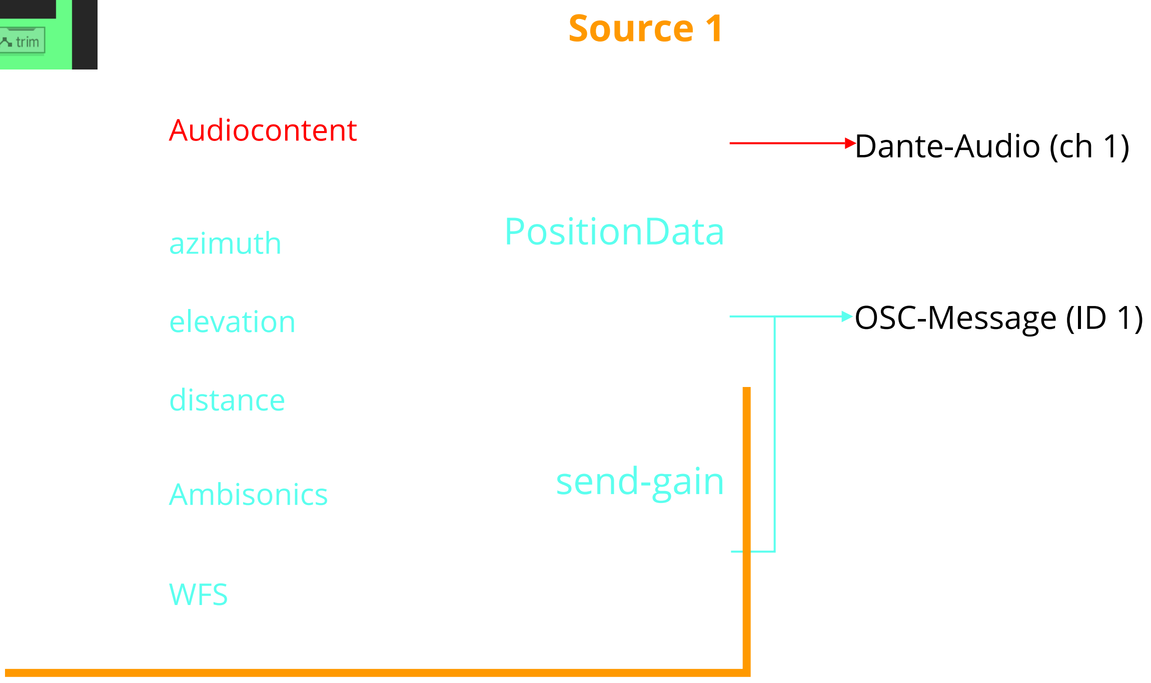
The rendering modules are running parallel for all sources. Which rendering module is used can be determined with the additional "send to" parameters, which control the amount of the audio-signal being sent to a specific rendering module. In that way a fades from on to another system as well as mixtures can be realized.

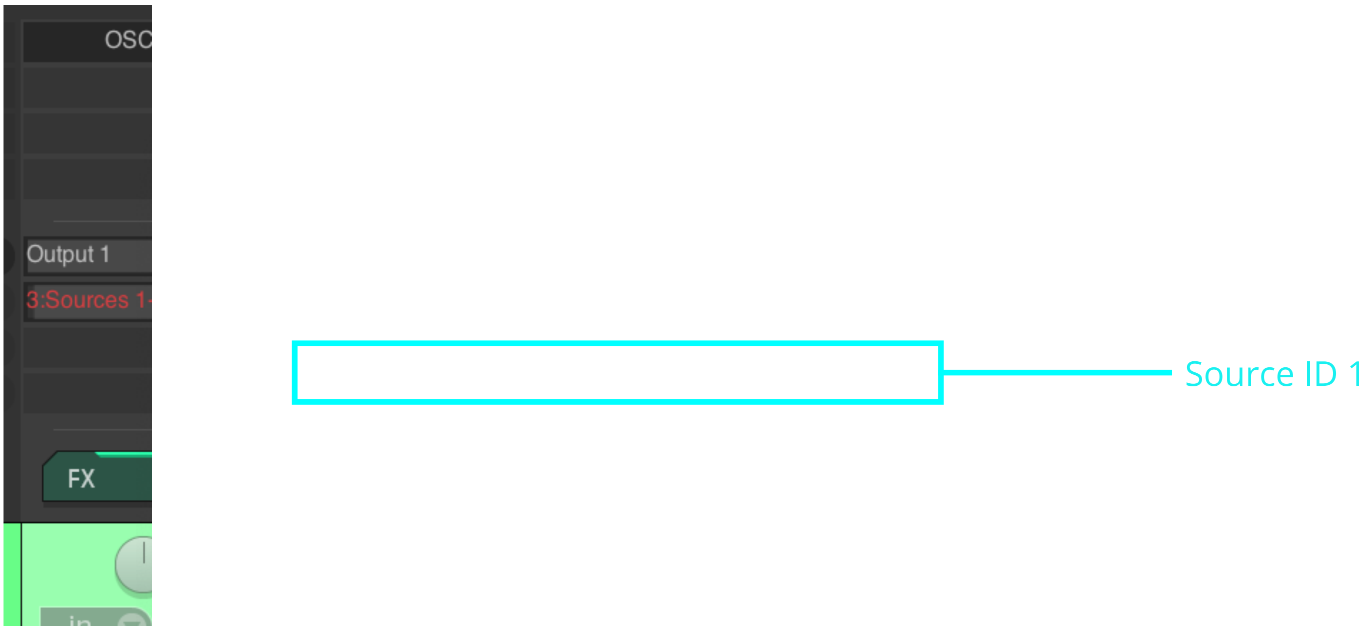
By definition all Audio-Sources are Mono-Audio sources and best results in spatial precision are achieved by sending all audio-element to a single audio source. Of course stereo or other multichannel material can be used too, by using audio sources as virtual speakers which are rendered by the system.

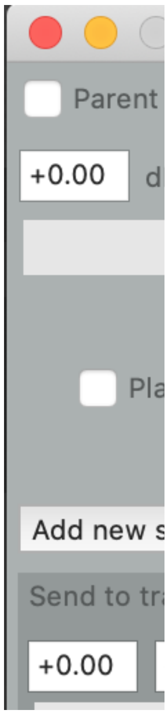
2.2 Audio Sources in REAPER

For defining an audio-source in REAPER, a Plugin is used for sending the source attributes via OSC synchronized to the audio playback (here OSCar by IRCAM is used). Every audio source is identified by an ID (1-64) which is set in the plugin and corresponds to the channel the audio is sent out. This also means every instance of the plugins represents one audio source. The output of the audio tracks should be set be sent to a mono channel, either as direct hardware output or if you are using bus-systems to a mono channel on the bus-track.


Sources 1-16
3
Route
FX
trim

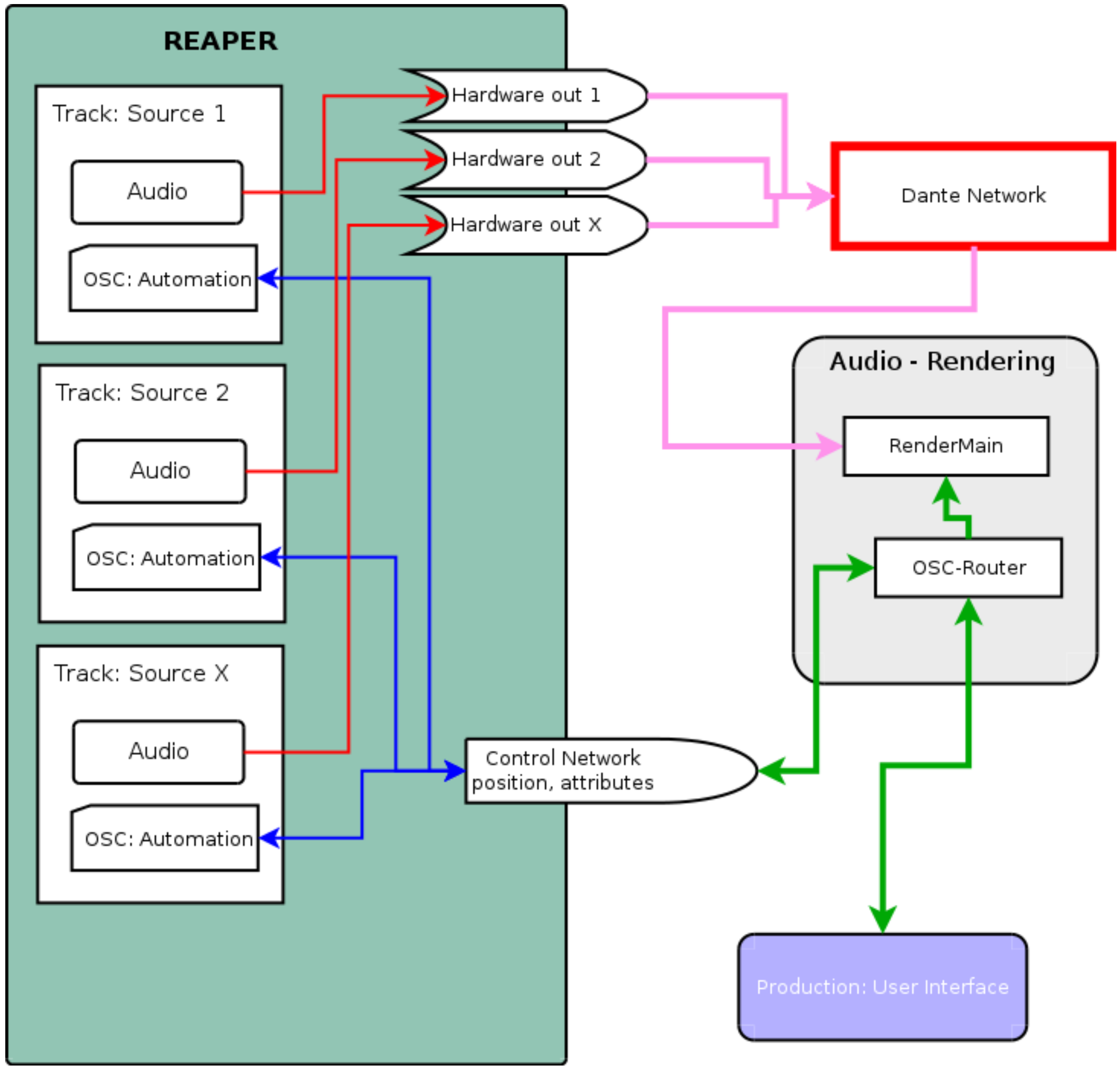

Source 1
Audiocontent
Dante-Audio (ch 1)
azimuth
PositionData
elevation
OSC-Message (ID 1)
distance
Ambisonics
send-gain
WFS


OSC
Output 1
3:Sources 1-
FX
in
Source ID 1



MONO
Hardware Out 1

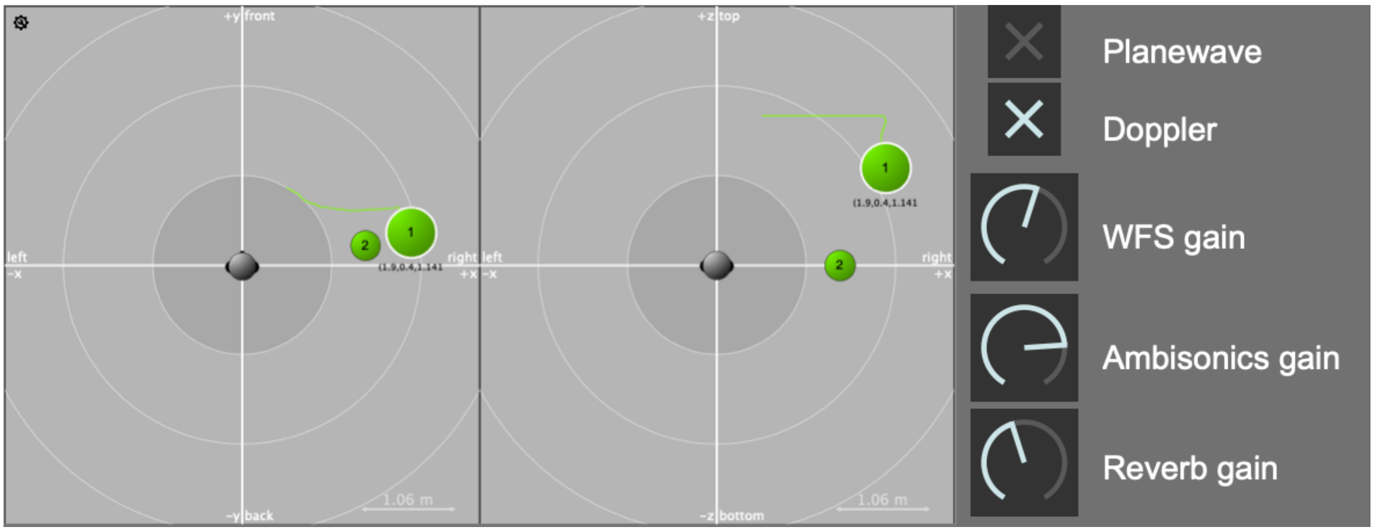
2.3 Content Production Process



In the production process the automation data can either be written with the pencil tool inside of REAPER. Also a dedicated interface-client or hardware controllers can be used, which connect also to the central OSC router. The OSC messages are then distributed to the REAPER-plugin where it can be written using the Reaper's automation modes.

Since it is not defined which state the sources have at the beginning of the playback all source-attributes must be set at the beginning of the project.

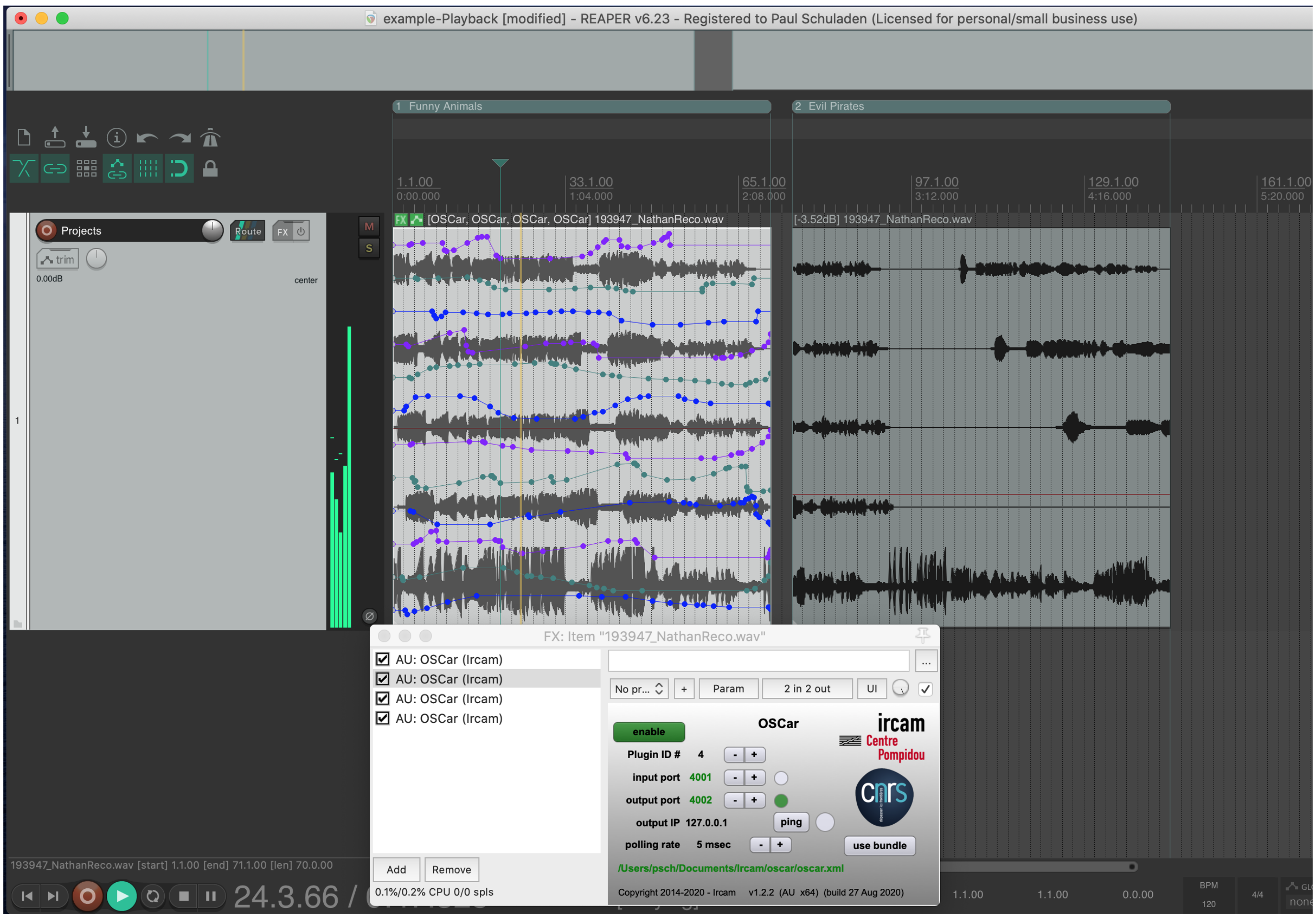
For more information about the OSC-Data have a look in the chapter OSC-Router.



2.4 Content Playback

The result of a production is a multichannel audio file in which the plugin automation data will be embedded with REAPER's Take FX function. This file including the automations will be saved in Reaper Project Bay from it then can be pasted and arranged in a playback project.

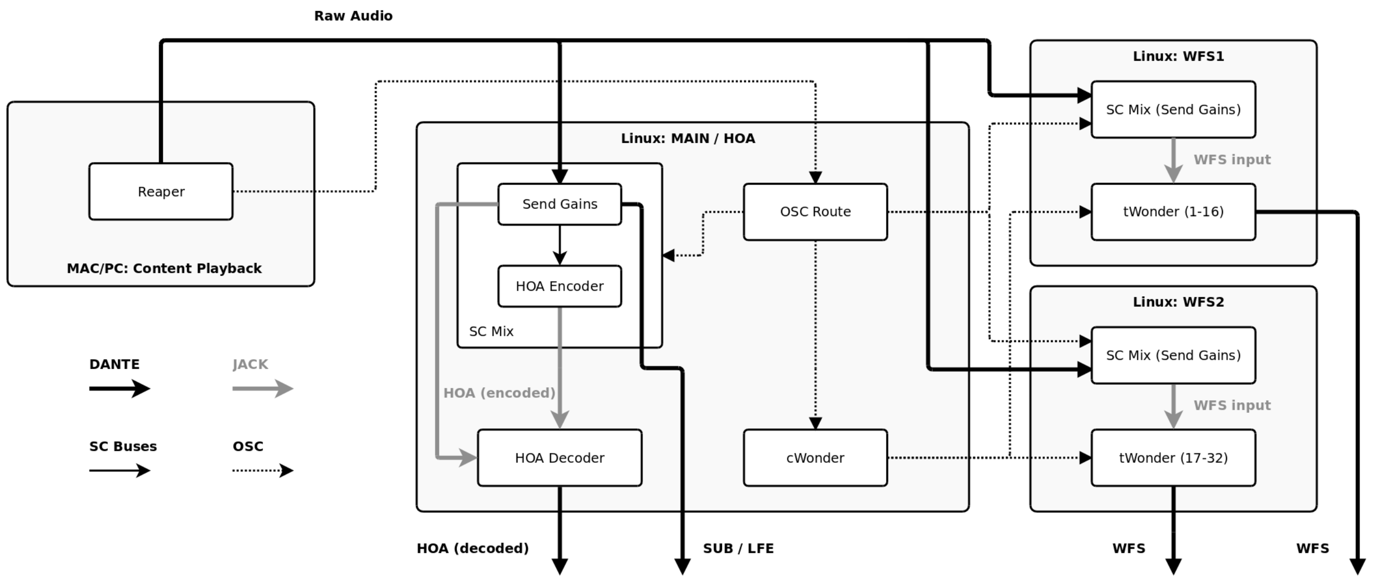
For the project-export a Reaperscript is provided which automates the rendering and data-embedding process.


example-Playback [modified] - REAPER v6.23 - Registered to Paul Schuladen (Licensed for personal/small business use)
1 Funny Animals
2 Evil Pirates
Projects
[OSCar, OSCar, OSCar, OSCar] 193947_NathanReco.wav
[-3.52dB] 193947_NathanReco.wav
FX: Item "193947_NathanReco.wav"
AU: OSCar (Ircam)
AU: OSCar (Ircam)
AU: OSCar (Ircam)
AU: OSCar (Ircam)
Add
Remove
0.1%/0.2% CPU 0/0 spls
OSCar
enable
Plugin ID # 4
input port 4001
output port 4002
output IP 127.0.0.1
ping
polling rate 5 msec
use bundle
/Users/psch/Documents/Ircam/oscar/oscar.xml
Copyright 2014-2020 - Ircam v1.2.2 (AU x64) (build 27 Aug 2020)
ircam Centre Pompidou
193947_NathanReco.wav [start] 1.1.00 [end] 71.1.00 [len] 70.0.00
24.3.66

3. Technical Documentation

3.1 Signal Flow

The following flow chart visualizes the interconnection between the software components. Scroll down to get the exact input and output assignment.



3.1.1 Input-Output Assignment

3.1.2 Linux: MAIN (DANTE)

Inputs	Source
1-64	Source channels from media machine.

Outputs	Target
1-22	DAC 1 (Ambisonics speakers 1-22)
23-44	DAC 2 (Ambisonics speakers 23-44)
45	DAC 1 (Ambisonics spekaer 45)
46-47	DAC 1 (SUB 1-2)
48-49	DAC 2 (SUB 3-4)

3.2 SuperCollider

Mixing, signal distribution and Ambisonics encoding is based on SuperCollider. Two main scripts are used:

3.2.1 HuFo_SERVER.scd

HuFo_SERVER is a mixing and distribution instance for the source signals, as well as for Ambisonics encoding.

SC IO Assignment

The following assignment is only valid for the 3rd order configuration:

Inputs	Description
00-63	raw audio input

Outputs	Description
00-63	WFS input channels
64-79	HOA encoded
80	reverb send
81	sub send

3.2.2 HuFo_WFS.scd

HuFo_WFS is a mixing and distribution instance for the source signals, limited to WFS for.

Both instances are take the same optional input arguments:

```
$ sclang HuFo_SERVER.scd <N_inputs> <server_port>
$ sclang HuFo_WFS.scd <N_inputs> <server_port>
```

Defaults:

- N_inputs: 64
- server_port: 58010

3.2.3 Setting Up

Build Decoders

Decoders in ADT have a problem for more than speakers 24? speakers. Use standalone decoder in that case.

Install Decoders

Copy the directory `decoders` to the SC user extension directory. It can be obtained in SC:

`Platform.userExtensionDir`

For larger loudspeaker setups (<24), standalone Jack decoders are used.

3.2.4 SuperCollider OSC Paths

The sueprcollider mixer listens to several OSC messages for signal routing and conrolling positions

Sends

TO AMBISONICS + WFS

`/source/send/spatial i i f`

- first argument = channel ID
- second argument = system ID
- 0 = Ambisonics (inside SC)
- 1 = WFS (external, WONDER)

TO SUBWOOFER

`/source/send/direct i i f`

- first argument = channel ID
- second argument = output ID
- 0 = Subwoofer

AMBISONICS SPATIAL CONTROL

`/source/pos/azim i f`

- first argument = channel ID
- second argument = angle (rad)

`/source/pos/elev i f`

- first argument = channel ID
- second argument = angle (rad)

`/source/pos/dist i f`

- first argument = channel ID
- second argument = distance (meters)

`/source/pos/aed i f f f`

- first argument = channel ID
- second argument = azimuth (degree)
- third argument = elevation (degree)
- fourth argument = distance (meters)

3.3 OSC-Router and Processor

The OSC-Router written in python serves as central interface for OSC-messages in a multiclient spatial rendering environment. It automatically translates incoming OSC-messages to match the expected formats of the rendering-engines and distributes it to all connected clients including UI-Clients and Data-clients for automation data. (Here Ircams OSCar).

3.3.1 OSC-Router config

The OSC-Router is started by executing the `MAIN_oscrouter.py`. Per default it loads the configurationfile `oscRouterConfig.txt` in the same folder. With the argument `--config` another file can be loaded.

The configfile consists of a number of blocks which are divided with `***`. For more information have a look at `configSampe.txt`

It provides an additional debug option: using the option `--oscdebug` followed by a string with ip and port (e.g. `--oscdebug "192.168.3.2 55112"`) an additional OSC-listener can be defined, which receives a copy of every OSC-message sent out by the OSC-router.

3.3.2 accepted OSC messages

The Osc-Router accepts various message formats and take care of converting them to the right format. Every OSC-Message consists of a OSC-Address Prefix followed by a number of values. Here we only work with integer- (i) and float-values (f)

Source IDs begins with Index 1 and can go up to 64. The following examples demonstrate which OSC-Messages are accepted for position data.

- Source ID can be in the OSC-Prefix `/source/1/aed f f f` with f f f = *azimuth elevation distance*
- Or part of the Message-Values `/source/xyz i f f f` with i f f f = *sID x y z*
- The last component of the OSC-Prefix defines the coordinate format. Besides full sets of coordinates single coordinate values can be send too `/source/azim i f` , `/source/1/x f`
- as well as pairs `/source/1/ad f f` , `/source/xy i f f`

Polar and Cartesian formats will automatically converted according to the target renderer.

A **gain-send** value has to be given which is the amount of audio sent to a specific rendering module (e.g. Ambisonics, WFS).

`/source/send/spatial i i f` with i i f = *sourceID renderingID gain(linear)*

A **direct-send** to a speaker can be given:

'/source/send/direct i i f' with i i f = *sourceID outputID gain(linear)* (0 = Subwoofer)

Settubg render-specific attributes:

- `/source/doppler i i` with i i = *sourceID 0/1*
- `/source/planewvae i i` with i i = *sourceID 0/1*

3.4 Setting up the System

3.4.1 Python Dependencies

```
pip3 install oscpy
```

3.4.2 SuperCollider

1: Build and install SC

2: Build a headless version of SC:

Follow https://github.com/supercollider/supercollider/blob/develop/README_LINUX.md but call cmake:

```
cmake -DCMAKE_BUILD_TYPE=Release -DNATIVE=ON -DSC_EL=no -DSC_ABLETON_LINK=off -NO_X11=ON -DSC_QT=OFF ..
```

Install SC3-Plugins:

```
https://github.com/supercollider/sc3-plugins
```

Install SC-HOA classes:

```
Quarks.install("https://github.com/florian-grond/SC-HOA")
```

3.4.3 Developer Mode

The scripts `supercollider_gui.sh` and `supercollider_nogui.sh` allow to switch between the standard SC install and the headless version by setting symlinks in `/usr/local/bin`

Once the standard version is lionked, `scide` can be started for development and debugging.

3.4.4 System Services

copy all services to:

```
/usr/lib/systemd/system
```

3.4.5 Disable Automatic Updates

Automatic updates are not desired, since they can change the system behavior. In addition can disable the DANTE drivers.

Edit:

```
/etc/apt/apt.conf.d/20auto-upgrades
```

It needs to be:

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Download-Upgradeable-Packages "0";
APT::Periodic::AutocleanInterval "0";
APT::Periodic::Unattended-Upgrade "1";
```