Audio Engineering Society
Convention Paper 7100

Presented at the 122nd Convention
2007 May 5–8 Vienna, Austria

Optimized Binaural Modeling for Immersive Audio Applications

Christos Tsakostas[1] and Andreas Floros[2]

[1] HOLISTIKS Engineering Systems, Digeni Akrita 29, 122 43 Athens, Greece
tsakostas@holistiks.com

[2] Dept. of Audiovisual Arts, Ionian University, 49 100 Corfu, Greece
floros@ionio.gr

ABSTRACT

Recent developments related to immersive audio systems are mainly originating from binaural audio processing technology. In this work, a novel high-quality binaural modeling engine is presented suitable for supporting a wide range of applications in the area of virtual reality, mobile playback and computer games. Based on a set of optimized algorithms for Head-Related Transfer Functions (HRTF) equalization, acoustic environment modeling and cross-talk cancellation, it is shown that the proposed binaural engine can achieve significantly improved authenticity for 3D audio representation in real-time. A complete binaural synthesis application is also presented that demonstrates the efficiency of the proposed algorithms.

1. INTRODUCTION

The continuous evolution of multimedia and consumer electronics technologies combined with the enhancements on computational characteristics and power of both hardware and software allows the implementation of advanced sound reproduction applications. New delivery formats (such as DVD-Audio and SACD) and many typical perceptual multichannel audio coding currently exist [1] – [3], that enhance the playback performance of stereo installations at the expense of increased number of loudspeakers with specific placement requirements and sometimes complicated cabling interconnections. Binaural technology represents an attractive alternative of the legacy stereo (or multichannel) audio reproduction, allowing the subjectively accurate 3-Directional (3D) representation of sound through a couple of loudspeakers or headphones [4].

3D audio refers to the perceptually accurate positioning of a number of sound sources around a listener and represents a significant feature for implementing realistic virtual world environments, game engines and specific purpose simulators. Binaural technology achieves 3D audio environment recreation by synthesizing a two-channel audio signal using the well-

known Head Related Transfer Functions (HRTFs) between the sound source and each listener's human ear. Hence, only two loudspeakers or headphones are required for binaural audio playback. The simple setup of a binaural reproduction system renders it convenient for a number of state-of-the-art applications, including mobile applications and communications, especially when headphones are employed, which are relatively easy to use and introduce low power consumption and frequency response [5].

In this work, a real-time, high-quality binaural 3D audio modeling engine is presented suitable for supporting a wide range of applications, such as virtual reality, mobile playback and game platforms. The proposed engine provides a set of significant features (such as optimized HRTF equalization, efficient room acoustics modeling and cross-talk cancellation for transaural reproduction) that result into high 3D audio playback quality. An Application Programming Interface (API) is also provided that enables 3D audio designers to develop a large set of high-quality applications. For the purposes of this work, an amphiotik synthesis application was developed and is presented here for demonstrating the above features.

The paper is organized as following: Section 2 presents a general overview of binaural technology while in Section 3, the proposed Amphiotik 3D Audio Engine is introduced and the optimized HRTF equalization algorithm is described. In Section 4, the performance of the proposed binaural engine is assessed using a 3D audio application developed for the purposes of this work. Finally, Section 5 concludes this work.

2. BINAURAL TECHNOLOGY BACKGROUND

Directional hearing has been already intensively investigated [6]. Generally, it can be performed using wavefield synthesis [7] through a large number of loudspeakers that reproduce the desired sound field. Alternatively, for reduced number of playback sound sources, binaural processing can be employed. In this work we focus on binaural hearing which is based on two basic cues that are responsible for human sound localization perception: a) the interaural time difference (ITD) imposed by the different propagation times of the sound wave to the two (left and right) human ears and b) the interaural level difference (ILD) introduced by the shadowing effect of the head. Both sound localization cues result into the reception of two different sound waves by the human ears that perceptually provide information on the direction of an active sound source [8].

In binaural modeling, the effect of the above basic cues is incorporated into directional-dependent transfer function termed Head Related Transfer Functions (HRTFs). In practice, the HRTFs represent the transfer function between the listener's ear canal and the specific place of the sound source [9]. Hence, convolving the mono sound source wave with the appropriate pair of HRTFs derives the sound waves that correspond to each of the listener's ears. This process is called binaural synthesis. Binaural synthesis can be also combined with existing sound field models producing binaural room simulations and modeling. This method facilitates listening into spaces that only exist in the form of computer models. In more detail, the sound field models can output the exact spatial-temporal characteristics of the reflections in a space. In this case, the summation of binaural synthesis applied to each reflection produces the Binaural Room Impulse Response. As mentioned in Section 1, the binaural left and right signals can be reproduced directly using headphones or a pair of conventional loudspeakers. In the latter case, the additional undesired crosstalk paths that transit the head from each speaker to the opposite ear must be cancelled using crosstalk cancellation techniques [10].

The HRTFs strongly depend on four variables: distance, azimuth, elevation and frequency. For distances longer than about one meter, the sound source is in the so-called "far field", where the HRTFs fall off inversely with the distance range. In order to obtain accurate ITD and ILD models, the HRTFs must be measured for the targeted human head and for different azimuth and elevation angles. This represents one major drawback of binaural hearing, since the employment of a specific HRTF set (such as the Kemar Dummy-head [11] measurements or the CIPIC HRTF database [12]) reduces the spatial localization subjective accuracy [13]. An alternative approach is the employment of parametric HRTFs models, which are based on the HRTF data for estimating frequency-dependent scaling factors. In this respect, they do not impose significant advantages over HRTFs measurements, while they introduce low accuracy [14].

The recorded HRTFs measurements contain the frequency response of the measurement system components (i.e. the microphone and the loudspeaker). To compensate for the response of the above system,

HRTF measurements must be equalized. A number of HRTFs equalization techniques have been published in the literature [6], [15], providing a reference response which is then inverted and used to filter the entire data set.

It must be noted that binaural technology is also employed for realizing binaural auditory models [16]. Such models consider advanced human hearing mechanisms, such as separation of concurrent sound sources and evaluation of properties of hearing events. However, these research topics are out of the scope of the present work and will not be further considered here.

3. AMPHIOTIK API AND 3D AUDIO ENGINE OVERVIEW

The proposed Amphiotik 3D Audio Engine is a software library, which offers the capability of rapid 3D-Audio applications development, yet preserving a carefully designed balance between authenticity and real-time operations. It incorporates state-of-the-art binaural processing algorithms, such as a novel algorithm for HRTFs equalization, crosstalk cancellation techniques and room acoustics modeling for accurate acoustic representation of virtual environments. The Amphiotik 3D Audio Engine state-machine is also responsible for the signal routing that must be performed in order to perform all calculations required for producing the binaural signal within the defined virtual world. The communication of an application layer with the Amphiotik 3D Audio Engine is performed using the Amphiotik API, which provides easy to use software methods for defining the binaural model and the virtual world parameters in real-time.

Figure 1 illustrates the above Amphiotik Technology architecture. The Amphiotik API provides the necessary functions for the definition of the overall virtual auditory environment, that is: (a) the geometry and materials of the virtual world, (b) the sound field model and (c) the virtual sound sources and receivers characteristics and instantaneous position. In addition, it provides functions that interact directly with the Amphiotik 3D Audio Engine to parameterize various aspects of the engine, such as the HRTF data set to be used, the cross-talk cancellation algorithm activation, the headphones equalization, as well as user-defined parametric frequency equalization. A typical example that demonstrates the usage of the Amphiotik API is presented in the Appendix.

The shape of the virtual world can be arbitrary, but for the shake of real-time processing in moderate power computers, “shoebox” like spaces are better supported. For this case, the API provides simple functions for defining the dimensions of the room (length, width and height) and the materials (absorption coefficients) of each surface. An internal materials database is utilized for the re-use of the above materials parameters.

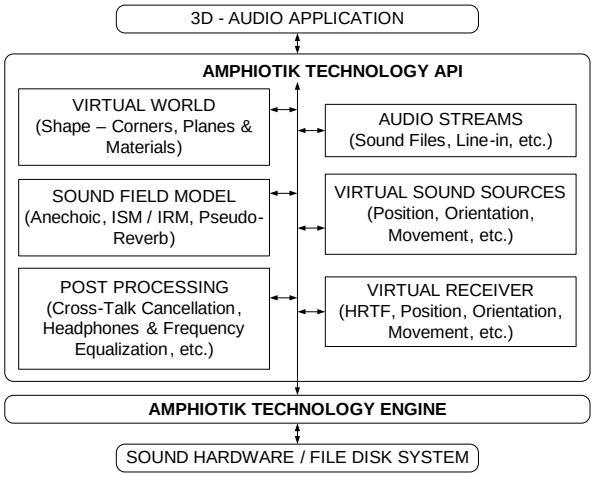

Figure 1 Typical architecture of the proposed Amphiotik Technology

The API also allows the definition of one virtual binaural receiver, while there are no limitations on the number of the defined sound sources. Both the virtual receiver and sound sources can be placed arbitrarily and in real-time in the 3D virtual auditory environment, concerning both their position and orientation. An arbitrary number of audio streams can be defined, which are linked to the virtual sound sources. An audio stream can be associated with more than one virtual sound sources. Audio streams are usually sound files on the local disk system but they can also originate from the soundcard’s line-in input or even an Internet media file link.

Focusing on the Amphiotik 3D Audio Engine, one of the key features for perceptually improved and authentic binaural performance is the novel HRTFs equalization algorithm employed termed as Amphiotik Technology Diffuse Field Equalization (ATDFE). ATDFE is based on (a) a specific, direction-dependent weighting which is applied to the HRTF filters prior to averaging, (b) the smoothing of the average magnitude and (c) the low-frequencies compensation. As depicted in Figure 2, the ATDFE equalization algorithm, results into a much

better approximation of the original HRTF data than the legacy DFE strategy, reducing the maximum deviation from the measured magnitude to only 3dB. It must be noted that the Amphiotik 3D Audio Engine supports installable sets of HRTFs. For this reason, a file format has been designed - the Amphiotik Technology HRTF (ATHRTF) format - that stands as a median between Amphiotik Technology and various, well - known HRTF formats. A software converter that automates this process has been also developed (namely the ATHRTF Converter), supporting some the most widely known HRTF formats. As it will be described in the following Section, a viewer of the HRTF dataset, called "Amphiotik Technology HRTF Tool" is also available for efficient monitoring of the selected HRTF data set.

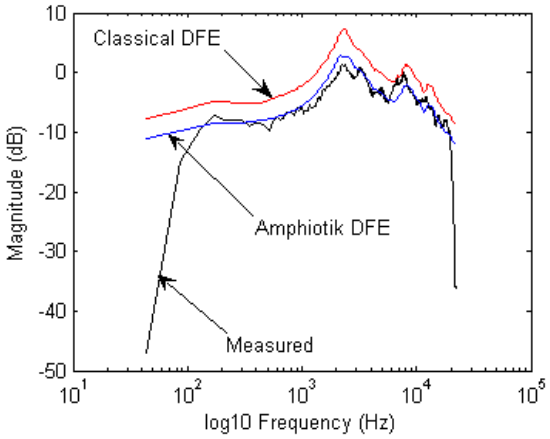

Figure 2 Typical ATDFE equalization performance - ATDFE & Classical DFE estimated magnitude of unwanted effects filter vs. the measured one

Using the Amphiotik 3D Audio Engine, the acoustical environment modeling can be performed using one of the following sound field models: (a) anechoic, (b) early part, and (c) early part & pseudo-reverb. For the anechoic case reflections are not considered, consequently the geometry of the room is ignored. On the other hand, the early part is simulated by means of the "Image Source Method" and "Image Receiver Method" [17]. The order of the reflections can be altered in real-time and its maximum value is limited to five. Early part & pseudo-reverb uses a hybrid algorithm in which the early part is estimated as described earlier and the reverberation part (i.e. the late part) of the room impulse response is estimated with digital audio signal processing algorithms. Specifically, two reverberation algorithms are currently supported: (a) Schroeder [18] and (b) Moorer [19]. According to the Schroeder algorithm the late part is calculated by the means of comb-filters and all-pass filters, whilst for the case of the Moorer technique the late part is approximated with an exponentially decaying white noise. The reverberation time is calculated using the Sabine equation [20]. A proprietary algorithm has been employed in order to combine the binaural early part with the monaural late part.

Figure 3 depicts a general overview of the Amphiotik 3D-Audio Engine. For each pair of virtual receiver and sound source, a binaural IR is calculated, taking under consideration their instantaneous positions, orientation and room geometry and materials as well, if the early part option is enabled. Real-time convolution is accomplished by the means of un-partitioned and partitioned overlap-and-add algorithms. Each convolution produces two channels: Left (L) and Right (R). All the L and R channels, produced for each virtual sound source, are summed up producing finally only two channels.

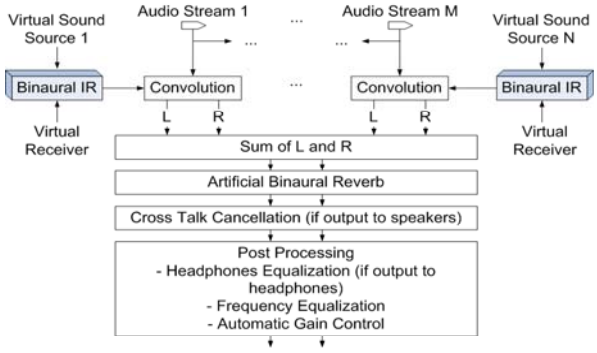

Figure 3 Amphiotik 3D Audio Engine general structure

Crosstalk cancellation (CTC) is applied if the audio playback is performed over stereo loudspeakers. CTC filters are built, in real-time, with HRTFs. The Amphiotik API gives the capability to select a different set of HRTFs for the crosstalk cancellation processing than the one used for spatialization. In general, CTC filters built with HRTFs impose the problem that the low-power low frequencies are excessively amplified and vice-versa. Two strategies have been used in order to overcome this problem: (a) the employment of band-limited CTC and (b) a special equalization method called "Transaural Equalization". Band-limited CTC simply partially overcome the above problem by not using the very low and the very high frequencies. The outcome is that musicality becomes significantly better,

and at the same time the loss of the very low and the very high frequencies is not particularly perceptible. Transaural equalization on the other hand, is based on post-filtering of the transaural audio channels, in order to approximate the magnitude that they would have if listening over headphones was selected. In addition, the software gives the capability to apply the crosstalk cancellation filters directly to the stereo input without prior processing through the 3D-Audio engine. It is also important to note that the Amphiotik API gives the capability to select non-symmetric loudspeakers positions (e.g. for loudspeakers in cars).

Additionally, as it shown in Figure 3, post-equalization is applied to the synthesized binaural signal, which may optionally include headphones equalization, user-defined frequency equalization and Automatic Gain Control (AGC). In addition, pre-equalization is also supported for the stereo audio signals before they are spatialized.

Finally, for effective real-time operation and interaction with the user, the Amphiotik Engine checks for any possible change of the parameters in time frames, which are defined by the block length used (typically 512 - 8192 samples at a sampling rate equal to 44.1 KHz) and re-initializes all the appropriate modules.

4. AMPHIOTIK SYNTHESIS APPLICATION: A CASE STUDY

In order to demonstrate the capabilities and the performance of the Amphiotik 3D Audio Engine, the Amphiotik Synthesis application was developed during this work, which uses the Amphiotik API and 3D audio engine for binaural or stereo mixing and mastering of multichannel high-quality audio content (Figure 4).

Amphiotik Synthesis incorporates all the features described in Section 3, through a graphical user interface (GUI), i.e. the manipulation of audio streams, virtual sound sources, virtual binaural receiver, sound field models, HRTF, crosstalk cancellation, headphones and frequency equalization. It is important to note that the GUI supports 3-dimensional view of the virtual world, for better user-perception of the intended simulations and auralizations. The GUI also supports a very convenient way for mixing audio channels, even if just standard stereo output is desired, by moving the virtual sound sources with the mouse.

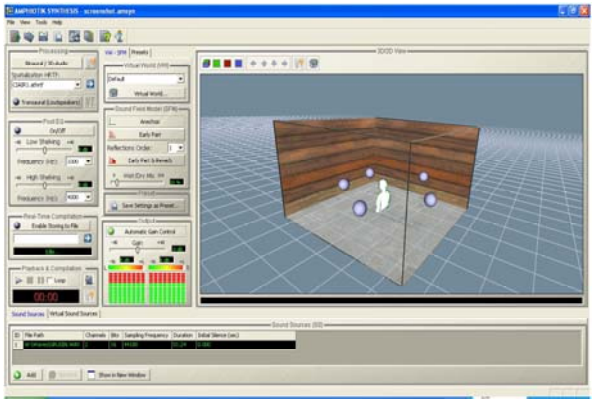

Figure 4 Amphiotik Synthesis application

In addition, Amphiotik Synthesis includes some extra tools like A/B Player (for comparison purposes – especially stereo vs. binaural), Batch Processing and the Amphiotik Technology HRTF Tool shown in Figure 5 or efficiently handling and monitoring the selected HRTF library.

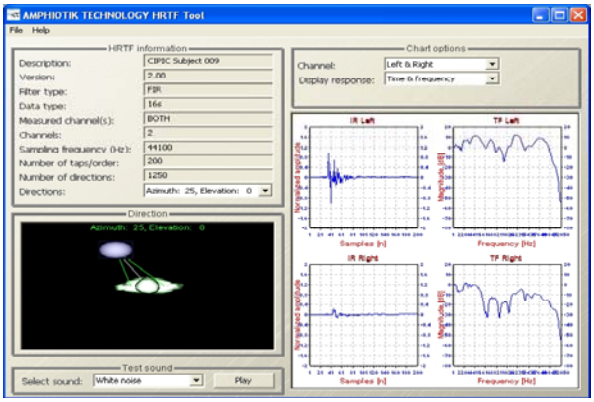

Figure 5 Amphiotik Technology HRTF Tool

Amphiotik Synthesis allows an arbitrary number of virtual sound sources to be inserted into the virtual world. On moderate power computers, it was found that it can process up to 8 sound sources at 96kHz/24bit, whilst even more can be supported for lower sound quality (44.1kHz/16bit). In the cases that the engine cannot operate in real-time (i.e. very high number of virtual sound sources, low CPU performance, etc.) offline processing can be alternatively utilized. Moreover, it was observed that for less virtual sound sources and for up to 5th order ISM/IRM room acoustic analysis and simulation methods, real-time processing can still be achieved. Finally, the processing can be set

with a single click to standard stereo, transaural stereo, binaural for headphones or binaural for loudspeakers.

Amphiotik Synthesis application has extensively been used for the generation of stimuli for psychoacoustical experiments and the mixing (stereo and/or binaural) of multichannel audio tracks. In all test cases it was found that the achieved spatial impression was higher when binaural mixing was performed. The above perceptually assessed performance was additionally improved when playback was performed through headphones and the ATDFE HRTF equalization algorithm was employed.

5. CONCLUSIONS

In this work the Amphiotik 3D Audio Engine is presented which combines the well-known binaural technology and HRTF theory for creating state-of-the-art virtual audio environments. The proposed audio engine incorporates a novel HRTF equalization method that significantly improves the spatial position perception of the active sound sources. Moreover, crosstalk cancellation algorithms are supported for stereo loudspeaker support, with a large number of FIR coefficients (2048 at 44.1kHz sampling frequency), while an Image Receiver Method (IRM) instead of Image Source Method (ISM) for fast and accurate calculation of the environment's impulse responses are employed.

All binaural calculations and processing/mixing are performed in real-time. Due to the efficient Amphiotik Engine implementation, up to 8 sound sources are currently supported at 96kHz/24bit. The above number of sound sources can be significantly increased for CD-quality sound (44.1kHz/16bit). Moreover, up to 5th order ISM/IRM room acoustic analysis and simulation methods are also running in real-time. A user friendly Application Protocol Interface (API) is also provided for easy integration of the Amphiotik 3D Audio Engine with any computer application. Some of the features that future versions of the Amphiotik Audio Engine will include are Equivalent Rectangular Bandwidth (ERB) filter bank HRTF equalization, as well as embedded hardware support.

6. REFERENCES

[1] J. Herre, C. Faller, S. Disch, C. Ertel, J. Hilpert, A. Hoelzer, K. Linzmeier, C. Spenger, P. Kroon, "Spatial Audio Coding: Next-Generation Efficient and Compatible Coding of Multi-Channel Audio", 117th AES Convention, San Francisco 2004, (preprint 6186).

[2] M. Davis, "The AC-3 Multichannel Coder", presented at the AES 95th Convention, AES, New York, October 1993, (preprint 3774).

[3] K. Brandenburg and M. Bosi, "ISO/IEC MPEG-2 Advanced Audio Coding: Overview and Applications"," presented at the AES 103rd Convention, New York, September 1997, (preprint 4641).

[4] W. Gardner, "3D Audio using loudspeakers", Ph.D. thesis, Massachusetts Institute of Technology, September 1997.

[5] AES Staff Technical Writer, "Binaural Technology for Mobile Applications", J. Audio Eng. Soc., Vol. 54, No. 10, Oct. 2006, pp.990 – 995.

[6] J. Blauert, "Spatial Hearing" (revised edition), The MIT Press, Cambridge, Massachusetts, 1997.

[7] A. J. Berkhout, P. Vogel, and D. de Vries, "Use of wave field synthesis for natural reinforced sound", presented at the AES 92nd Convention, 1992, (preprint 3299).

[8] V. Pulkki, "Compensating Displacement of amplitude-panned Virtual sources", presented at the AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio, 2002 Espoo, Finland, pp. 186-195.

[9] V. Pulkki, "Virtual Sound Source Positioning Using Vector Base Amplitude Panning", J. Audio Eng. Soc., Vol. 45, No. 6, June 1997, pp. 456 – 466.

[10] A. B. Ward, G. W. Elko, "Effect of loudspeaker position on the robustness of acoustic crosstalk cancellation", IEEE Signal Processing Letters, Vol. 6, No. 5, May 1999, pp. 106-108.

[11] http://sound.media.mit.edu/KEMAR.html

[12] V. Algazi, R. Duda, D. Thompson and C. Avendano, "The CIPIC HRTF database", in Proc. IEEE Workshop on Applications of Signal

Processing to Audio and Acoustics, New York, Oct. 2001, pp. 99 – 102.

[13] H. Moller, M. Sorensen, D. Hammershoi and C. Jensen, "Head Related Transfer Functions of Human Subjects", J. Audio Eng. Soc., Vol. 43, No .5, May 1995, pp. 300 – 321.

[14] H. Viste and G. Evangelista, "Binaural Source Localization", in Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx'04), Naples, Oct. 2004, pp. 145 – 150.

[15] H. Moller, "Fundamentals of Binaural Technology", Applied Acoustics, Vol. 36, No. 2, 1992, pp. 171 – 218.

[16] J. Blauert, "An Introduction to Binaural Technology", in Binaural and Spatial Hearing, R. Gilkey & T. Anderson, Eds., Lawrence Erlbaum, USA-Hilldale NJ,1995.

[17] C. Tsakostas, "Image Receiver Model: An efficient variation of the Image Source Model for the case of multiple sound sources and a single receiver" presented at the HELINA Conference, Thessaloniki Greece, 2004.

[18] M. R. Schroeder, "Natural Sounding Artificial Reverberation", J. Audio Engineering Society, Vol. 10, No 3, 1962.

[19]] J. A. Moorer, "About This Reverberation Business", Computer Music Journal, Vol. 3, No 2. 1979.

[20] W. C. Sabine, "Reverberation" originally published in 1900. Reprinted in Acoustics: Historical and Philosophical Development, edited by R. B. Lindsay. Dowden, Hutchinson, and Ross, Stroudsburg, PA, 1972.

7. APPENDIX

Following is a sample code in C++ for demonstrating the communication of a software application with the Amphiotik 3D Audio Engine, using the Amphiotik API.

```
//DEFINE AN AMPHIOTIK 3D ENGINE OBJECT
CATSDK m_atsdk;

//INITIALIZE THE APPLICATION
void init()
{
  //DEFAULT WORLD DEFINITION FROM PRESET
  m_atsdk.at_Wrld_Set_File("Room-
  Medium");

  //DEFINE AN AUDIO STREAM FROM A FILE
  int iIDSource1 = 0;
  m_atsdk.at_Ssrc_Add_File(iIDSource1,
  "C:\\AUDIO_FILE.WAV");

  /*ADD A VIRTUAL SOURCE, SPECIFY ITS
  POSITION, AND LINK IT WITH THE
  PREVIOUS AUDIO STREAM AND OPTIONALLY
  GIVE A DESCRIPTION*/
  int iID3DSource1 = 0;
  m_atsdk.at_Vsrc_Add(iID3DSource1,
  iIDSource1, 0, -1, 2.75, 1,"DESC");

  //ADD A VIRTUAL BINAURAL RECEIVER
  int iID3DSink = 0;
  m_atsdk.at_Vrcv_Add(iID3DSink);
  //AND SET ITS POSITION AND ORIENTATION
  m_atsdk.at_Vrcv_Set_Position_LookAt(
  m_iID3DSink, 0, 2, 1, 0, 3, 1);
}

//START PLAYBACK
void play()
{
  m_atsdk.at_Play();
}

//STOP PLAYBACK
void stop()
{
  m_atsdk.at_Stop();
}
```