Delivering Object-Based 3D Audio Using The Web Audio API And The Audio Definition Model

Chris Pike
BBC Research & Development
Dock House, MediaCiyUK
Salford, UK
chris.pike@bbc.co.uk

Peter Taylour
BBC Research & Development
Dock House, MediaCiyUK
Salford, UK
peter.taylour@bbc.co.uk

Frank Melchior
BBC Research & Development
Dock House, MediaCiyUK
Salford, UK
frank.melchior@bbc.co.uk

ABSTRACT

This paper presents an application that demonstrates object-based 3D audio rendering in the web browser using the Web Audio API. The application loads audio files containing object-based meta-data and provides head-tracked dynamic binaural rendering of the content to create an immersive 3D audio experience for headphone listeners. The user can interact with the rendering by muting individual audio objects and switching between the binaural rendering mode and conventional stereo rendering.

This application demonstrates the future of broadcast sound experiences over the web, where immersive content is rendered on the client and can be adapted to listener context, as page layout is adapted to device context today with responsive design.

Categories and Subject Descriptors

H.5.1: [Multimedia Information Systems] Audio input/output

General Terms

Algorithms, Experimentation, Human Factors, Standardization, Verification.

Keywords

Object-Based Broadcasting, Web Audio API, Audio Definition Model

1. INTRODUCTION

The BBC is working to provide more immersive and engaging media experiences for its audiences. Towards this goal, a topic of research interest in recent years has been 3D audio. Producing programmes in surround sound with an added sense of height and depth. This field of research is well developed and there is a wide range of techniques available e.g. [1][2][3].

This work has led to the development of object-based audio formats, where programme sound is delivered as separate elements with audio essence and accompanying time-varying meta-data to describe its role in the overall scene [4][5]. An object-based approach enables delivery of 3D audio in a format independent of the reproduction system and allows rendering decisions to be made by the client, which can tailor the reproduction to best create the intended experience on the listener's equipment. In a heterogeneous media environment, this will allow a content creator to produce and distribute programmes in one format, rather than manually rework content for each targeted reproduction scenario.

The Web Audio API [6] presents a means of realising this idea over the web. This paper presents an application that demonstrates object-based 3D audio rendering in the web browser using the Web Audio API. The application loads audio files containing object-based meta-data described using a new model called the Audio Definition Model (ADM) [5]. Using this meta-data it provides head-tracked dynamic binaural rendering of the content to create an immersive 3D audio experience for headphone listeners. The user can interact with the rendering by muting individual audio objects and switching between the binaural rendering mode and conventional stereo rendering.

This application demonstrates the future of broadcast sound experiences over the web, where immersive content is rendered on the client and can be adapted to listener context, as page layout is adapted to device context today with responsive design. It also serves to demonstrate the capabilities of the ADM for representing advanced audio formats.

2. 3D BINAURAL AUDIO

In conventional stereo panning the position of a sound source is controlled by adjusting the relative amplitude in the left and right channels [7]. When played on stereo loudspeakers this creates a simple but effective approximation of the inter-aural level difference changes that provide a psychoacoustic cue to sound source location in the horizontal plane. However when played on headphones conventional stereo panning creates an inside-the-head impression [8].

In contrast binaural rendering aims to recreate the complex effect that the human body has on a sound reaching the auditory system from a defined 3D position, to create a more realistic spatial impression for the listener. This involves frequency-dependent time and level differences between the left and right ears due to the scattering of a sound source off the human anatomy, notably the head and outer ears (pinnae) [9].

For any sound source at particular location relative to the listener these psychoacoustic cues are captured in a head-related transfer function (HRTF). By applying an HRTF to a sound source signal and reproducing the result on headphones, one can create the impression of the sound coming from this position [10]. Commonly a set of HRTFs measured at a range of source directions with constant distance is used to create a 3D binaural panner, with control of source azimuth and elevation. HRTFs vary for each individual due to their unique morphology but approximate localisation cues can be created with a generic HRTF set [11]. The HRTF set can have a significant effect on timbral and spatial quality of the listening experience [12].

The spatial impression of a binaural rendering can be improved by using head tracking. When the listener rotates their head, the tracking data is used to change the selected HRTFs in order to maintain constant perceived sound source position. Without head tracking the perceived location of binaurally rendered sound will rotate with the head. Head tracking improves plausibility but also leads to fewer front-back confusions and a better sense of out-of-the-head impression [13].

Binaural reverberation is also important for sound externalisation in binaural rendering [14]. This could be created with measured binaural room impulse responses but an artificial binaural reverberator would be more efficient, such as in [15]. Such a reverb could potentially be built from existing Web Audio components i.e. BiquadFilterNode, GainNode, and DelayNode objects.

2.1 3D Audio in the Web Audio API

The Web Audio API has a 3D audio engine using a global listener model, the AudioListener interface, and PannerNode objects to represent sound sources within an AudioContext instance. Each PannerNode object uses the position and orientation of the AudioListener as well as its own position and orientation during rendering to calculate rendering cues for distance and direction. Different distance and direction models can be chosen. Direction cues are provided using the panning models which convert a one-channel input to a two-channel output, either using equal power amplitude panning or HRTF panning.

Although PannerNode objects always incorporate the source and listener 3D positions and orientations, equal power panning mode can only give the impression of sounds moving between the left and right ears when listening on headphones. The HRTF panning model of the PannerNode allows true 3D positioning of sounds for headphone listeners using the binaural technique. The HRTF set is currently fixed in the browser implementation, it cannot be changed by the web application developer and is not standardised as part of the Web Audio API specification. Currently the authors use a custom build of the Chromium browser [16] with their own chosen HRTF set to achieve the desired sound quality. Future versions of the Web Audio API could support provision of a SOFA file [17] containing a set of HRTF measurements for use in the HRTF panner, allowing developers greater control over rendering quality or even personalisation for the individual listener.

Distance cues are created using one of a set of simple gain decay functions. Gain changes due to source directivity can also be controlled using a simple “sound cone” model common to game audio engines. These features are useful in interactive gaming applications, but for rendering of object-based 3D audio programmes the level of each object should be set in the audio essence itself to ensure that the creative intent of the producer is conveyed. Therefore in this application these features are turned off.

3. ADM & BROADCAST WAVE FORMAT

The Audio Definition Model (ADM) is a recently published meta-data model for describing audio formats [5]. It can be included in an audio file to explicitly describe the format of content contained in the file, enabling an application to handle it appropriately.

The conventional way of representing audio content is a channel-based approach where the reproduction method is implicit in the content format. For example, a two-channel audio file implies that the content should be reproduced on a stereo pair of loudspeakers placed in front of the listener at ±30° azimuth. With the development of many varieties of multichannel audio for surround sound the meaning of each channel has become ambiguous. The ADM introduces explicit description of channel-based audio formats, defining the role of each channel. Known loudspeaker layouts can be referenced or novel layouts can be described using 3D coordinates. If the specified loudspeaker layout is not available for reproduction, this enables the content to be adapted appropriately to the system available.

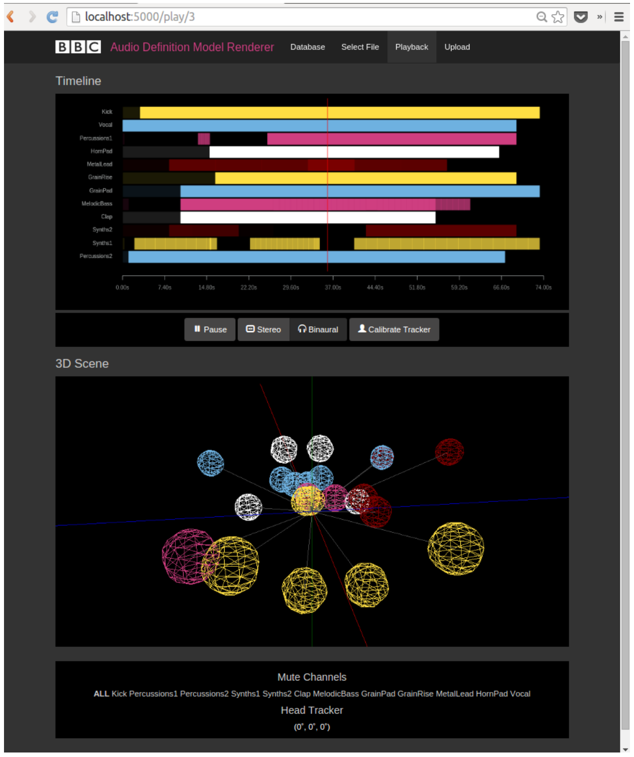


Figure 1 - Screenshot of the application’s playback interface

The ADM can also describe object-based content. Here the streams of audio content no longer correspond directly to intended loudspeaker positions rather elements within the scene that have a descriptive label and a specified set of meta-data including its position; these can also vary over time. An object-based audio programme might contain, for example, a collection of objects relating to the instruments in a musical ensemble, or the dialogue and effects elements of a drama.

The web application presented in this paper uses the object-based part of the ADM to describe programmes stored in Broadcast Wave Format (BWF) files. The BWF is an extension to the WAVE format, containing additional header chunks, designed to facilitate interchange within the production environment [18]. The *axml* header chunk enables carriage of ADM meta-data in XML [19].

Time varying positions of objects within the programme are contained within blocks of meta-data. Each block provides the rendering instruction for an object for a specified duration of time. For example, a block of meta-data could describe the position of an object for 10 seconds into the performance, at which point a second block could specify the position of the object for the next 30 seconds.

4. ADM OBJECT-BASED AUDIO PLAYER

A web application has been built to play 3D audio programmes from BWF files containing ADM meta-data. The audio output is rendered according to the ADM meta-data in real-time on the client-side using the Web Audio API. Figure 1 shows the user interface presented to the listener on playback of a file.

In addition to playback controls, the application offers a timeline visualisation in an HTML5 canvas, where each block of meta-data is represented by a coloured rectangle and the active blocks are highlighted. The current position of each object is set in a PannerNode and is also displayed in a 3D visualisation of the scene using THREE.js [20], with the listener at the origin. The objects are represented by spheres with colours matching the timeline visualisation. The interface allows users to switch between stereo and binaural playback and calibrate a head tracking system.

The client-server application structure is pictured in 2. To reproduce a programme the renderer must interpret the ADM meta-data. This is handled by JavaScript code, running client-side in the browser. This data is then used to control the Web Audio API processing, which produces a real-time stereo or head tracked binaural rendering of the programme, depending on the selected option.

The user can upload BWF files to the application server, where they are parsed and processed for use in a browser client. The server is written in python using the Flask framework [21]. The ADM meta-data is extracted from the header and parsed on the server using an open-source python library [22]. The XML is converted to JSON and stored in an SQLite database [23]. The SoX audio manipulation tool [24] is then used to split the BWF audio tracks, which are stored as separate WAVE files. This step is required because existing browser implementations were found to limit the number of tracks that could be decoded in a single file when using the AudioContext's decodeAudioData method. This limitation is not specified in the Web Audio API itself.

A user can select any uploaded and parsed programme using the file selection interface. The client application then loads the associated one-track audio files and JSON meta-data from the server.

The JavaScript client uses two main components to render the programme, an AudioObject class and a Scheduler class. An AudioObject instance is created for each audio track extracted from the original BWF file. Each instance holds a single AudioBufferSourceNode and the processing nodes required to render the audio. A single Scheduler instance produces a schedule of rendering instructions from the JSON meta-data and then controls the render on playback, setting attributes of the Web Audio API nodes.

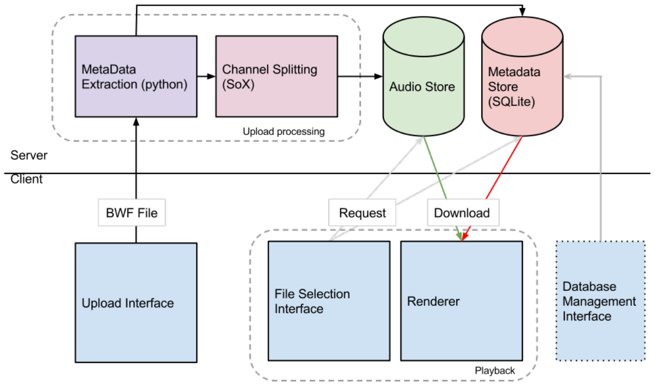


Figure 2 - Client-server application structure

4.1 Setting up the Node Graph

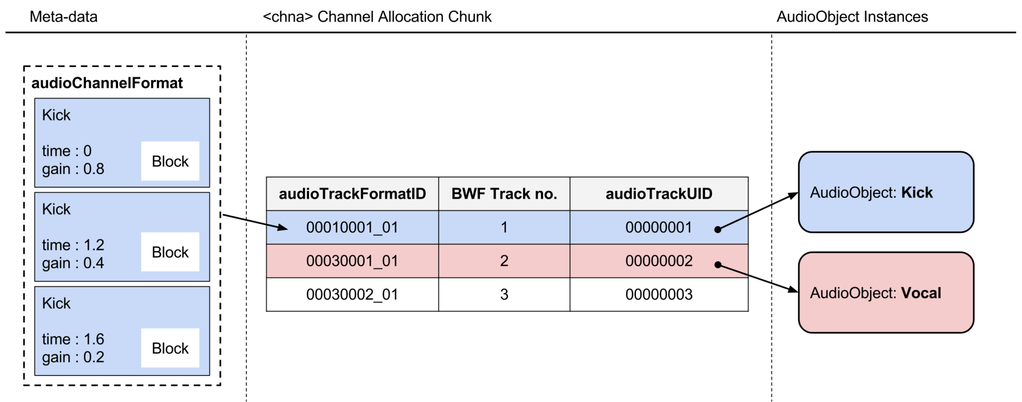
The Web Audio node graph is constructed by creating instances of the AudioObject class. Instantiating an AudioObject creates an AudioBufferSourceNode connected to a GainNode and a PannerNode. The PannerNode is then connected to the AudioDestinationNode. The AudioObject class provides methods for updating attributes such as the GainNode *gain* and the PannerNode *panningModel*, and *position*. The methods perform transformations between the ADM and Web Audio coordinate systems where required and handle scheduling of attribute updates in advance.

On loading a file the client creates an AudioObject instance for each track in the BWF file. A recording of a musical ensemble, for example, might contain an audio track for each instrument, accompanied by ADM meta-data describing how each instrument should be rendered. In this case an AudioObject instance would be created for each instrument and the corresponding BWF track decoded and attached to the AudioBufferSourceNode *buffer*. The resulting node graph is a collection of AudioObjects, each containing an AudioBufferSourceNode, GainNode and PannerNode, then connected to the AudioDestinationNode.

4.2 Building a Schedule

Before playback can commence the client must process the meta-data received from the server and create a schedule of timed rendering instructions to give to the Web Audio nodes at the appropriate times during playback, this is done by the Scheduler class.

Within the ADM, blocks of meta-data that refer to a single audio channel are grouped together by reference to the same AudioChannelFormat identifier. With object-based audio each



audioTrackFormatID	BWF Track no.	audioTrackUID
00010001_01	1	00000001
00030001_01	2	00000002
00030002_01	3	00000003

Figure 3 - Illustration of the link between ADM elements and JavaScript objects in the web application

AudioChannelFormat represents an audio object and has a list of metadata instructions describing time-varying rendering parameters.

Each AudioChannelFormat is linked back to a track in the BWF file via a table within the header called the Channel Allocation Chunk, as illustrated in Figure 3. The data in the Channel Allocation Chunk is used to link meta-data blocks with the correct AudioObject instance in the application, so that the rendering instructions relate to the audio asset they were intended for. The Scheduler makes a link between each meta-data block and the corresponding AudioObject and then sorts these rendering instructions chronologically. This process is illustrated in Figure 4. The result is a list of timed instructions, each with reference to an AudioObject.

4.3 Accurate Playback Scheduling

A key problem addressed by the Web Audio API is the inability to accurately schedule JavaScript events in the browser. Without accurate timing, correctly producing an audio rendering from a schedule of events would not be possible.

As the Web Audio API offers sample accurate scheduling outside of the main JavaScript execution thread, setting the start time of an AudioObject's AudioBufferSoundNode relative to the AudioContext.currentTime property is sufficient to ensure an entirely reproducible and accurate render of the ADM instructions.

To avoid the inflexibility that would be caused by scheduling all events in advance on starting playback, regular calls to setTimeout during playback are used to add small chunks of the schedule to the Web Audio event queue [25]. This method ensures that only imminent events are scheduled in the AudioContext itself and enables playback functions such as pause and rewind to be neatly implemented. To ensure that delays to the JavaScript event-loop will not cause events to be missed, each call to the Window setTimeout method schedules events occurring after the next timeout is due to occur.

The AudioBufferSourceNode's start method provides an attribute for scheduling playback ahead of time, and the AudioParam setValueAtTime method is used to queue updates to the GainNode. This approach cannot be used to set the position parameters of the PannerNode as they do not use the AudioParam interface but must be set instantaneously through the PannerNode setPosition method. A setTimeout callback is used instead, therefore accurate timing of object position changes cannot be guaranteed.

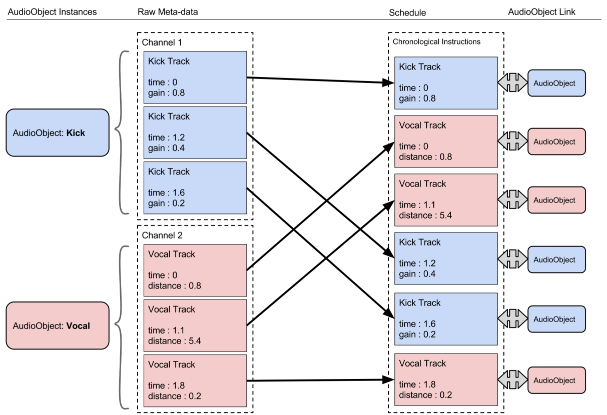


Figure 4 - Creating the playback schedule for the renderer (right) from AudioObject meta-data (left)

4.4 Head Tracking

To improve the spatial impression in the web application, it interfaces with a head-tracking device. The tracker's orientation data are used to adjust the orientation of the Web Audio API's AudioListener to match the rotations of the user. The position of the AudioListener is used by the Web Audio API to adjust the panning of objects such that sound stage remains stationary.

In this prototype, the tracker runs in a small standalone C++ application on the client machine and sends head orientation data to the web browser using a simple WebSocket interface [26]. An inertial tracking device is currently used [27], connecting via USB. Different trackers could also be used provided that the same WebSocket interface is implemented.

The use of head tracking applies a constraint on the latency of the system, to prevent distracting delay when the sound scene is updated according to head movement. Lindau found that a latency <53ms was not detectible by any experienced listener [28]. Besides processing latency in the Web Audio API and the head tracking hardware, the WebSocket and head tracker connection interface will add further latency, for example state-of-the-art consumer head tracking devices use a Bluetooth LE connection, e.g. [29].

5. DISCUSSION

The Web Audio API is used to create a client-side renderer for the BWF audio file format with accompanying ADM meta-data. The prototype provides a proof-of-concept example of how the web could be used deliver object-based audio to a large audience. It also successfully demonstrates use of the browser as a possible rendering device, with support for playback of 3D audio content binaurally rendered over headphones.

If the browser is to be considered as a rendering device for object-based audio it must be able to accurately control playback of individual audio assets and follow time-varying meta-data describing the performance. The BWF programme material tested is rendered with the appropriate spatial layout and timing. However in its current state the Web Audio API does not offer the quality and performance required for a full object-based rendering client. Accurate timing of object position changes cannot currently be guaranteed and the quality of binaural rendering is strongly influenced by the HRTF dataset used in the browser implementation.

In its current form the web application uses the GainNode and PannerNode objects to render sound according to the ADM meta-data describing the volume and position of an object. Additional features for the ADM player application could be added in the future using existing Web Audio features. Custom panning could also be implemented using an AudioWorker to extend rendering to multichannel loudspeaker systems, perhaps even for arbitrary speaker layouts.

To deliver live object-based broadcasts over the web it would be necessary for the application to support streaming. Media Source Extensions [30] could be used with the Web Audio API MediaElementAudioSourceNode to allow streaming, as in [31].

The application could be of use as a framework for future research into delivering new audio formats and in development of new user experiences that take advantage of the benefits object-based broadcasting and the Web Audio API.

6. CONCLUSION

The Web Audio API makes it possible to render object-based 3D audio in a web browser without requiring plug-ins. A prototype application has been presented which demonstrates these capabilities and the features of the API that are used. The server-client application parses and renders BWF files that use the ADM object-based format description. The Web Audio API is used to control the gain and position of audio objects rendered either using conventional stereo equal-power panning and or to 3D binaural audio using HRTF panning and head tracking. The application changes the rendering parameters in real-time during playback according to a schedule of object-based meta-data instructions.

The application could be further developed to provide more object-based 3D audio features using existing Web Audio functionality. In the future the Web Audio API could be improved to allow the developer more control over the accuracy and quality of object-based 3D audio rendering.

7. REFERENCES

[1] Pulkki, V. 1997. Virtual Sound Source Positioning Using Vector Base Amplitude Panning. *Journal of the Audio Engineering Society*, 45(6), 456–466.

[2] Daniel, J. 2003. Spatial sound encoding including near-field effect: Introducing distance coding filters and a viable, new ambisonic format. *Proceedings of the 23rd International Conference of the Audio Engineering Society* (Copenhagen, Denmark, May 2003).

[3] Jot, J.-M. 1999. Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces. *Multimedia Systems*, *7*(1), 55–69. DOI= http://dx.doi.org/10.1007/s005300050111

[4] Armstrong, M., Brooks, M., Churnside, A., Evans, M., Melchior, F. and Shotton, M. 2014. Object-based broadcasting – curation, responsiveness and user experience, *Proceedings of IBC2014* (Amsterdam, September 2014).

[5] EBU. 2014. Tech 3364: Audio Definition Model. https://tech.ebu.ch/docs/tech/tech3364.pdf

[6] Web Audio API. W3C Working Draft. http://www.w3.org/TR/webaudio/

[7] Blumlein, A. 1932. Improvements in and relating to sound-transmission, sound-recording and sound-reproducing systems. *GB Patent 394 325*.

[8] Toole, F. E. 1970. In-Head Localization of Acoustic Images. *Journal of the Acoustical Society of America* 48(4), 943–949. DOI= http://dx.doi.org/10.1121/1.1912233

[9] Møller, H. 1992. Fundamentals of binaural technology. *Applied Acoustics*, 36(3–4), 171–218. DOI= http://dx.doi.org/10.1016/0003-682X(92)90046-U

[10] Wightman F. L. and Kistler, D. J. 1989. Headphone simulation of free-field listening. II: Pyschophysical validation. *Journal of the Acoustical Society of America* 85(2), 868–878. DOI= http://dx.doi.org/10.1121/1.397558

[11] Møller, H., Sørensen, M. F., Jensen, C. B., and Hammershøi, D. 1996. Binaural Technique: Do We Need Individual Recordings? *Journal of the Audio Engineering Society* 44(6), 451-469.

[12] Merimaa, J. 2009. Modification of HRTF Filters to Reduce Timbral Effects in Binaural Synthesis. *Proceedings of 127th AES Convention* (New York, USA, October 2009).

[13] Brimijoin, W. O., Boyd, A. W., and Akeroyd, M. A. 2013. The contribution of head movement to the externalization and internalization of sounds. *PLoS One* 8(12) (January 2013). DOI= http://dx.doi.org/10.1371/journal.pone.0083068

[14] Begault, D. R., Wenzel, E. M., & Anderson, M. R. 2001. Direct Comparison of the Impact of Head Tracking, Reverberation, and Individualised Head-Related Transfer Functions on the Spatial Perception of a Virtual Speech Source. *Journal of the Audio Engineering Society*, *49*(10), 904-916.

[15] Jot, J.-M., Larcher, V., & Warusfel, O. 1995. Digital Signal Processing Issues in the Context of Binaural and Transaural Stereophony. *Proceedings of 98th Convention of the Audio Engineering Society* (Paris, France, February 1995).

[16] Chromium browser. http://www.chromium.org/

[17] Majdak, P. *et al.* 2013. Spatially Oriented Format for Acoustics: A Data Exchange Format Representing Head-Related Transfer Functions. *Proceedings of 134th Convention of the Audio Engineering Society* (Rome, Italy, May 2014).

[18] EBU. 2011. Tech 3285: Specification of the Broadcast Wave Format. https://tech.ebu.ch/docs/tech/tech3285.pdf

[19] EBU. 2003. Specification of the Broadcast Wave Format A format for audio data files in broadcasting Supplement 5: <axml> Chunk. https://tech.ebu.ch/docs/tech/tech3285s5.pdf

[20] three.js JavaScript library. http://threejs.org

[21] Flask python framework. http://flask.pocoo.org

[22] BBC R&D ADM XML python library. https://github.com/bbcrd/adm_xml

[23] SQLite3 database library. http://www.sqlite.org

[24] SoX - Sound eXchange audio processing software. http://sox.sourceforge.net

[25] Wilson, C. 2013. A Tale of Two Clocks - Scheduling Web Audio with Precision. *HTML5Rocks.* Web resource accessed on 22nd October, 2014: http://www.html5rocks.com/en/tutorials/audio/scheduling/

[26] The WebSocket Protocol. Internet Engineering Task Force RFC6455. Web resource accessed on 23rd October, 2014: https://tools.ietf.org/html/rfc6455

[27] Razor Attitude and Heading Reference System. https://github.com/ptrbrtz/razor-9dof-ahrs

[28] Lindau, A. 2009. The Perception of System Latency in Dynamic Binaural Synthesis. *Proceedings of NAG/DAGA 2009* (Rotterdam, The Netherlands, March 2009).

[29] Intelligent Headset. https://intelligentheadset.com/

[30] Media Source Extensions. W3C Editor's Draft. https://dvcs.w3.org/hg/html-media/raw-file/tip/media-source/media-source.html

[31] Pike, C., Nixon, T., Evans, D. 2014. Under Milk Wood in Headphone Surround Sound. *BBC R&D Blog.* Web resource accessed on 23rd October, 2014: http://www.bbc.co.uk/rd/blog/2014-10-under-milk-wood-in-headphone-surround-sound