Spatial audio OSC control

Dresden SyncID container

Introduction

You can control spatial audio inside HTW container lan either manually by directly controlling Reaper DAW with mouse and keyboard OR you can do that programmatically with Open Sound Protocol (OSC for short). OSC is widely used in the audio industry and its successor to MIDI protocol. OSC ought to be more universal and offer more precision.

When conducting experiments inside the container you may want to have interactive control of spatial audio or change audio sources positions via algorithms.Video rendering and light control can be done with other software but you might want to keep in sync with audio. To achieve that you can send OSC commands from software of your choice to spatial audio setup. Below you will find examples on how to do that.

OSC is available as libraries or plugins for many software solutions. However if you find yourself missing that possibility contact me to provide a websocket or plain UDP proxy. That way you can still control spatial audio setup but instead of OSC you can use websockets or UDP packets - I am currently working on that.

In computer game development it is common to have separate engines for the game and the audio. We are using the same concept. This allows musicians to use all the tools available for DAWs and do not limit the interactive control. It also allows to separate the audio processing development and interactivity development.

Turning it on

1. Turn on all speakers
2. Turn on Hapi
3. Turn on Anubis
4. Turn on PC
5. Open Reaper DAW project (in documents folder find folder called reaper - inside double click the project)
6. Make sure the tracks you want to use are not muted and ones you don't are muted
7. Press play button if needed (not necessary for microphone / line inputs)

Video overview of the Reaper DAW

https://youtu.be/dGiwkbu3KS0

Java OSC commands examples

Introduction

Even though the examples below are written in Java you should have no problem adapting them to your chosen programming language.

First you need to start OSC server and client:

```
oscP5 = new OscP5(this, 12000); //listen on port 12000 for incoming messages fromReaper DAW - optional, not necessary but useful for getting status back
myRemoteLocation = new NetAddress("127.0.0.1", 8000); //define the IP and port where you will send OSC commands
```

See general function description:

```
Void nameOfTheFuntion( typeOfVariable variableName, typeOfVariable variableName)
//note that osc message is always constructed from pattern name (text string) + value or values
 //declare variable type osc message with given text pattern
OscMessage myMessage = new OscMessage("/pattern");
myMessage.add(variableName); //add some value into the message - order matters!
oscP5.send(myMessage, myRemoteLocation); //send message to Reaper DAW
}
```

Code excerpts

```
//control spatial audio VST plugin

//set room size
//should be set once at the start of the program for all tracks
void setRoomSize(float x, float y, float z, int track ) {
  OscMessage myMessage = new OscMessage("/track/"+track+"/fx/2/fxparam/5,6,7/value"); //
  myMessage.add(x);
  myMessage.add(y);
  myMessage.add(z);
```

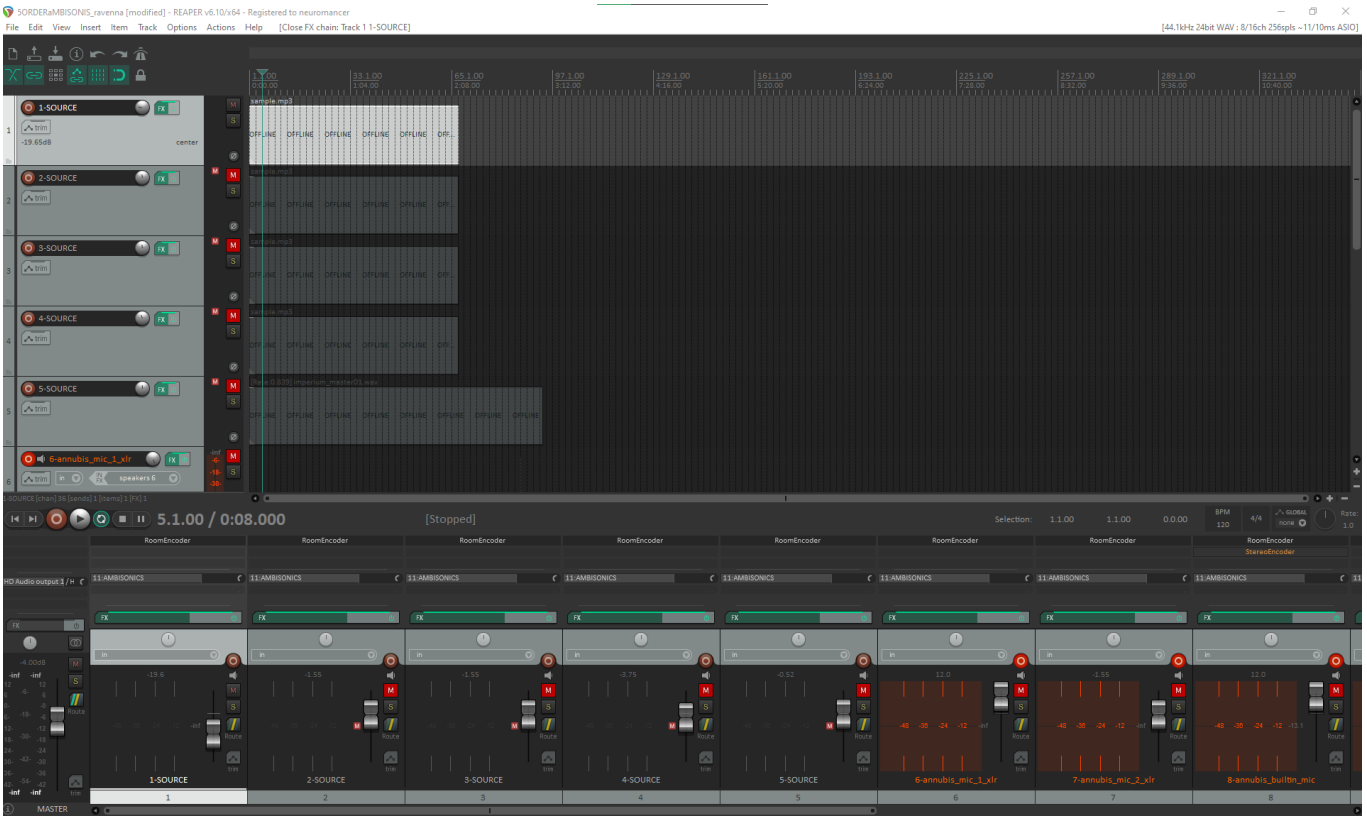
```
  oscP5.send(myMessage, myRemoteLocation);
}


//dynamically change audio source position for each audio source/track
void setSourcePosition(float x, float y, float z, int track ) {
  OscMessage myMessage = new OscMessage("/track/"+track+"/fx/2/fxparam/8,9,10/value"); //
  myMessage.add(x);
  myMessage.add(y);
  myMessage.add(z);
  oscP5.send(myMessage, myRemoteLocation);
}


//set listener position
//should be set once at the start of the program for all tracks and then kept in sync for all tracks
void setListenerPosition(float x, float y, float z, int track ) {
  OscMessage myMessage = new OscMessage("/track/"+track+"/fx/2/fxparam/11,12,13/value"); //
  myMessage.add(x);
  myMessage.add(y);
  myMessage.add(z);
  oscP5.send(myMessage, myRemoteLocation);
}


//General controls

//Adjust track volume
//use this to adjust volume of each audio source or ambisonics bus (all audio sources are send to
ambisonics track)
void volume(int in, float val) {
  OscMessage myMessage = new OscMessage("/track/"+in+"/volume");
  myMessage.add(val); /* add an int to the osc message */
  oscP5.send(myMessage, myRemoteLocation);
  println("volume set for track "+in+" value: "+val);
}


//adjust master volume
void masterVolume(float val) {
  OscMessage myMessage = new OscMessage("/master/volume"); //n/master/volume
s/master/volume/str
  myMessage.add(val); /* add an int to the osc message */
  oscP5.send(myMessage, myRemoteLocation);
  println("master volume: "+val);
}

//control reaper play button
void playAudio() {
  OscMessage myMessage = new OscMessage("/play");
  oscP5.send(myMessage, myRemoteLocation);
  println("play");
```

```
}


//control reaper stop button
void stopAudio() {
  OscMessage myMessage = new OscMessage("/stop");
  oscP5.send(myMessage, myRemoteLocation);
  println("stop");
}


//control reaper pause button
void pauseAudio() {
  OscMessage myMessage = new OscMessage("/pause");
  oscP5.send(myMessage, myRemoteLocation);
  println("stop");
}


//trigger custom action (lua script)
void triggerAction(String actionID) {
  OscMessage myMessage = new OscMessage("/action");
  myMessage.add(actionID);//ID of the action (see in Reaper in Actions->Show Action List)
  oscP5.send(myMessage, myRemoteLocation);
  println("custom action trigerred");
}

//rename track
void renameTrack(String newname, int trackID) {
  OscMessage myMessage = new OscMessage("/track/"+trackID+"/name"); //rename first track
  myMessage.add(newname);
  oscP5.send(myMessage, myRemoteLocation);
  println("track renamed");
}


//Advanced functions

//load media file to given track - hot swap audio source
//this is actually combined function that requires other OSC calls and lua script in Reaper Scripts folder
void loadMediaFile(String filePath, int sourceIndex) {
  //first set path as a name of the track - later that will be red by lua script in reaper and used as a
variable
  renameTrack(filePath,1); //ie "C:/path/to/file.mp3"
  //volume acts as variable here - it will be used as a index of the track where we should place new
media file
  volume(0, sourceIndex);
  //trigger lua script inside reaper to load file based on first track name
  //script will read first track name and volume and use it as path and index variables
  triggerAction("_RS90361aec4e2b820549ce9f17feb68ac0b293fc22");
}
```

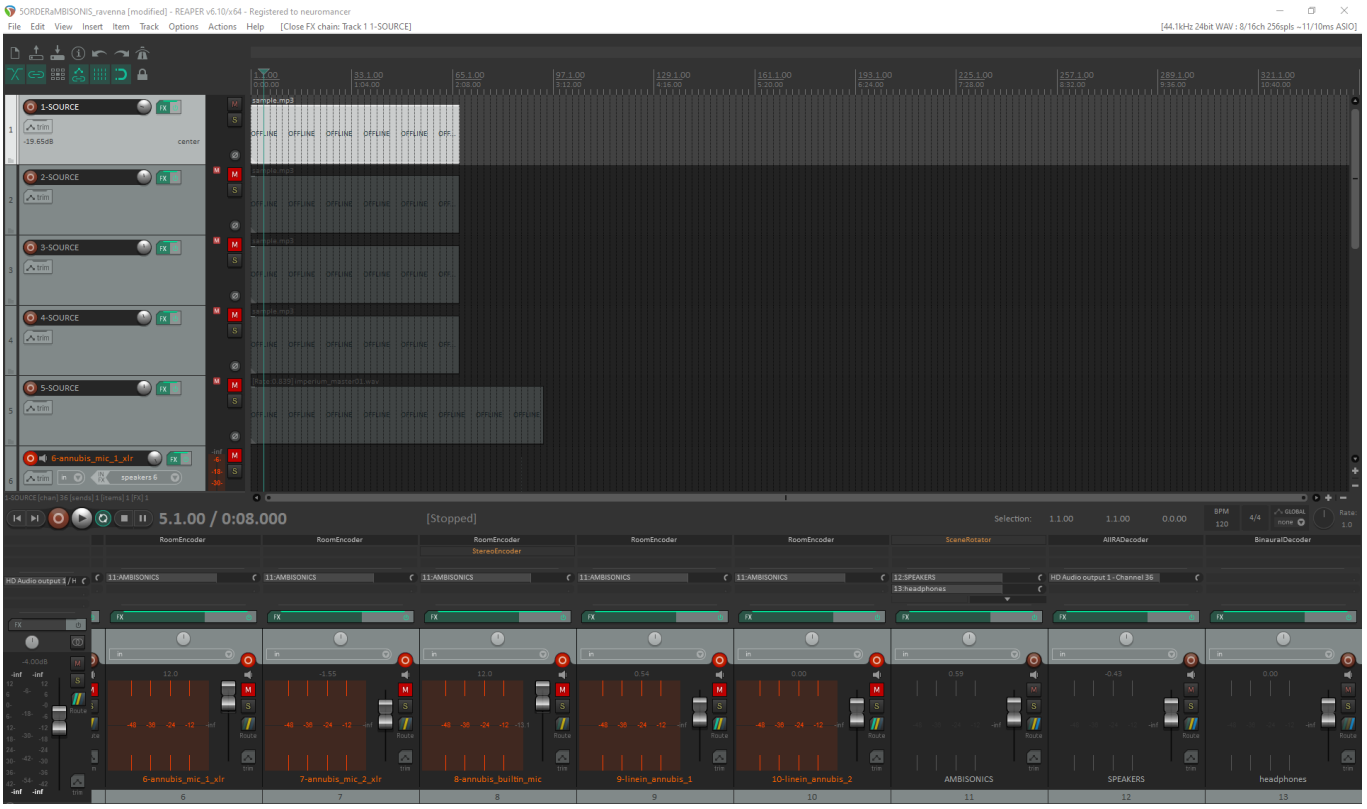
DAW overview

Below you can see the Reaper DAW interface. On the left and also on the bottom you can find all tracks listed. Some tracks represent audio sources (stereo 1-SOURCE to 5-SOURCE), microphone or line inputs of Annubis (mono), general AMBISONICS track and outputs - speakers and headphones track.

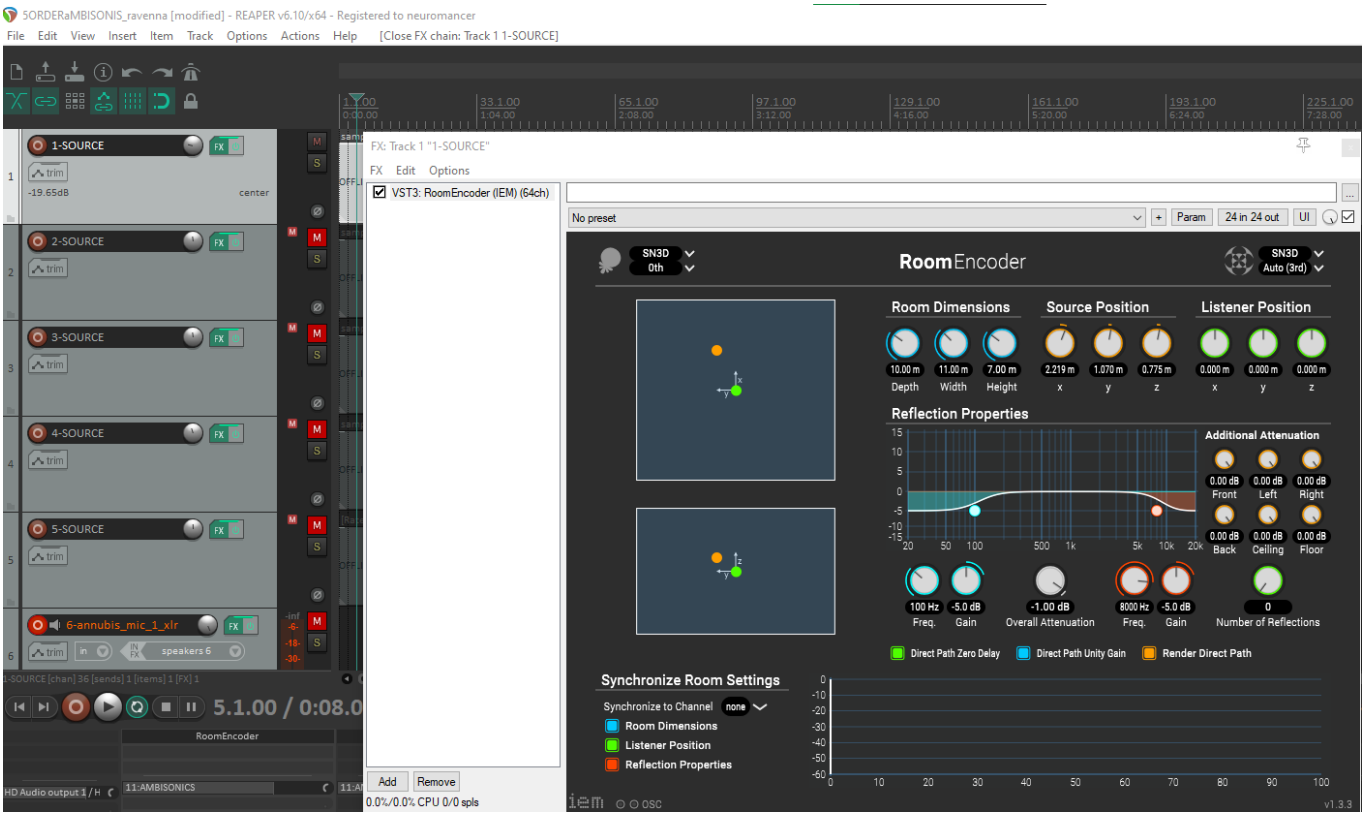
All sources are first converted from stereo or mono into 36 spatial audio and sent to ambisonics track. From ambisonics track the audio is rendered into speakers track with aiira decoder and also into headphone track as binaural stereo.



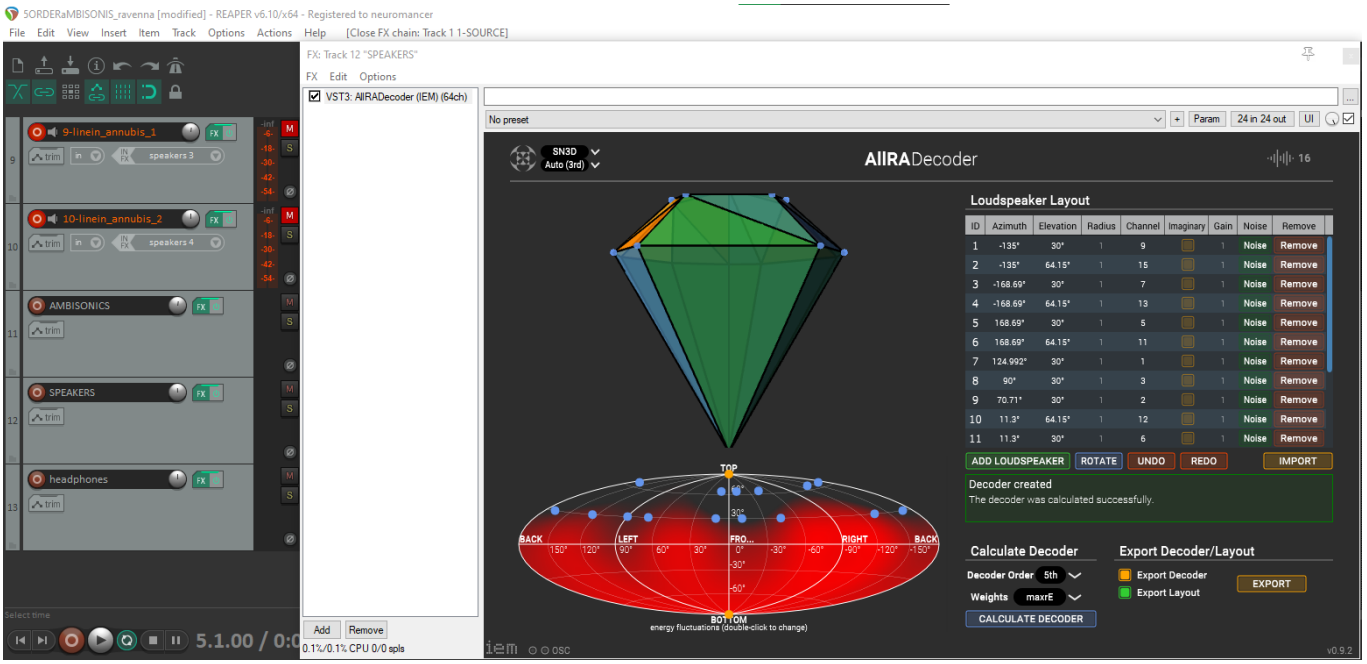
Screenshot of Reaper interface 1



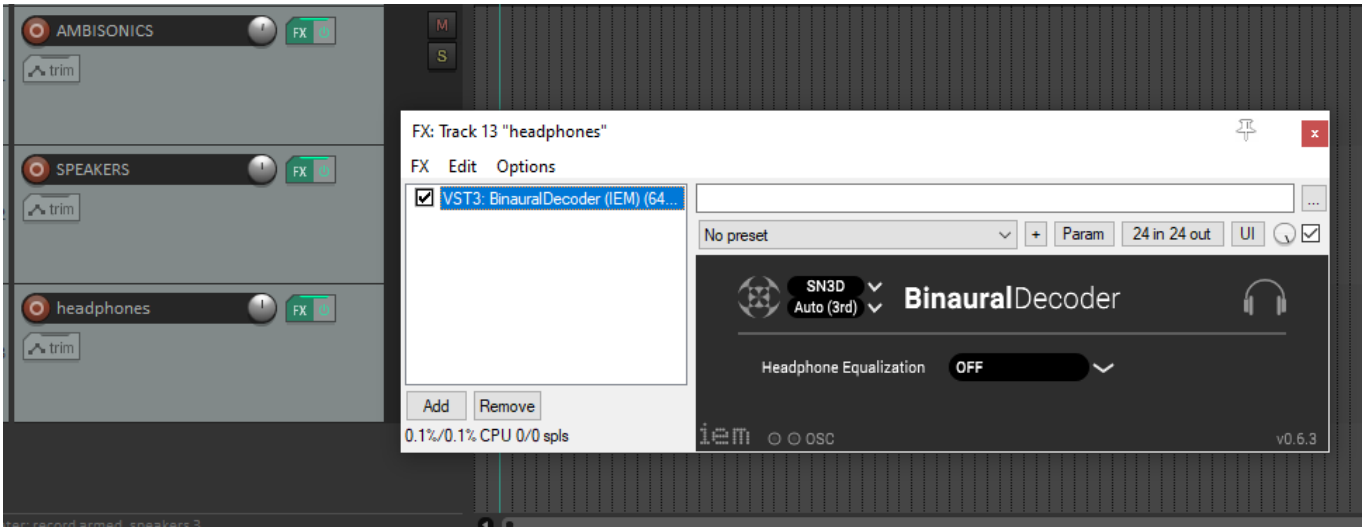
Screenshot of Reaper interface 2



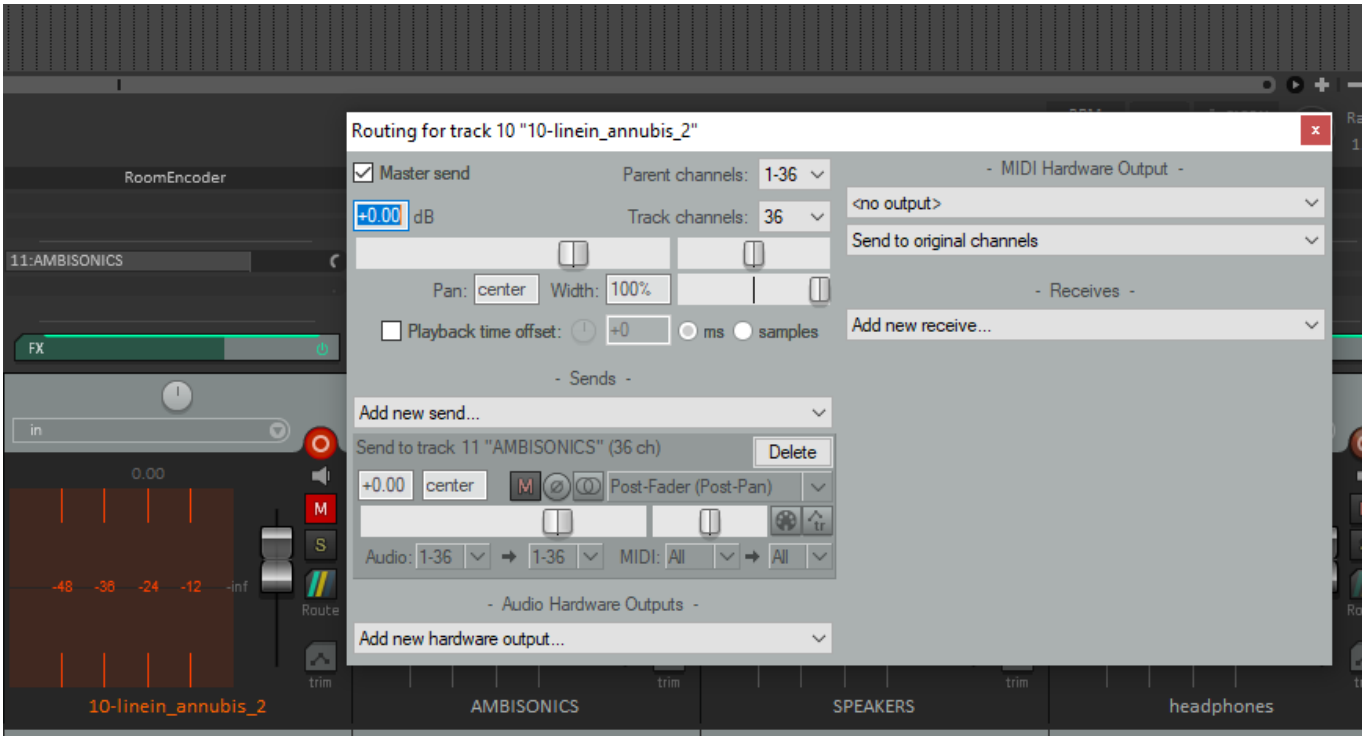
Stereo and mono sources are converted into 36 channel ambisonics with spatial audio VST called Room encoder - this is what you can adjust to change source or listener position.



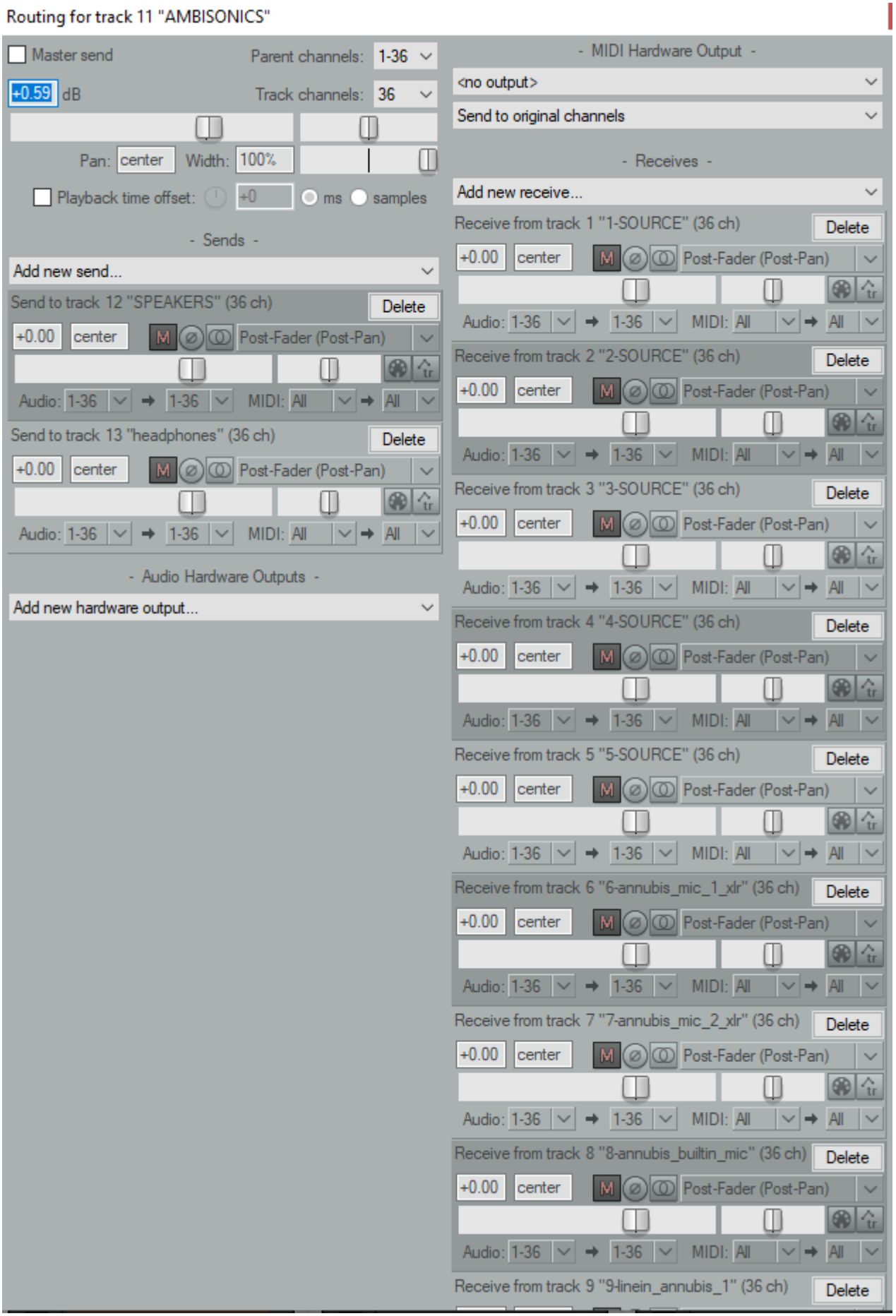
From ambisonics track audio is rendered into speakers with aiira VST decoder. Decoder is set to reflect physical speakers position and cabling order.



From ambisonics track audio is rendered into headphones with binaural VST decoder.



Each source is routed into an ambisonic track…



Ambisonics track includes all audio sources…

Troubleshoot

You can contact me at info@tricktheear.eu or consult with Jimmy jimmy.orawetz2@stud.htw-dresden.de who was observing the setup I have created and can help you to understand it. Please have a look at the video overview of the DAW Reaper setup and read this documentation first.