Spatial Audio with the *SoundScape Renderer*

Matthias Geier, Sascha Spors
Institut für Nachrichtentechnik, Universität Rostock
{Matthias.Geier,Sascha.Spors}@uni-rostock.de

Abstract

The SoundScape Renderer (SSR) is a versatile tool for realtime spatial audio reproduction, implementing a variety of headphone- and loudspeaker-based methods. Among others this includes Wave Field Synthesis, Higher Order Ambisonics and dynamic binaural synthesis. The SSR is free software licensed under the GNU General Public License. It uses the JACK audio framework and is currently available for Linux and Mac OS X. Interaction with the program is possible using the built-in graphical user interface and via a network interface. For headphone reproduction native support for a number of head trackers is included.

Sound sources including their position and other information are represented in a so-called spatial audio scene. Scenes can be loaded from and saved to XML files. The scene description is independent from the rendering method. For instance, a spatial audio scene can be authored in a studio with headphones, using the binaural renderer. Later on, the scene can be performed in a different venue using loudspeakers, using Wave Field Synthesis.

Recently, the signal processing core was rewritten for multi-threaded processing and independence of the audio backend. Since then, all rendering modules of the SSR are also available as a software library which can be used to create stand-alone programs as well as plugins for existing audio software. Currently, all rendering modules are available as Externals for Puredata and as MEX-files for Octave/Matlab.

1. Introduction

In the pursuit of more spatial accuracy and overall fidelity a number of different audio reproduction systems have been developed and installed in different venues for quite different target applications. Most current high-resolution systems need a large number of loudspeakers, but also vast amounts of associated hardware like amplifiers, digital-to-analog converters and cabling. And they use computers running software which is in many cases custom-made. It is quite time-consuming to implement software for audio reproduction and much more so to maintain it. Often the software is specifically engineered for the loudspeaker setup at hand and mostly it is limited to one reproduction algorithm.

To avoid re-inventing and – more importantly – re-implementing the wheel, the *SoundScape Renderer* (SSR) was created. The SSR is a tool for spatial audio reproduction providing several rendering algorithms for loudspeaker systems and headphones. The necessary output signals for each rendering method are generated in realtime from a scene description (see section 3). Therefore, interaction is possible during playback. Currently, reproduction is limited to sound sources in the horizontal plane, but an extension to 3D is planned.

The SSR was developed at Quality and Usability Lab, Technische Universität Berlin and it is now maintained and continuously improved at Institut für Nachrichtentechnik, Universität Rostock. It is implemented in C++ and it can make use of multi-processor and multi-core computers

by means of multi-threading. It is available as native application (see section 5) and as a library for use in other software (see section 6). The SoundScape Renderer is free software, released under the GNU General Public License (GPL). It can be downloaded – including documentation, example scenes and more – from `http://SoundScapeRenderer.github.com`.
The SSR is an example for the concept of object-based audio reproduction, which is described in the following section.

2. Channel-based vs. object-based; data-based vs. model-based

Since the invention of stereophony, the assignment of a given number of audio channels to a set of loudspeakers is (in most cases) fixed, as well as the positions of the loudspeakers themselves. The loudspeaker setup is implicitly coded in the loudspeaker driving signals. This concept – known as *channel-based* reproduction – requires a separate mix for each new reproduction setup and the intended spatial impression can only be achieved if the loudspeaker positions are reasonably well matched to the positions used for the original creation of the mix.

In the last few decades, several massive multichannel reproduction systems were developed. For such large systems – some of them having several hundreds of loudspeaker channels – the channel-based approach is not feasible anymore. Not only is the sheer number of loudspeaker channels impractical to store and transmit, but also are there no standardized layouts and each system potentially has a different number of loudspeakers and different loudspeaker positions which would require an individual mix for each system. To avoid these limitations, a so-called *object-based* paradigm can be employed. This means that instead of storing output signals, the source signals are stored as audio objects, together with their intended position and other parameters. All audio objects together form a spatial audio scene which can be stored as described in section 3. To reproduce such a scene, a rendering software like the SoundScape Renderer can be used to compute the necessary output signals in realtime. This method not only avoids creating and storing a mix for each reproduction setup but also makes interaction with the scene during playback possible. This concept is not limited to loudspeaker systems, a scene can also be rendered for binaural reproduction with headphones. A channel-based mix can easily be created from an object-based representation by rendering the scene to a given setup and storing the signals of the output channels.

Changing a production workflow from channel-based to object-based results in moving some work from the mixing step in the production process to the realtime rendering process. Instead of generating the final output signals by, for example, manually panning individual tracks between pairs of output channels, the sound engineer uses them as source objects and stores their virtual position and other parameters in a scene description. How exactly the actual output signals will be generated from this information is not known during the mixing process, as it depends on the reproduction system which will be used. Therefore, the task of the sound engineer changes from creating the best possible output signals to creating the best possible scene description.

Apart from the differentiation between channel-based and object-based, a spatial reproduction method can also be characterized with regard to data-based and model-based reproduction. In *data-based* reproduction a sound event is recorded with a set of microphones and there are certain rules how to process the recorded microphone signals to obtain the driving signals of a given reproduction setup. The number of sound sources, their individual properties as well as the properties of the recording room are all part of the recorded signals and cannot be individually manipulated. Recording with traditional main microphone setups can be regarded as data-based technique.

In *model-based* reproduction, on the other hand, dry recordings of individual sound sources are used as input signals to a spatio-temporal model of a virtual sound source. This model can be, for example, a point source or a plane wave or a more complex source having a certain size, shape and directivity. Creation of phantom sources by panning a spot microphone between stereo loudspeakers can be seen as a special case of model-based rendering. Data-based and model-based reproduction are not mutually exclusive, both can be used in the same reproduction system.

It is important to note that an object-based representation is not limited to model-based rendering. It can also contain data-based objects such as Ambisonics B-format recordings. An object-based scene can even contain channel-based recordings as objects. Stereophonic signals can be incorporated into a scene by adding source objects which are reproducing the signals as virtual loudspeakers. This technique is referred to as *Virtual Panning Spots* [1].

3. Spatial Audio Scenes

Along with the paradigm shift from channel-based to object-based audio reproduction there comes the necessity to store not only audio data but also additional control data. Instead of storing a collection of audio channels representing the loudspeaker signals, a so-called *audio scene* is created. Such an audio scene consists of sounding objects which are associated with source signals, a certain position or trajectory in the virtual space and other source parameters. In addition to source objects there can be objects with acoustical properties like reflection, absorption and diffraction. The virtual room can also be described with higher-level acoustical parameters.

The SSR uses an interchange format for audio scenes called *Audio Scene Description Format* (ASDF) [2]. The ASDF is independent of the rendering algorithm and therefore a given audio scene can be reproduced on any reproduction setup. The ASDF is human-readable, i.e. scene files can be opened in a plain text editor and can be inspected and edited without specialized editor software. Therefore, the ASDF can also be used for scene authoring. The ASDF in its current form – which is still work-in-progress – is based on the eXtensible Markup Language (XML). Source signals are stored in traditional audio file formats and linked to the scene description file.

It is obvious that not all reproduction methods will be able to achieve perceptually identical reproduction of a given audio scene. Depending on the target system (and room acoustics of the target venue), it might be necessary to make manual adjustments.

Currently, the ASDF can only describe static scenes, dynamic features like trajectories are planned.

4. Rendering Algorithms

The SSR was created with the goal to be able to use many different rendering algorithms within the same framework. All currently available rendering methods are described in the following sections. The flexible architecture of the SSR allows the simple and efficient implementation of new rendering algorithms based on common building blocks which can be re-used between renderers.

4.1. Dynamic Binaural Synthesis

Binaural reproduction uses *Head-Related Impulse Responses* (HRIRs) to recreate the acoustic signals at the ears of a listener which are typically played back via headphones. The ability of the human auditory system to localize sound is based on exploiting the acoustical properties of the human body, especially the head and pinnae [3]. These acoustical properties can be captured by HRIRs by measuring the transfer function between a sound source and the listener's ears. These measurements have to be undertaken for all desired angles of incidence.

HRIR-based reproduction is implemented in the SSR by filtering a source signal with the appropriate HRIR for the left and right ear. In order to cope with varying head orientations, head tracking can be used. Arbitrary HRIR sets can be used in the SSR. A multi-channel WAV file with two channels for each angle of incidence has to be provided. Thus, if an angular resolution of one degree is desired, a 720-channel file is used. HRIRs measured with the FABIAN system [4] are included in the SSR package. Further HRIR sets as described in [5] can be downloaded[1]. When using HRIRs, it is important to apply appropriate headphone compensation [6].

4.2. Wave Field Synthesis (WFS)

Wave Field Synthesis aims at physically recreating a desired sound field within a given listening area. WFS is based on the quantitative formulation of the *Huygens-Fresnel-Principle*, which states that a propagating wave front can be synthesized by a superposition of simple sources placed on the wave front. WFS has initially been developed for linear secondary source distributions [7] and has been extended to arbitrary convex secondary source layouts [8] which may even only partly enclose the listening area. The theoretically continuous secondary source distribution is approximated by spatially discrete loudspeakers in practice. The resulting spatial sampling artifacts for typical geometries differ considerably from NFC-HOA (see next section) [9]. WFS exhibits no pronounced sweet spot and the sampling artifacts are rather evenly distributed over a large listening area. The sampling artifacts may be perceived as coloration of the sound field [10]. The localization of sound sources is very accurate for typical setups [11].

The SSR can compute the loudspeaker driving signals very efficiently by filtering the virtual source signals with a pre-filter and weighting and delaying them. This way, virtual point sources and plane waves can be generated. Focused sources (virtual sources inside the listening area) are also possible. Although the pre-filter should depend on source and listener positions, it can be kept static for practical reasons. In the current implementation of the SSR it only depends on the given loudspeaker setup and an individual pre-filter has to be generated for each loudspeaker setup. An automatically generated dynamic pre-filter is planned [12].

4.3. Near-Field-Compensated (Higher Order) Ambisonics (NFC-HOA)

Higher Order Ambisonics (HOA) [13, 14] is based on the concept of orthogonal sound field representations. The underlying theory assumes that a continuous single layer potential (secondary source distribution) surrounds the listening area. Appropriate weights (driving functions) applied to the secondary source distribution allow to reproduce almost any desired sound field within the listening area. Although arbitrary secondary source contours are theoretically possible, explicit solutions are currently only available for spherical and circular geometries. In reality, the continuous distribution of secondary source is approximated by discrete loudspeakers. This constitutes a spatial sampling process which may lead to spatial aliasing. This affects the whole listening area except a pronounced artifact-free area in the center of the loudspeaker

[1] `https://dev.qu.tu-berlin.de/projects/measurements/wiki/`

arrangement [9]. The size of this area decreases with increasing frequency of the signal to be reproduced. For feasible loudspeaker setups the size of the artifact-free reproduction area is typically smaller than a human head at the upper end of the audible frequency range. Outside, spatial sampling artifacts arise that may be perceived as coloration of the desired sound field [15].

In the SSR, virtual sources can be modeled as point sources and plane waves. Depending on the number of available loudspeakers, the Ambisonics order is automatically chosen appropriately. For example, a ring of 56 loudspeakers is driven with order 28. The NFC-HOA renderer is currently the most computationally demanding algorithm available in the SSR. To achieve realtime performance, an efficient implementation based on IIR filters is used [16]. Only circular setups can be used in the SSR, support for spherical setups is future work.

4.4. (Higher Order) Ambisonics Amplitude Panning

Ambisonics amplitude panning [17] is a much simpler variety of Ambisonics. As the name suggests, only amplitude panning is used. Each source is reproduced by a weighted combination of potentially all loudspeakers. There are two options available: *in-phase* and *out-of-phase*. In the latter, negative weighting factors can occur (i.e. a phase-inverted signal is reproduced by the loudspeaker) – with the potential of audible artifacts.

The main difference to the method described in the previous section is that it lacks near-field compensation. This means that the loudspeakers are assumed to be far away and that they emit plane wave fronts. In reality this is not the case, which leads to a small sweet spot and localization errors outside of this sweet spot. As in NFC-HOA, the reproduction order is automatically matched to the number of available loudspeakers

4.5. Vector Base Amplitude Panning (VBAP)

VBAP [18] is a simple panning method where pairs of adjacent loudspeakers create the illusion of a phantom source. For 3D-reproduction, the phantom source is created in between a loudspeaker triple. The SSR, however, can only use 2D-setups in the horizontal plane. 3D-setups are planned as a future extension. VBAP has a very small sweet spot, out of which localization of sources is distorted towards the nearest active loudspeaker. VBAP works best for circular setups.

4.6. (Dynamic) Binaural Room Synthesis (BRS)

The BRS renderer [19] is "special" insofar as it does not render sources at their given positions. Instead, it uses Binaural Room Impulse Responses (BRIRs) for rendering. Source positions, room acoustics and other characteristics are encoded in the impulse responses. An extension to the ASDF is used to store BRIR sets for each source as multi-channel WAV files. Two channels are needed for each head orientation, so if a resolution of one degree is desired, a 720-channel WAV file has to be provided for each source.

Head tracking is possible, which generally improves localization and externalization of sound sources [20]. While the source position is ignored, the volume of sources can still be changed. With BRIRs measured in real rooms, the acoustic properties can be reproduced very authentically. For example, the BRIRs measured in a recording studio can be used to "virtually" make a mix in this studio while sitting in an outside broadcast van wearing headphones. For research applications, BRIRs can also be artificially created. They can be used to listen to and evaluate "virtual" loudspeaker systems which do not even exist.

4.7. Binaural Playback Renderer

The binaural playback renderer is probably the most "special" renderer in the SSR. It is used to play pre-recorded binaural recordings with head tracking. That means it uses not only two-channel files for binaural playback but two channels for each head orientiation. If a one-degree resolution is desired, a 720-channel sound file has to be used.

This can be used to test algorithms which are hard to realize with a block-based approach like, for example, rapidly moving sources. It can also be used to listen to algorithms which do not have a realtime implementation yet. The binaural signals have to be generated off-line for each head orientation and can then be played back in realtime including head tracking.

Again, an extension to the ASDF is used to store multi-channel sound files for each source. Several sources (i.e. several different recordings) can be played back at the same time. As in the other "special" renderers, the source position is ignored but the volume of the source – and therefore of the associated recording – can still be changed.

4.8. Generic Renderer

The generic renderer is another "special" renderer which ignores the source positions. Instead, all characteristics are contained in impulse responses. Arbitrary impulse responses can be specified for each source-output-pair independently. Therefore an extension to the ASDF is used where for each source a multi-channel WAV file is used to store the impulse responses. This file must have as many channels as there are loudspeakers. A loudspeaker setup must be specified although the loudspeaker positions are also ignored for rendering. Although most scene information is ignored, sources can still be muted and their volume can be changed.

The generic renderer is a great tool for prototyping novel reproduction algorithms. Loudspeaker driving functions can be calculated in the form of impulse responses and can then be used for realtime reproduction with the SSR. However, the generic renderer is not limited to loudspeaker-based reproduction. It can also be used for – potentially multiple – headphones or any mixture of transducers. Basically, it is a completely generic multiple-input/multiple-output system with a static filter between each source and output.

5. SSR as Native Application

The SoundScape Renderer was initially developed for GNU/Linux starting in the year 2006. It was first released as open-source software in May 2010. Since May 2011 it is also available for Mac OS X. The SSR uses the JACK Audio Connection Kit[2] for input from and output to the channels of a soundcard. Additionally, any other JACK-aware application can be used for input and output. This way, the SSR can be connected, for example, to the open-source digital audio workstation Ardour[3].

The SSR has a built-in graphical user interface (GUI). It shows the current position of all sound sources and can be used to move them around, set their volume, their source model and other parameters. Depending on the current rendering algorithm, the GUI also shows either a head or the loudspeaker setup (see figure 1). Both can be rotated and moved around. If not needed, the GUI can also be disabled. The same GUI is used for all rendering algorithms.

As alternative to the GUI, all aspects of the SSR can also be controlled via a network interface. It uses TCP/IP sockets to send and receive XML-like messages. This way the SSR can be

[2] `http://jackaudio.org`

[3] `http://ardour.org`

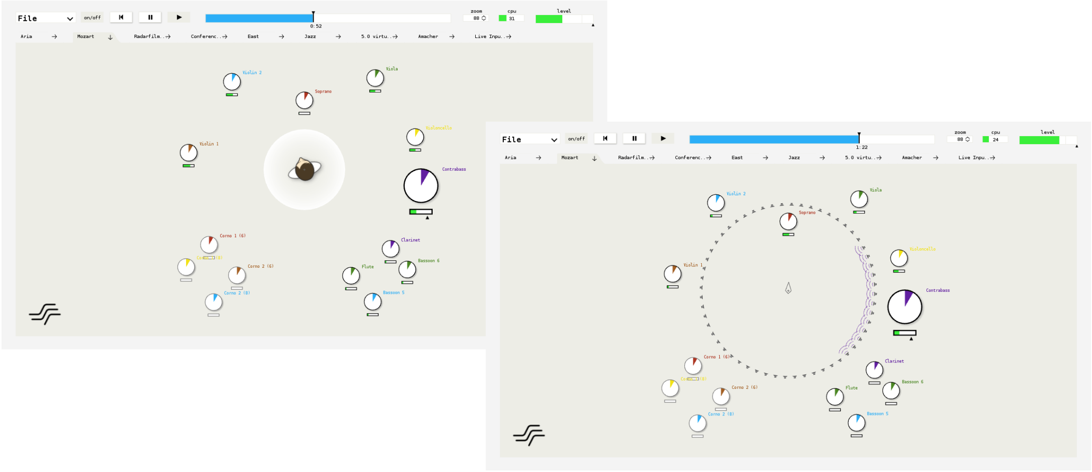

Figure 1: Screenshots of the built-in GUI. *Left:* binaural renderer, *right:* WFS

controlled from different computers on a network, potentially using different operating systems. Several network clients can be connected to the SSR at the same time. One example for a full-featured client application is the Android application described in the next section. The network interface has also been used to control the SSR from Pure Data, Python and Matlab.

As previously mentioned, head tracking is an important feature for binaural sound reproduction. The SSR comes with built-in support for the trackers Polhemus Fastrak, InterSense InertiaCube3 and for the (much cheaper) do-it-yourself head tracker Razor AHRS[4].

5.1. SSR Remote for Android

A showpiece example for a network client is the SSR remote control for Android smartphones. It features bi-directional communication with the SSR and works great on a local wireless network.

The full source code as well as an APK file for immediate installation on an Android-based device is available at `http://github.com/SoundScapeRenderer/android-remote`. There is also a Video for whetting your appetite: `http://vimeo.com/17159650`.

6. SSR as Plugin

The signal processing core of the SSR has recently been rewritten [21], mainly to allow parallel processing by using multi-threading. As a side effect, the audio backend is now exchangeable and therefore not limited to JACK anymore.

The core of the SSR is now available as C++ library which can be used to implement plugins for different host programs. This way, the SSR could be integrated within your favorite graphical patching application and digital audio workstation. Currently, all renderers are available as an External for PureData[5] (using the *flext* library – so it may also work as a Max/MSP External) and MEX-files for Octave/Matlab. Implementations for other plugin APIs – realtime or non-realtime – should be fairly straightforward.

[4] `http://dev.qu.tu-berlin.de/projects/sf-razor-9dof-ahrs/wiki/`

[5] `http://puredata.info`

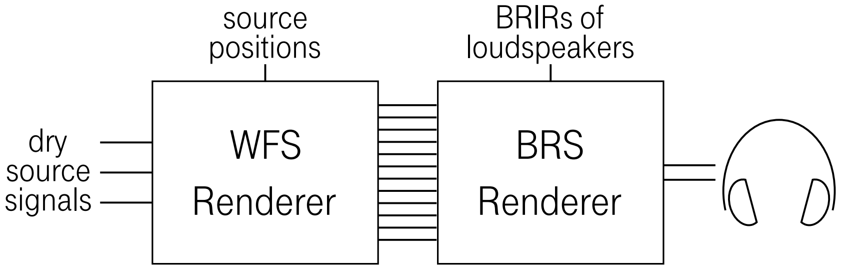

Figure 2: Example for a *virtual* WFS system realized with the BRS renderer

7. Applications

The main application area of the SSR is of course the reproduction of spatial audio scenes. These can be music pieces, audio plays or movie soundtracks. They can also be interactive or non-interactive sound installations.

When audio scenes are targeted for large loudspeaker setups, the SSR can also be used for system-independent mixing and binaural monitoring [22]. Figure 2 shows a possible setup where two instances of the SSR are used. One instance is running the WFS renderer to create the driving signals for the "virtual" loudspeakers, the second instance is running the BRS renderer with measured BRIRs of the actual loudspeakers and rendering the result – with optional head tracking – to headphones.

With its background in research, the SSR can of course be used for comparison and evaluation of different reproduction algorithms and it can be used as audio backend in listening or interaction tests for basic research, as for example in [23]. Thanks to its flexible network interface, the SSR can easily be connected to other systems, for example for visual or multimodal interaction [24].

8. Future Work

One limitation of the SSR is that source positions are limited to the horizontal plane. In the near future, the extension to three-dimensional positions and orientations is planned. It should be straightforward to make the VBAP renderer and the binaural renderer 3D-aware, for other renderers it might not be that easy. For the handling of three-dimensional HRIRs it will be practical to use a specialized file format instead of multi-channel WAV files.

Another limitation is that only static audio scenes can be stored. The Audio Scene Description Format will have to be extended to allow source movements along trajectories and other dynamic features. Further possible features for the SSR are simple room models, separate headphone and loudspeaker compensation filters and further network protocols (e.g. HTML5 WebSockets). It could also be interesting to add some features for data-based rendering like the inclusion of Ambisonics B-Format sources.

There are always things left to do, and contributions from the community are very welcome!

9. References

[1] G. Theile, H. Wittek and M. Reisinger. Potential Wavefield Synthesis applications in the multichannel stereophonic world. In *24th AES Conference*. 2003.

[2] M. Geier and S. Spors. ASDF: Audio Scene Description Format. In *International Computer Music Conference*. 2008.

[3] J. Blauert. *Spatial Hearing: The Psychophysics of Human Sound Localization*. MIT Press, 1996.

[4] A. Lindau and S. Weinzierl. FABIAN – An instrument for the software-based measurement of binaural room impulse responses in multiple degrees of freedom. In *24. Tonmeistertagung (VDT International Convention)*. 2006.

[5] H. Wierstorf et al. A free database of Head-Related Impulse Response measurements in the horizontal plane with multiple distances. In *130th AES Convention*. May 2011.

[6] Z. Schärer and A. Lindau. Evaluation of equalization methods for binaural signals. In *126th AES Convention*. May 2009.

[7] A. J. Berkhout. A holographic approach to acoustic control. *Journal of the AES*, 36(12):977–995, 1988.

[8] S. Spors, R. Rabenstein and J. Ahrens. The theory of Wave Field Synthesis revisited. In *124th AES Convention*. May 2008.

[9] S. Spors and J. Ahrens. A comparison of Wave Field Synthesis and Higher-Order Ambisonics with respect to physical properties and spatial sampling. In *125th AES Convention*. October 2008.

[10] H. Wittek. *Perceptual differences between Wavefield Synthesis and Stereophony*. Ph.D. thesis, University of Surrey, 2007.

[11] H. Wierstorf, A. Raake and S. Spors. Localization of a virtual point source within the listening area for Wave Field Synthesis. In *133rd AES Convention*. October 2012.

[12] F. Schultz et al. Derivation of IIR-pre-filters for soundfield synthesis using linear secondary source distributions. In *AIA-DAGA Conference on Acoustics*. 2013.

[13] J. Daniel. *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*. Ph.D. thesis, Université Paris 6, 2000.

[14] J. Ahrens and S. Spors. An analytical approach to sound field reproduction using circular and spherical loudspeaker distributions. *Acta Acoustica united with Acoustica*, 94(6):988–999, December 2008.

[15] J. Ahrens and S. Spors. Alterations of the temporal spectrum in high-resolution sound field reproduction of different spatial bandwidths. In *126th AES Convention*. May 2009.

[16] S. Spors, V. Kuscher and J. Ahrens. Efficient realization of model-based rendering for 2.5-dimensional near-field compensated Higher Order Ambisonics. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2011.

[17] M. Neukom and J. C. Schacher. Ambisonics equivalent panning. In *International Computer Music Conference*. 2008.

[18] V. Pulkki. Virtual sound source positioning using Vector Base Amplitude Panning. *Journal of the AES*, 45(6):456–466, 1997.

[19] U. Horbach et al. Design and applications of a data-based auralization system for surround sound. In *106th AES Convention*. 1999.

[20] P. Minnaar et al. The importance of head movements for Binaural Room Synthesis. In *International Conference on Auditory Display*. 2001.

[21] M. Geier, T. Hohn and S. Spors. An open-source C++ framework for multithreaded real-time multichannel audio applications. In *Linux Audio Conference*. 2012.

[22] M. Geier, J. Ahrens and S. Spors. Binaural monitoring of massive multichannel sound reproduction systems using model-based rendering. In *NAG/DAGA International Conference on Acoustics*. 2009.

[23] T. Weißgerber and U. Baumann. Multi-channel sound reproduction for precise measurements in audiology. In *27. Tonmeistertagung (VDT International Convention)*. 2012.

[24] K. Bredies et al. The Multi-Touch SoundScape Renderer. In *9th International Working Conference on Advanced Visual Interfaces (AVI)*. 2008.