University of Erlangen-Nuremberg

Multimedia Communications and Signal Processing

Prof. Dr.-Ing. André Kaup

Introduction to the Sound Scape Renderer

B. Sc. Maximilian Schäfer

April 2014

Supervisor: Prof. Dr.-Ing. habil. Rudolf Rabenstein

Contents

1. Introduction **1**

2. The Audio-Laboratory **2**

3. Installation **4**

3.1. JACK 5

3.2. Ecasound 5

3.3. Sound Scape Renderer 6

3.4. QjackCTL 6

4. Soundcard Configuration **8**

5. Starting the System **10**

6. Using the SSR **14**

6.1. Example 14

7. Demos **16**

7.1. Demos by Maximilian Schäfer 16

7.2. Demos by Josef Rettelbach 17

7.3. Demos from DaFX14 17

7.4. Other Demos 18

A. Appendix **19**

A.1. Folder Structure 19

A.2. Known Issues 19
A.3. Short Instructions 21

References **22**

1. Introduction

This Report deals with the installation and using of the "Sound Scape Renderer (SSR)" in the Audio-Laboratory of the Chair of "Multimedia Communications and Signal Processing (LMS)" at the Friedrich-Alexander University Erlangen-Nürnberg.
The report describes which programs must be installed and how these programs work together to make the system complete for "Wave Field Synthesis"-Reproduction. The report gives a detailed description of the programs, an overview about the demos located in the audio laboratory and provides some short examples how to start a demo. [1, Chapter 3.1]

2. The Audio-Laboratory

The Audio-Laboritory is a laboratory at the Chair of Multimedia Communications and Signal Processing of the University Erlangen-Nürnberg. The room belongs to the group of Walter Kellermann at the Chair.

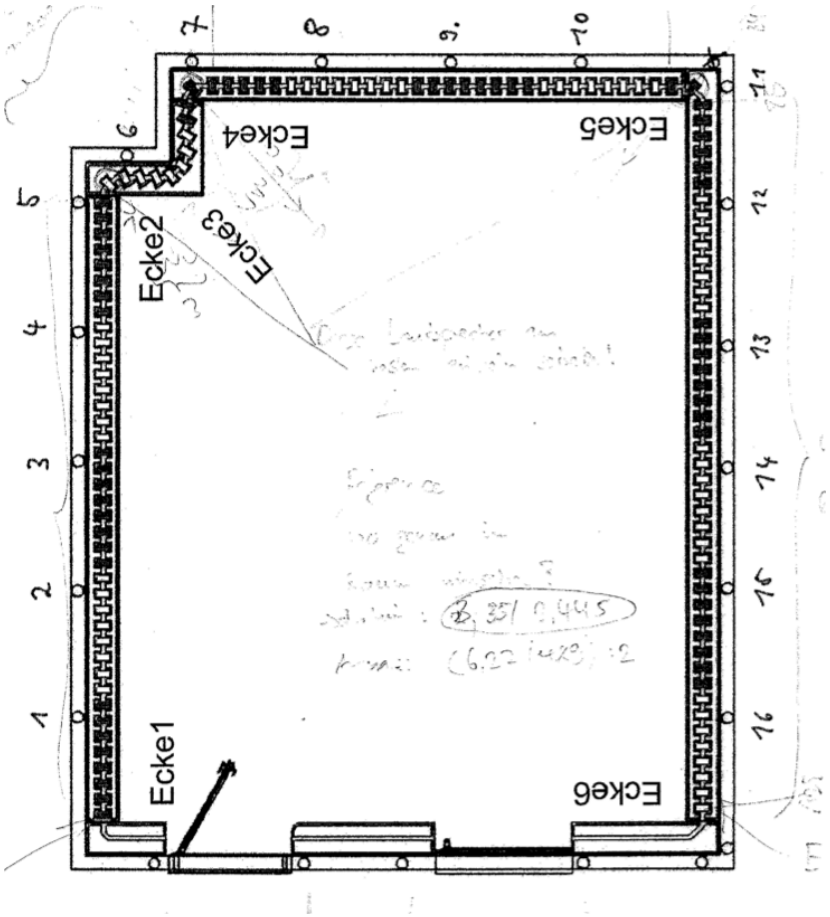

Figure 2.1.: Scheme of the audio laboratory

The Laboratory is built for spatial recording and reproduction, it is used for research in the area of Speech Signal Processing and Audio Signal Processing (Fig. 2.1).
The Room contains an array of 128 identical loudspeakers, which are splitted in 16 panels with 8 loudspeakers for each (Fig. 2.2). In addition the room has a variable acoustical performance, so if you need less echoes and reverberation in the room, you can take out plates of the ceiling or turn the wall elements around. These elements are on the one side reflective and on the other side absorbent.

Figure 2.2.: Picture a loudspeaker panel

3. Installation

To use the Sound Scape Renderer four programs must be installed on your operation System:

- `Jackd`
- `ecasound`
- `Sound Scape Renderer`
- `QjackCTL`

In Fig. 3.1 the connection between the single programs is shown.The `ssr` uses ecasound to create the audio files for each channel. `Ecasound` sends these audio files to the `jack-server` and then the signals are forwarded to the respective channels of the sound cards. From there they go to the amplifiers in the room and to the loudspeakers.

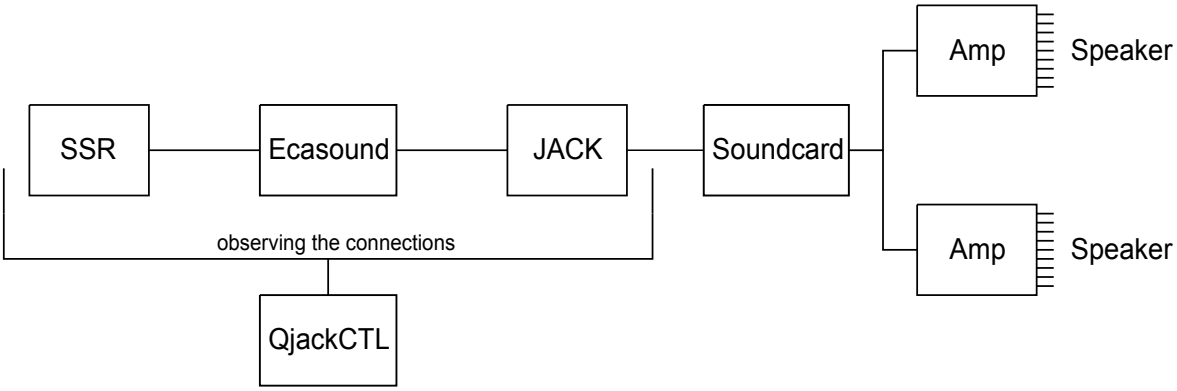

Figure 3.1.: Connection of the Software-Components

The `qjackctl` is used to observe all the connections between the `ssr`, `ecasound` and the `jack-server`. In the following chapter the different programs are described.

3.1. JACK

The design of JACK is focused on synchronous execution of all clients, and low latency operation. More detailed JACK is described in [3] as a "system for handling real-time, low latency audio (and MIDI). It runs on GNU/Linux, Solaris, FreeBSD, OS X and Windows (and can be ported to other POSIX-conformant platforms). It can connect a number of different applications to an audio device, as well as allowing them to share audio between themselves. Its clients can run in their own processes (i.e. as normal applications), or can they can run within the JACK server (i.e. as a "plugin"). JACK also has support for distributing audio processing across a network, both fast and reliable LANs as well as slower, less reliable WANs."([3]).
In the audio laboratory we are working with Version `1.9.9.5`.
In Figure 3.1 can be seen, that the `jack-server` is the direct connection between the audio- signals coming from `ecasound` and the Soundcard. How to start the JACK and how to work with it will be discussed in chapter 5.

3.2. Ecasound

`Ecasound` is responsible to create for each source in the `ssr` an output port on your system and send this information, including the correspondent audio signal to the `jack-server`. In [6] the ecasound audio processing software is described as "a software package designed for multitrack audio processing. It can be used for simple tasks like audio playback, recording and format conversions, as well as for multitrack effect processing, mixing, recording and signal recycling. Ecasound supports a wide range of audio inputs, outputs and effect algorithms. Effects and audio objects can be com-

bined in various ways, and their parameters can be controlled by operator objects like oscillators and MIDI-CCs. A versatile console mode user-interface is included in the package." ([6]) For further information you can visit [6].
In chapter 5 you find a detailed description how these things work together. On the system in the audio laboratory we are working with Version `2.8.1`.

3.3. Sound Scape Renderer

The spatial audio package Sound Scape Renderer (SSR) is described in [5] as " a tool for real-time spatial audio reproduction providing a variety of rendering algorithms, e.g. Wave Field Synthesis, Higher-Order Ambisonics and binaural techniques. The SSR is currently available for Linux and Mac OS X and has been released as open source software under the GNU General Public License (GPL). It was developed at the Quality and Usability Lab at TU Berlin (`http://qu.tu-berlin.de/`) and is now further developed at the Institut für Nachrichtentechnik at Universität Rostock (`http://www.int.uni-rostock.de/`) ([5])". For further information about the SSR-Project you can visit the project-page [5]. Before working with the `ssr` you should read the documentation of the `ssr` [1]. It contains a detailed description for installing and working with the `ssr`. On the system in the audio laboratory we are working with the Version `0.4.0-pre5` of the `ssr`.

3.4. QjackCTL

The QjackCtl application is described in [2] as "a simple Qt application to control the JACK sound server daemon, specific for the Linux Audio Desktop infrastructure ([2])." With `QjackCTL` you can control the `jack-server` and can control your system setup. So on the one hand you can control, wether the jack server is running or not and start and stop the server (3.2) and on the other hand you can check the connections of the `ssr` to the audio outputs (3.3).

Figure 3.2.: GUI of the `QjackCTL`

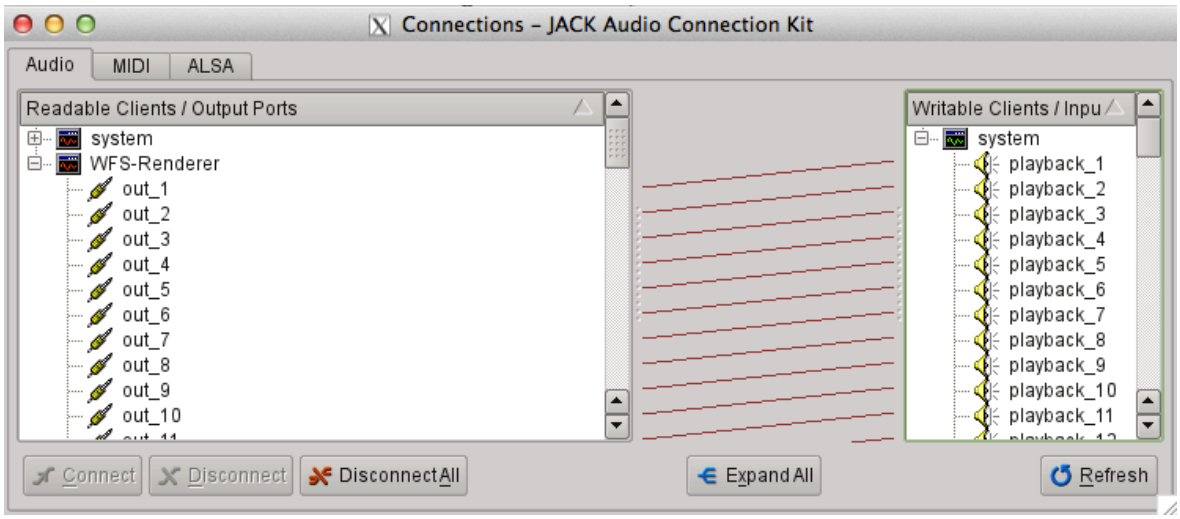

Figure 3.3.: GUI of the `QjackCTL`

4. Soundcard Configuration

In the audio laboratory of the LMS are built in 128 loudspeakers. To use all these loudspeakers we are working with two sound cards. Each sound card (`c2`, `c3`) has 64 channels. The jack server can handle only one sound card, so we defined a new sound card (`hdsp_128`) via the file `.asoundrc` which has to be placed in the home directory of the computer (`lnt217`).

Synchronisation:

The two sound cards have to be synchronized in the right way. At first the two sound cards must be connected at the backside of the computer. In the audio laboratory the sound cards are connected in the way described in figure 4.1. The next step is to check

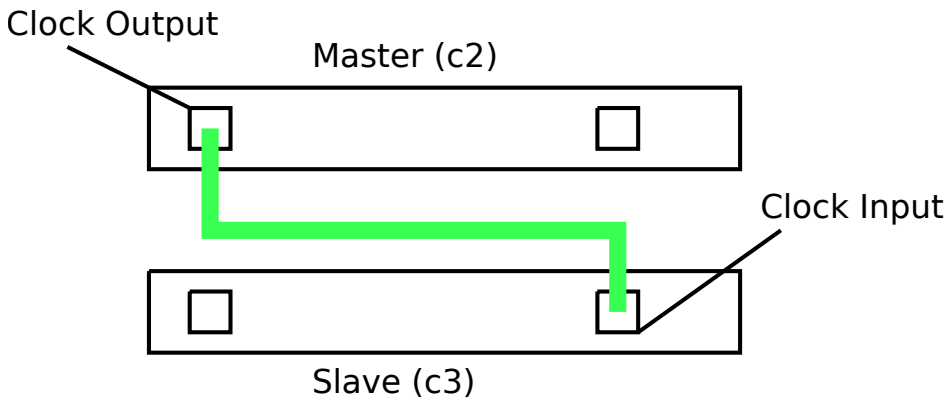

Figure 4.1.: Connection of the two sound cards (`c2`, `c3`) at the backside of the computer. Sound card `c2` is the Master giving the sync signal and sound card `c3` is the Slave, the receiver of the sync signal.

the right configuration in the mixer of the sound cards. To start the mixer of a sound card use the following command on the command line

```
alsamixer -(soundcard)
```

where `soundcard` is the name of the sound card you want to check. The GUI of the mixer can be seen in figure 4.2. In our configuration in the mixer of the master (`c2`) under the point `System C` it has to be `master`, and `Auto- Sync` in the mixer of the slave (`c3`).

So if the `jack` server won't start please check the sound cards.

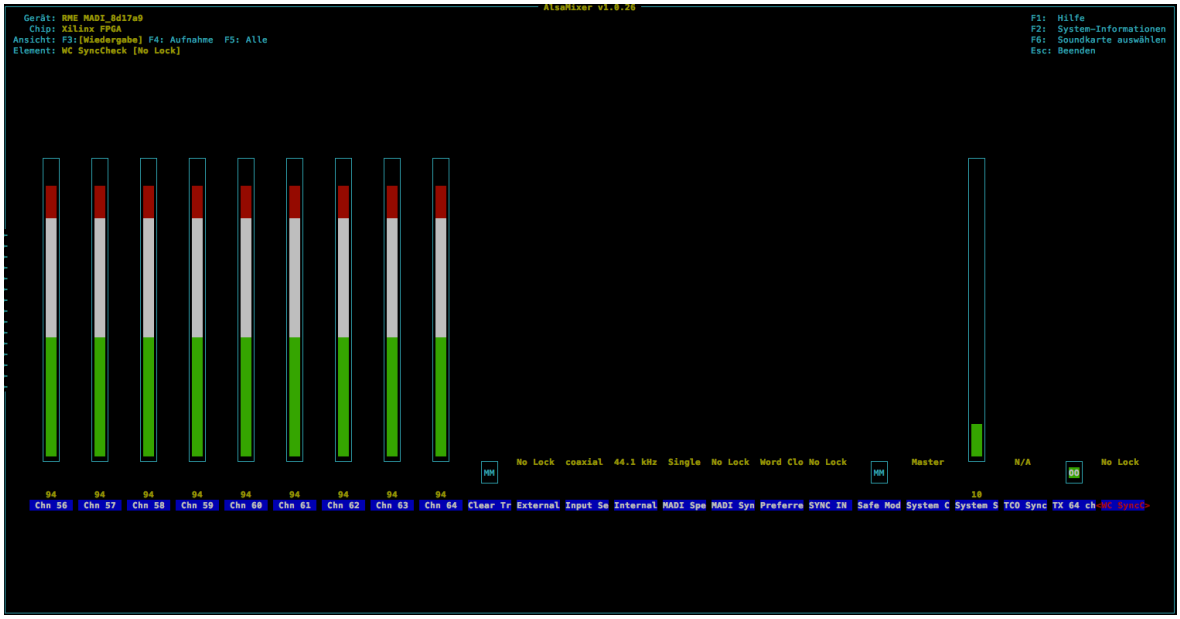

Figure 4.2.: GUI of the alsa mixer for sound card `c2`.

5. Starting the System

JACK-Server:

Before starting the system check the configuration of the sound cards described in chapter 4. After checking this, the first thing to start is the `jack` server via the terminal with the command:

```
jackd -d alsa -d soundcard -r 44100
```

Where the argument `soundcard` stands for the name of the sound card you want to use. For using all loudspeakers you should use the `hdsp_128`. The different arguments are:

- `-d alsa` is for selecting the audio driver
- `-d (soundcard)` is a alsa backend option to select the device, here the sound card
- `-r` is for deactivating the real-time mode of the sound card
- `44100` is the sampling frequency

There are more arguments to use `jackd` with other skills, for further information you can read the man page of `jackd`.

QjackCTL:

Second step is to start the `qjackctl` for controlling and observing the server and the `ssr`. To start use:

`qjackctl`

Sound Scape Renderer:

To start the `ssr` you can use different commands, depending on the application where you want to use the `ssr`. In general the command could look like:

`ssr --(mode) --setup (loudspeaker configuration) (ggf. audiofile/scene)`

Which command you have to use for the different modes of the `ssr` can be read in [1]. We are using the whole speaker setup, and we want to process wave field synthesis.

- `(mode)`: Is the mode you want to use, here `wfs`.
- `(loudspeaker configuration)`: Is the path to the loudspeaker configuration.
- `(audiofile/scene)`: If you want you can add a scene file or an audio file which is loaded while starting the `ssr`.

ASD-Files are files in a XML style which can be read by the `ssr`. ASD files define loudspeaker configurations or audio scenes. In the file for the loudspeaker configuration the number of loudspeakers and the positions of the loudspeakers are defined. Speakers can be defined as single speakers or as linear or circular arrays. A snippet of the config file for the audio laboratory at the LMS can be seen in the following code.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="asdf2html.xsl"?>
<asdf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="asdf.xsd"
      version="0.1">
    <header>
        <name>
        acoustic lab 128 loudspeaker setup
        </name>
        <description>
        Setup for the acoustic lab with 128 loudspeakers
        </description>
```

```
    </header>
    <reproduction_setup>
        <!-- linear array on the left side of the room with 40 loudspeakers -->
        <linear_array number="40" name="left linear array">
            <first>
                <position x="-3.135" y="2.7075" />
                <orientation azimuth="-90" />
            </first>
            <last>
                <position x="2.845" y="2.7075" />
            </last>
        </linear_array>
        ...
```

ASD can be used to describe audio scenes, which can be loaded by the `ssr`. In a audio scene you can place different sources in the room and give the sources different properties. The following code snippet is part of a ASD file for a simple audio demo with different sources.

```
<?xml version="1.0" encoding="utf-8"?> <?xml-stylesheet
type="text/xsl" href="asdf2html.xsl"?> <asdf
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="asdf.xsd"
    version="0.1">
    <header>
        <name>Augenblick@Solafari</name>
        <description>
            Studio Recording of the Band Solafari
        </description>
    </header>
    <scene_setup>
        <volume>6</volume>
        <source name="Vocals" model="point" volume="5">
            <file channel="1">../audio/augenblick_vocal.wav</file>
            <position x="2.40" y="3.26"/>
        </source>
        <source name="RythmGuitar" model="point" volume="4">
            <file channel="1">../audio/augenblick_rythmgit.wav</file>
            <position x="-3.45" y="0.5"/>
        </source>
```

If the load argument of the `ssr` starting command is the path to the ASD file, the `ssr` loads the audio files of the sources and places the sources in the room.
Fur further information about ASD files, starting the `ssr`, commands in the `ssr` please read the Documentation of the `ssr` ([1]).

6. Using the SSR

The usage of the Sound Scape renderer is described in detail in the documentation of the `ssr` ([1]). Here is a detailed description which commands to use for starting the `ssr` in the mode for Wave Field Synthesis.

6.1. Example

Here is a simple example which commands must be used to start the `ssr` in wfs mode with a simple demo from section 7.1.

1. Ensure that the sound cards are configured in the right way, described in section 4.

2. Start the `jack`-server with:

   ```
   jackd -d alsa -d hdsp_128 -r 44100
   ```

3. Start the qjackctl to observe the connections and the server using

   ```
   qjackctl
   ```

   and open the connection client.

4. Start the Sound Scape Renderer in the right mode with the room model for the acoustic lab and with the ASD-File for a demo using:

```
ssr --wfs --setup
/home/lmsdemo/WFS_LNT217/Roommodels/acoustic_lab_128.asd
/home/lmsdemo/WFS_LNT217/Demos/DEMO_Schaefer/ASD_FILES/augenblick.asd
```

5. Start the playback in the `ssr`-gui.

7. Demos

On the computer in the audio laboratory different demos are located. The demos are stored in the folder `/home/lmsdemo/WFS_LNT217`.
In the folder `lmsdemo` are also some older demos - based on the older ring with 64 loudspeakers - which are not tested or ported to the new system of the audio laboratory.
A detailed description of the folder structure can be found in section A.1.

7.1. Demos by Maximilian Schäfer

The demos created by Maximilian Schäfer are located in the folder:

`/home/lmsdemo/WFS_LNT217/Demos/DEMO_Schaefer`.

The folder contains the ASD-Files of the demos, and some additional scripts. Both Demos are from a recording of the band "Solafari" (`https://www.facebook.com/solafari`).
All rights of the recording belong to the Band, here represented by Maximilian Schäfer.

The first demo is a simpler demo with five different sources:

- Vocals
- Drums
- Bass
- Rhythm Guitar

- Solo Guitar

which are located at different positions in the room. Most sources are represented as a single point source, only the bass guitar is modeled as a plain wave. The solo guitar is the only source which is located in the room, all others are located behind the loudspeakers. To test the demo start the `ssr` in wfs mode with the room model for the acoustic lab - all coordinates of the sources are refer to this room - and with the ASD-File `augenblick.asd`.

The second demo created by Maximilian Schäfer is another recording of the band "Solafari". Its a more complex demo with doubled guitar, doubled voices. Further there is a moving source in this demo, the guitar solo is moving from the left to the right of the room. The moving is realized with a easy fading from one source to the other source. To test the demo start the `ssr` in wfs mode with the room model for the acoustic lab - all coordinates of the sources are refer to this room - and with the ASD-File `therapie.asd`.

7.2. Demos by Josef Rettelbach

The demoes created by Florian Josef Rettelbach are based on his Bachelor Thesis "Erstellung von Hörbeispielen für ein Wellenfeldsynthese-System" ([4]). The Demos are located in `/home/ lmsdemo/WFS_LNT217/Demos/DEMO_Rettelbach`.
The Folder with the ASD-Files contains a `readme.txt` where each demo is described in a short way. Additionally there are some demos with moving sources, the way to use these demos is also described in a file in the folder `PATH.../skripte`.

7.3. Demos from DaFX14

In the context of the `DaFX14` at the University of Erlangen, the audio laboratory get some Demos from Matthias Geier from the University of Rostock. These are different

Demos from the University of Rostock and some other public demo material from the `FH Koeln` and the `HMT Rostock`.

7.4. Other Demos

In the audio laboratory are also some older demos, which was build for the older setup with a 64-Channel setup. The Demos are located in the folder `/home/lmsdemo/` and the sub-folders. Some of the demos are working fine with the new setup, others are not working – including the wfs app –.

A. Appendix

A.1. Folder Structure

```
/home/lmsdemo/WFS_LNT217
├── Demos
│   ├── DEMO_Rettelbach
│   │   ├── ASD_FILES ..... Includes the ASD files of different Demos
│   │   └── skripte ..... Includes additional scripts
│   ├── DEMO_Schaefer
│   │   ├── ASD_FILES ..... Includes the ASD files of different Demos
│   │   └── skripte ..... Includes additional scripts
│   └── DEMO_DAFX
│       ├── fh-koeln-master ..... Demos from the FH Koeln
│       ├── hmt-rostock-master ..... Demos from the HMT Rostock
│       └── non_public_scenes ..... Demos from the University of Rostock
├── Literatur
│   ├── Abschlussarbeiten ..... Includes different thesis dealing with wfs
│   └── SoundScapeRenderer-0.3.4-manual.pdf ..... Manual of the ssr
├── Roommodels ..... Includes different roommodels
│   └── matlab ... contains a script to create a room model from measured positions
│       out of a matlab struct
└── tools ..... Includes different scripts and further files
```

A.2. Known Issues

There are some known problems which occur sometimes. Here is a list of the problems with temporal solutions.

1. Sometimes the main gui of the `ssr` is not shown.
   If this error occurs you can try to make a update of the operating system. For problems with the operation system or installing stuff, please contact the system administrator Bernd Westrich. Contact can be found in the list below.

2. `SSR` won't start with the error:

```
...
.../default_wfs_prefilter.wav" couldn't be loaded!
...
```

   This problem occurs when some files the `ssr` uses are stored in the wrong place. As a solution you can use a config file to set the paths manual while starting the `ssr`. The command would look like this:

```
ssr --wfs --config /home/lmsdemo/WFS_LNT217/tools/ssr.conf ...
```

   Alternatively you can create a folder `.ssr` in which lies a file `ssr.conf` in which different features are described. Then you don't need the `--config`, then the `ssr` will automatically read out the config file.

For further problems you can contact people from the following list:

- **Prof. Dr.-Ing. habil. Rudolf Rabenstein - rabe@lnt.de**: For problems with wfs and the demos.
- **Maximilian Schäfer - max@schaefer-gun.de**: For any problems with the ssr or the demos.
- **Bernd Westrich - westrich@lnt.de**: For any problems with the operating system or installation of programs

A.3. Short Instructions

1. Ensure that the sound cards are configured in the right way, described in section 4.

2. Start the `jack`-server with:

   ```
   jackd -d alsa -d hdsp_128 -r 44100
   ```

3. Start the qjackctl to observe the connections and the server using

   ```
   qjackctl
   ```

   and open the connection client.

4. Start the Sound Scape Renderer in the right mode with the room model for the acoustic lab and with the ASD-File for a demo using:

   ```
   ssr --wfs --setup
   /home/lmsdemo/WFS_LNT217/Roommodels/acoustic_lab_128.asd
   /home/lmsdemo/WFS_LNT217/Demos/DEMO_Schaefer/ASD_FILES/augenblick.asd
   ```

5. Start the playback in the `ssr`-gui.

Bibliography

[1] Jens Ahrens, Matthias Geier, and Sascha Spors. Introduction to the soundscape renderer (ssr). `http://spatialaudio.net/ssr/`, 2012.

[2] jackaudio. QjackCtl JACK Audio Connection Kit - Qt GUI Interface. `http://qjackctl.sourceforge.net`, 1999. [Online; accessed 19-January-2014].

[3] jackaudio. JACK Audio Connection kit. `http://jackaudio.org`, 2013. [Online; accessed 19-January-2014].

[4] Florian Josef Rettelbach. Erstellung von hörbeispielen für ein wellenfeldsynthesesystem. bachelor thesis, Friedrich Alexander Universität Erlangen-Nürnberg, 2011.

[5] Sascha Spors. Sound Scape Renderer. `http://spatialaudio.net/ssr/`, 1999. [Online; accessed 19-January-2014].

[6] Ecasound Kai Vehmanen. Ecasound. `http://ecasound.seul.org/ecasound/index.php`, 1999. [Online; accessed 19-January-2014].