Proceedings Paper:
McArthur, Angela, Van Tonder, Cobi and Knight-Hill, Andrew (2021) A survey of 3D audio through the browser : practitioner perspectives. In: 2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA). IEEE , pp. 1-10.

https://doi.org/10.1109/i3da48870.2021.9610839

A survey of 3D audio through the browser: practitioner perspectives

Angela McArthur
SOUND/IMAGE Research Group
University of Greenwich
London, UK
a.mcarthur@gre.ac.uk /
Orchid ID - 0000-0002-4564-4737

Cobi van Tonder
Dept of Theatre, Film, Television & Interactive Media
University of York
York, UK
cobi.vantonder@york.ac.uk

Leslie Gaston-Bird
Institute of Sound Recording
University of Surrey
Guildford, Surrey
l.gaston-bird@surrey.ac.uk

Andrew Knight-Hill
SOUND/IMAGE Research Group
University of Greenwich
London, UK
a.hill@gre.ac.uk / Orchid ID - 0000-0003-1521-0396

2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA) | 978-1-6654-0998-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/I3DA48870.2021.9610839

***Abstract* — This paper examines the current eco-system of tools for implementing dynamic 3D audio through the browser, from the perspective of spatial sound practitioners. It presents a survey of some existing tools to assess usefulness, and ease of use. This takes the forms of case studies, interviews with other practitioners, and initial testing comparisons between the authors. The survey classifies and summarizes their relative advantages, disadvantages and potential use cases. It charts the specialist knowledge needed to employ them or enable others to.**

The recent and necessary move to online exhibition of works, has seen many creative practitioners grapple with a disparate eco-system of software. Such technologies are diverse in their both their motivations and applications. From formats which overcome the limits of WebGL's lack of support for Ambisonics, to the creative deployment of Web Audio API (WAA), to third-party tools based on WAA, the field can seem prohibitively daunting for practitioners. The current range of possible acoustic results may be too unclear to justify the learning curve.

Through this evaluation of the current available tools, we hope to demystify and make accessible these novel technologies to composers, musicians, artists and other learners, who might otherwise be dissuaded from engaging with this rich territory. This paper is based on a special session at Soundstack 2021.

Keywords—3D audio, Web Audio, Ambisonics, Browser, Practice, Survey

I. Introduction

The last year and a half has witnessed an increased presentation of sound and musical works online. The pandemic has catalysed the development of these tools as promoters seek to employ more immersive impressions for audiences. Whilst this has afforded greater potential audiences for event organisers and artists, there are associated challenges in translating (or creating) works for this form of exhibition. This paper provides an overview of the current tools available for creative practitioners to work with spatial sound through the browser. Such tools overcome the limitations of pre-rendered binaural (stereo) streams, which may provide immersive sound but not dynamically (e.g. without head-tracking so that the sound field moves with head movement, decreasing immersion). Existing technologies can provide a greater range of possibilities, including an enhanced sense of immersion. Increased awareness of these, and hopefully an increased uptake, is the motivation for this paper.

The tools are grouped into three categories – Web Audio API (WAA), Ambisonics, and WebRTC. These groupings may not at first glance seem intuitive to practitioners, however they designate important components in the architecture of the tools. These components limit or potentiate their affordances

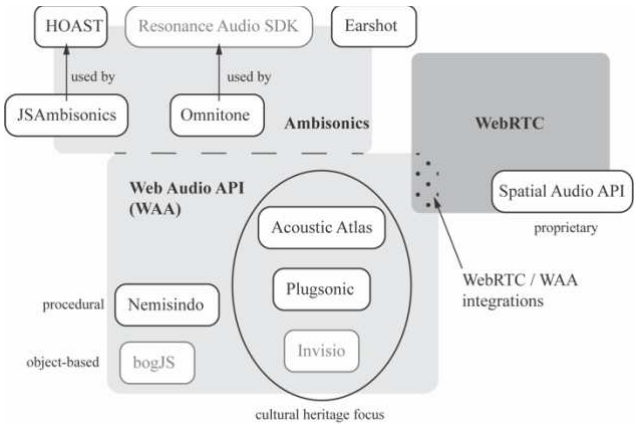

Fig. 1. A map of the tools surveyed in this paper

in fundamental ways. The tools described below were presented at Soundstack 2021, though do not represent an exhaustive sweep of available tools. A section outlining further tools is included towards the end of the paper for this reason. All tools (with just one exception) are free to use and/or open source. See Fig. 1 for an overall representation of the eco-system of tools set out in this paper.

In section II we outline the criteria for evaluation we applied when evaluating the tools. In section III we introduce WAA and discuss tools which extend WAA and provide certain spatial affordances. In section IV we turn our attention to tools which make an ambisonic representation of the sound field possible. Section V discusses one tool built using WebRTC (as this tool is unique in many ways and provides a useful point of comparison). Section VI briefly outlines tools which are available but were not presented at Soundstack 2021.

II. Criteria for Evaluation

A. Usefulness

The way practitioners use technologies is often symbiotic with their affordances. While it is not feasible to anticipate every use to which a creative practitioner could apply a given tool (indeed, the creative use of tools is what often extends their application beyond the original intent of developers) the affordances of the tools are addressed in pragmatic terms. Usefulness here therefore comprises two things: the number of barriers which the tool overcomes effectively on the behalf of a practitioner, and the range of use cases it affords (a diverse range of options being preferable).

B. *Ease of use*

This criterion designates an absence of barriers to using the tool for a non-specialist. Most practitioners have an intermediate level of technical knowledge if working with spatial sound. They would need an extended range of this knowledge, to consider working through the browser. Web development is in itself a specialism and often requires a large investment of time and effort to achieve reliable results (the challenges of working across operating systems, browsers, etc. are beyond the scope of this paper).

C. *Support & documentation*

Projects which produce open-source tools are often funded for finite periods of time. Active support and maintenance for such tools beyond the life of the project can be uncommon. Without communities of users to support one another, and/ or adequate documentation aimed at non-specialists, implementing specific artistic goals with such tools can be difficult. The burden of learning increases. This impacts a practitioner's experience, a project's time frame, and thus use of the tool.

III. WEB AUDIO API (WAA)

WAA is a high-level JavaScript (JS) API, a tool (or more accurately, set of tools) which enable optimised audio synthesis and processing, directly in the browser - the kinds of capabilities "found in modern game audio engines as well as some of the mixing, processing, and filtering tasks that are found in modern desktop audio production applications" [1] These range from basic processing like equalisation, to spatialisation, to more interactive processing such as real-time microphone input. WAA processing is done via JS control of Assembly/C/C++ code, and is implemented in all major browsers [2]. One can also write custom effects directly using JS. These range from basic processing like equalisation, to spatialisation, to more interactive processing such as real-time microphone input. WAA was developed by the World Wide Web Consortium (W3C) to afford more complex audio applications through the browser than was possible with HTML5's audio element. HTML5 had led to media players (plugins such as QuickTime or Flash) becoming obsolete. However, the audio streaming and playback afforded by HTML5 was basic. The development of WAA overcame this, and afforded many use cases as a result (see [3] for examples), enabled by its modular flexibility and integration with other web standards and tools (e.g.WebGL, WebXR).

"..we can write the code needed for our project with ease. If we run into a problem we will usually find a good solution to it on the web. We don't have to spend our time with learning […] some poorly documented proprietary audio engine" [5].

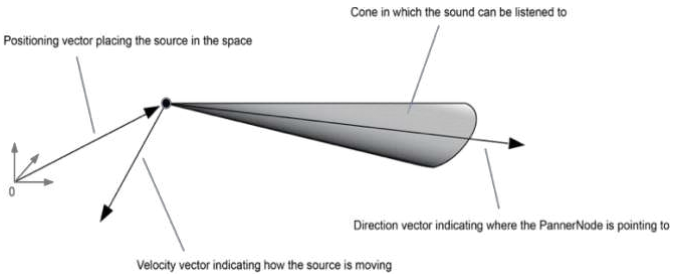

Fig. 2. WAA PannerNode Visualization. Source: [4]

WAA's PannerNode enables spatialisation of an audio stream, relative to an AudioListener, for an AudioContext instance. It renders distance and direction cues in cartesian space, based on the stream's position and orientation, relative to the AudioListener's. Different distance and direction models are available (though distance modelling is limited to attenuation and does not account for spectral changes, which is limiting). PannerNode objects (the audio streams) have an orientation vector (which direction the stream is facing) and a sound cone (how directional the stream is). The AudioListener (the listener's ears) have vectors representing their facing direction. These features alone provide a level of spatial control which can be helpful to the practitioner new to working through the browser (see Fig. 3).

The PannerNode requires that mono-to-stereo (if all inputs are mono) and stereo-to-stereo (where any input is not mono) panning is supported. PannerNode currently offers two panning algorithms: 'equal-power' and 'HRTF' (the default, which operates as a stereo-only panner). The HRTF ((Head-related Transfer Function) requires particularly optimized convolution [1], which is a key concern for any browser-based system. HRTF panning is more expensive than the 'equal-power' algorithm, but does provide a spatialized sound more true to perceptual experience [1] than the 'between the ears' experience of the former [6].

WAA currently specifies a maximum channel count of 32, which constrains reproduction of multitrack signals. This means streaming > 4OA (Fourth Order Ambisonics) is not currently possible. In theory, a custom Chromium [7] browser could be built for a greater number of (arbitrarily specified) channels, simply by changing the maximum channel count value before compiling the browser, though this is likely to be a prohibitive step for most practitioners.

WAA has been implemented using one fixed set of HRTFs (which developers cannot alter) from the IRCAM Listen database [8] as described in [9], in Google Chrome and Mozilla Firefox. This set is not identified in WAA documentation. Although issues can arise from non-individualized HRTFs (e.g. poor externalization) these issues can be offset if head-tracking is employed [10]. The BBC Audio R&D team undertook a custom build of the Chromium browser to implement a HRTF set of their choosing [6] though as previously stated, this is likely a prohibitive step for most

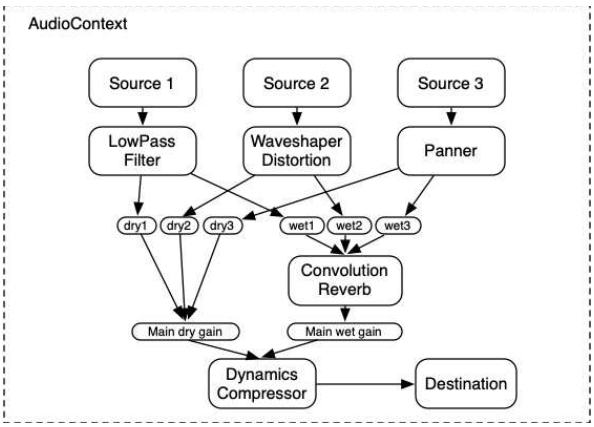

Fig. 3. An example of modular routing Source: [1]

practitioners. Work has been done to extend WAA in order to have custom HRTF capability (for example, as FIR filters via a Binaural-FIRNode which accept SOFA [11] HRTF files, see [9] for details) though implementing these this requires further development time and skills. The Ambisonic tools outlined in section IV also afford custom-loaded HRTFs. Future versions of WAA may support the SOFA file format, for loading custom HRTFs, giving developers more control over this aspect of binaural reproduction [6].

General limitations (-) and affordances (+)

- Maximum channel count of 32 limits ambisonic reproduction to <= 4OA

- One fixed set of HRTFs unless custom Chromium browser is built (a prohibitive step)

- Distance models based only on attenuation only, not combined with frequency changes, thus perceptually crude

- Without customization, the spatial parameters of WAA are aggressive (though obvious cues can be valuable at times)

- Implementations of WAA vary across browsers (e.g., channel-ordering of audio files) which can affect the behaviour of practitioners' work, making it unreliable

+ Widely supported and documented. Online communities make this a good starting place for practitioners

+ WAA uses a modular audio graph paradigm (inter-connecting nodes, a kind of visual programming, see Fig. 3), which may make it easier for practitioners to engage with (for a helpful interactive demo of this visit [12].

+ Native spatialization is obvious for non-expert listener audiences so can have immediate impact

+ Future versions of WAA may support custom HRTFs

+ For the practitioner with more advanced JavaScript skills, custom effects can be created. A key benefit of WAA is its extensibility.

A. Nemisindo

Nemisindo (formerly FXive) is a web-based, procedural audio synthesis framework for sound design. It builds on WAA with customized JS processors and functions. Its procedural nature makes it stand out here. Many of the Nemisindo's sound synthesis models are original. Some are based on work that has been published [13]–[16]. A useful list of references used for the project have been made available at [17] and key examples include [18]–[20].

The site provides a library of "synthesis models, audio effects, post-processing tools, temporal and spatial placement functionality for the user to create the scene from scratch" [21, p. 1]. The procedural, real-time nature of the platform enables a practitioner to shape the sound during its creation. These can then be post-processed (including spatialization) using the browser interface. In addition, 680 presets are available with an option to download a snippet.

Each model and effect is encapsulated using the JSAP audio plugin standard [22]. This affords flexibility through the creation of complex audio graphs, which can be connected in various configurations. The system's architecture allows for the models to be chained together.

The parameters for manipulating a sound's qualities vary with the sounds themselves, and include physical (e.g. 'density') and semantic (e.g. 'warmth') properties. The 'fan' for example, offers parameters such as 'motor ratio, 'fan pulse width', and 'brush level', while the model for 'wind' offers 'gustiness', 'squall', and 'branches'. Every model can be processed with EQ, reverb, delay, distortion, and compression. There are also controls for 'spatialization', 'random' (which randomizes the slider values), and 'trigger' (which permits triggering at specific times for each model's parameters, the presets, and for the randomizer). The randomizer and trigger combined form a useful way of introducing changes over time, thereby making sounds less 'synthesized'.

The spatialization tool offers sliding controls for room width and height, as well as for source position (see Fig. 4). These panning functions utilize the WAA PannerNode. The spatialization model also accounts for source direction, orientation, velocity and provides parameters for setting the distance model.

Helpfully, users can record and render the sounds they create to a downloadable stereo .wav file. It should be noted that Nemisindo has been optimized for Chrome and may not work well on other browsers.

Limitations (-) and affordances (+)

- The interface is at times a little clumsy, moving between different tools, audio playback and effects can be awkward

- The app is standalone. VST integration is planned but currently not available

- Although the target demographic is sound designers, a specific sub-group of amateur designers is the ideal audience. This is because professional sound designers require a specific workflow, seamless integration with a DAW, and the ability to insert sounds at specific timecodes

- Procedural, therefore potentially computationally demanding (though sounds can be downloaded to mitigate against this)

+ Procedural, with all the benefits of real-time synthesis

+ The interface is easy to use and provides user-friendly, real-time manipulation of parameters

+ Good documentation. A help button is available on every screen, and a FAQ section is also available

+ The sounds are convincing enough for a video game environment

+ The program is a good instructional tool in itself

+ Avoids pitch-shifting and time-stretching sample-based audio, which can be limiting

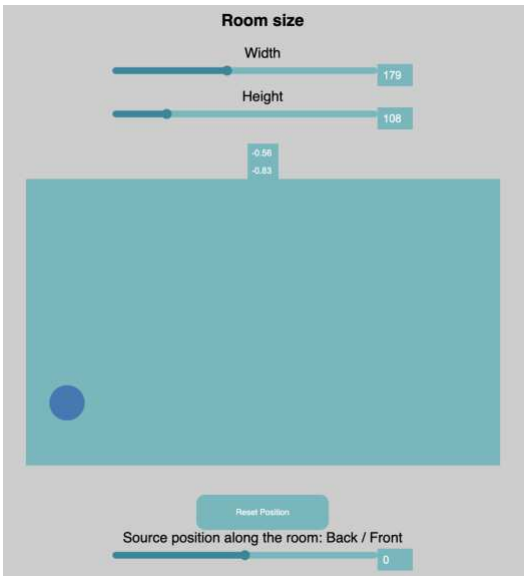

Fig. 4. Nemisindo Spatialization Interface

B. *The PlugSonic Suite*

Pluggy is an open-source social platform for creative content related to cultural heritage. It allows users to both store 'assets' and 'exhibitions'. Assets can be individual images, audio files, 3D models or video files, whilst exhibitions are categorized in Blog Stories, Tours, Soundscapes, AR/VR, Games and Timelines. The Pluggy platform offers useful tools called pluggable applications to edit, organize, license and share content. These free applications are Pluggy3d Suite (for creating AR/VR experiences), Pluggy Pins (for creating self-guided tours), Games Hunter (a video game authoring App that provides the basic bricks to develop without specific programming knowledge collaborative games) and the PlugSonic Suite (for creating soundscapes) [12].

The wider aim of this project is to encourage active participation in digital cultural heritage activities by creating immersive experiences, with easy to use, free tools to allow users to share their local knowledge and everyday experience with others. Here the focus is on describing the tools in terms of their spatial audio capabilities. The PlugSonic Suite [23] allows the user to edit samples, create soundscapes, create 3D objects, and integrate them into the online exhibitions hosted within the Pluggy social platform. From editing and processing, to licensing, combining into exhibition and sharing, everything is done from the user account in the browser. See [12] for examples and tutorials.

The PlugSonic Suite consists of PlugSonic Sample, PlugSonic Soundscape, Soundscape Experience Web, and Soundscape Experience Mobile. PlugSonic Sample is a sound file editor that includes basic editing functions as well as licensing tools to create individual audio samples stored as the previously mentioned assets. These assets next can be used in the PlugSonic Soundscape to create and experience 3D interactive soundscapes within the browser and on touch-enabled devices. It allows various details such as positioning sounds, setting whether they loop or play once, spatiality attributes such as panning and loudness relative to an object, and volume. It allows for setting interaction areas (when a visitor enters this area a sound becomes audible for example) and interaction behaviours (the sound fades in, or triggers at full volume for example). There are timing settings allowing for various time controls, such as order in which a series of samples are played. Soundscape Experience Web allows visitors to experience exhibitions in the browser, using the mouse or keyboard arrows to navigate and listen (see Fig. 5). Soundscape Experience Mobile enables navigating soundscapes using touch-based interfaces. Alternatively, it enables an immersive virtual experience delivered in real-world or AR environments. In this case, users explore a soundscape according to their movements in the real-world captured using ARkit. The 3D audio simulation in real-time according to the relative position of the device with respect to the detected ground plane. Spatialization is performed by the 3DTune-In audio toolkit [24].

Limitations (-) and affordances (+)

- When moving through an exhibition, it would be good if there is some sort of UI barrier/boundary condition to avoid the user moving through the sound source itself (a black dot) as this sometimes results in distortion

- If selecting individualized HRTFs, it would be useful to have test playback using an audio reference, to discern which HRTF is preferred. With unfamiliar /artistic sound files, it is hard to judge whether the experience reflects the spatial quality of the recording, or the selected HRTF

- The documentation places a strong emphasis on 'European Heritage'. The authors think it would be important to include a global audience and heritage sites/projects from all over the world

+ Having an entire suite allows even novices with basic technical skills access to tools that allow them to prepare samples, all the way up to preparing an exhibition in a spatial audio context

+ Comprehensive documentation and an easy-to-use interface

+ For playback settings, the visitor can control parameters such as 'head circumference'

+ The visitor can choose from various HRTFs to individualize the binaural experience

+ Audio sources are clear in terms of directivity, localization and disambiguation

+ Offers user accounts for creators to edit, save and share work in an online platform

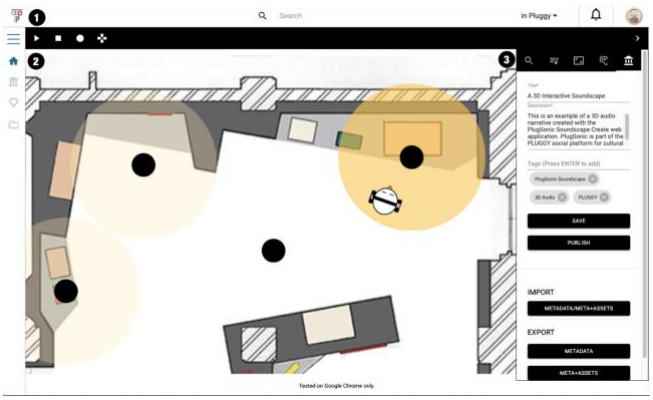

Fig. 5. PlugSonic Soundscape Create User Interface

IV. AMBISONICS

The tools designated in the following section make an ambisonic representation of the sound field possible. They are based on WAA, specifically the GainNode, ConvolverNode (used during binaural decoding) and ChannelMergerNode (which groups audio channels into single streams). However through Ambisonics, these tools use a different, scene based, paradigm. One key advantage of this approach for spatial sound representation, is computational efficiency. This is particularly helpful for audio-visual experiences, interactive / real-time experiences, and/ or cases where there are many sources, where source panning is frequent, or manipulations of the entire scene are likely. In short, where there are high demands placed on computational resource. The impact of this benefit is heightened for some hardware configurations, though becomes more complicated overall, due to the unpredictability of the client's system, an inherent issue with web development. For this reason, practitioners may decide to decode the ambisonic signal at the server side, rather than render the binaural signal independently on each client.

Currently only <= 4OA is possible due to the limit of 32 audio channels prescribed in the WAA specification. Meanwhile many tools are being developed for working offline in 5OA - 7OA. With the Eignenmike's 64 channel ambisonic microphone being released this year, a 7OA end-to-end production workflow (offline) is possible. IEM's ambisonics plugin suite [25], Aalto's SPARTA [26], and Ircam's Spat externals [27], are available for free. This is important for practitioners working creatively, while workflows remain uncertain. However, the ability of the browser to reproduce these is currently limited to static, pre-rendered streams.

Limitations (-) and affordances (+)

- Only <= 4OA currently possible due to the limit of 32 audio channels in the WAA specification

+ Relatively inexpensive, computationally

+ Custom HRTFs can be loaded

+ For practitioners who are familiar with Ambisonic representations, practices and workflows, their existing body of knowledge can be built upon, with less additional learning

A. JSAmbisonics

This JavaScript library comprises modules for real-time spatial audio processing. The extensive set of modules correspond to typical ambisonic production and manipulation processes (e.g. encoding, rotation, binaural decoding). It is deployed via Node.js, and each 'node' can be used independently or in combination. It has been designed for computational efficiency "particularly regarding spatial-encoding and decoding schemes, optimized for real-time production and delivery of immersive web content" [28, p. 1]. It works for Ambisonics of various specifications:

- ACN channel ordering and N3D (as default) or SN3D normalization
- FuMa format (up to 3rd-order)
- FOA B-format signals in some cases (see [28])

For some examples of the library's modules at work see [29]. For a (perhaps partisan) appraisal of this library over Google's Omnitone see [30].

Limitations (-) and affordances (+)

- Not actively maintained

+ Overall an advanced and comprehensive suite of components

+ Customizable features (e.g. binDecoder: implements ambisonic to binaural decoding, using user-defined HRTF-based filters)

+ Provides many audio classes which provide helpful ready-to-use features (such as conversion between different Ambisonic ordering conventions)

+ Provides ability to rotate the sound scene of an ambisonic stream, with real-time control of yaw, pitch, and roll rotation angles

+ Well documented despite lack of active maintenance

B. Omnitone

Google's Omnitone library is another JavaScript implementation based on WAA, for multichannel streaming and binaural rendering of ambisonic signals <= 3OA. It does this through the AudioBufferSourceNode. As with JSAmbisonics, head position (with additional implementation) can be translated via the rotation matrix to create head-tracking. Binaural rendering is enabled through WAA's ConvolverNode and GainNodeinterfaces. Omnitone uses SADIE HRTFs [31] though custom HRTFs can be loaded. Omnitone's goal is to be a transparent framework for binaural rendering. Additional effects are possible due to the extensible nature of Omnitone, but only through further development. An example of this can be seen in OpenAirLib [32] which is a modified version of Omnitone. An implementation of 2OA using Omnitone is available at [33]. A primer is available at [34].

Although there is a forum [35] which is active, it is not populated with many issues, so resolving issues may require more proactivity on the part of the practitioner. Not all practitioners are comfortable interacting with developers in such environments, but are encouraged to do so by the authors. Feature requests, raising issues, etc. all contribute to a repository of information, which is helpful to future practitioners.

Limitations (-) and affordances (+)

- Forum only nominally active

+ Codec-agnostic. Providing that the audio stream is available via WAA(e.g. BufferSource or MediaStreamSound) it performs the spatialization as expected

+ Optimized performance as Google is heavily involved with web specifications, including WAA

+ Good suite of components alongside documentation

C. HOAST

HOAST (Higher-Order Ambisonics STreaming) is a browser-based multimedia platform for HOA audio and 360º videos. It is based on JSAmbisonics but provides some useful modifications. It uses OPUS files in a WebM container, and streams using MPEG-DASH. This stream is received by the client via dash.js and sent to the HTML5 audio element. It is then decoded and included in the WAA audio context using a MediaElementAudioSourceNode. The audio part of the stream (if the file is a 360º video) is then sent to nodes (based on the JSAmbisonics nodes) for processing - namely rotation and acoustic zoom. Finally, the binaural decoding filters are convolved with the ambisonics stream using the WAA convolver nodes and sent to the client audio hardware via the AudioDestinationNode. Internally, the video is rendered using the JavaScript 3D library (three.js) which itself uses a WebGL renderer. The acoustic zoom effect allows the user to focus on far-field sources in a particular direction (the view direction), essentially 'pulling' the field toward the listener from this direction. Consequently, it emphasizes sound sources in that direction, and effectively attenuates sources outside that direction.

The OPUS codec (a versatile audio codec for interactive speech and music transmission over the internet, also intended for storage and streaming applications) was chosen as it supports the highest number of channels per file (for browsers tested by the developers, see [36] with the exception of Safari, which can be difficult to develop for). Despite it being a lossy codec, research [37] has shown any degradation to be negligible in 3OA for a bit rate of 32 kbit/s per channel or higher. The balance between quality and performance is heightened for browser-based applications.

Limitations (-) and affordances (+)

- iOS devices and some browsers do not support OPUS

+ Simultaneous audio and video rendering has been given particular attention (this is expensive in general, so codecs, and decoding methods have been selected with this in mind)

+ State-of-the-art dynamic binaural decoding based on FOV (MagLS filters used, see [38])

+ Adaptive bitrate streaming for video

+ Acoustic zoom/ dominance effect (in 3DoF media)

+ Audio content can be accompanied by 360º video or dual-mono video content, for playback in the browser, where the FOV is controlled via the mouse, or via head-mounted displays (HMDs) via a custom video.js plug-in using WebXR API (WebXR needs to be enabled explicitly in Chromium-based browsers)

+ Support of further 360º and HMD formats in videojs-xr is planned for the future

+ Option to use the IEM servers to host your 360º HOA video for playback

+ Documented case of live streaming 3OA (during Audio Mostly 2020 in Graz)

D. Earshot

Envelop Ambisonic RTMP Streaming Higher-Order Transcoder (Earshot) forms part of a series of tools created by Envelop for creating and streaming multi-channel immersive 3D sound experiences. These include offline spatial audio production tools 'Envelop for Live' (E4L) (for Ableton Live Suite). Earshot can be used to transcode higher-order Ambisonics and other multichannel live streams for the web. Earshot is mostly aimed at developers creating custom applications for livestreaming ambisonics. Earshot integrates with E4L in such a way that one can for example work in Ableton Live with the E4L to do panning and then, by using Loopback [39] or a digital or physical alternative, stream it with Earshot (see Fig. 6).

Earshot allows one to stream HOA up to 3OA (16 channels) through the browser. Earshot is a containerized multichannel Real-Time Messaging Protocol (RTMP) -->

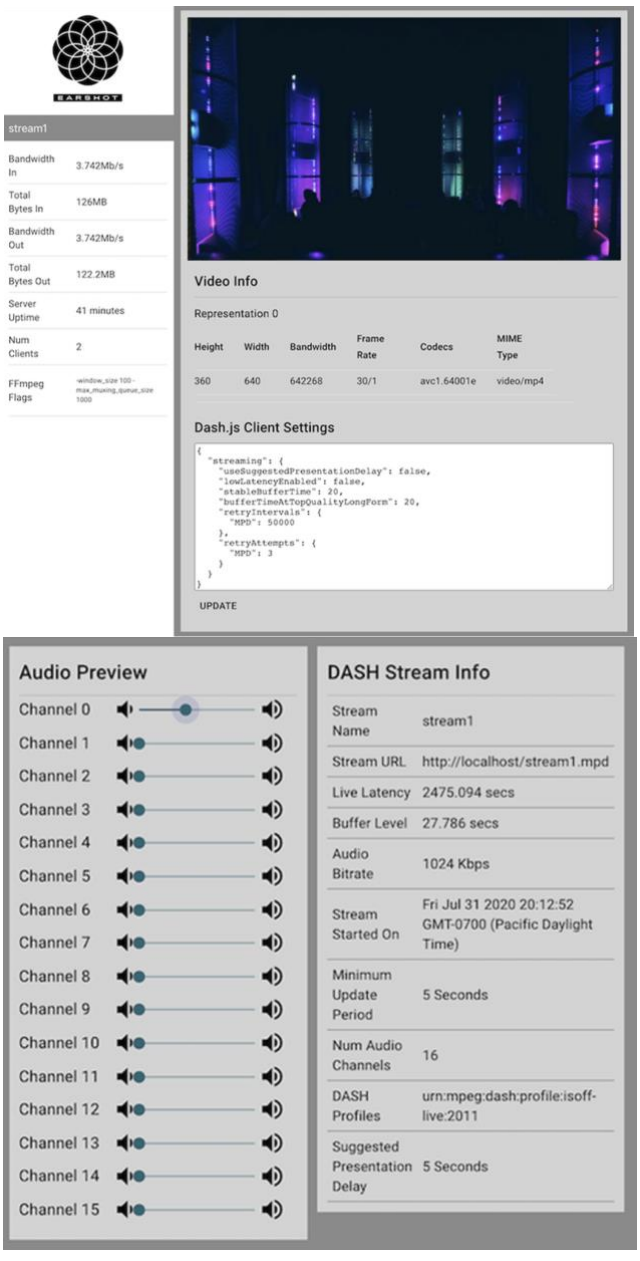

Fig. 6. Earshot User Interface

Dynamic Adaptive Streaming over HTTP (DASH) or in short RTMP --> DASH transcoder, based on Nginx' [40]. Nginx is open-source software for web serving, caching, load balancing, media streaming, and more [41].

Earshot can be used with OBS Studio Music Edition [42] an open-source software for video recording and live streaming. It supports multichannel AAC encoding up to 16 channels (OPUS provides support for up to 255 channels, however OBS only supports up to 16 channels on the encoding side – therefore limiting the stream to 3OA). In any case, the limit for <= 3OA web streaming (due to WAA) persists.

Earshot is based on Docker (a tool to create, deploy, and run applications by using containers, that allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package); FFMPEG (a command line toolbox to manipulate, convert and stream multimedia content), MPEG-DASH (an adaptive bitrate streaming technique that enables high quality streaming of media content over the internet delivered from conventional HTTP web servers) and OPUS [43]. Earshot is designed to be easily deployed to a cloud-based hosting solution, such as AWS ECS or DigitalOcean.

Earshot is mostly aimed at developers creating custom applications for livestreaming ambisonics. Earshot integrates with E4L in such a way that one can for example work in Ableton Live with the E4L to do panning and then, by using Loopback [39] or a digital or physical alternative, stream it with Earshot (see Fig. 6).

Limitations (-) and affordances (+)

- iOS devices and some browsers do not support OPUS

+ Well documented with and active community of users

+ Relatively easy to use (requires some knowledge of the syntax of networking)

+ Codec supports live stream up to 255 audio channels (i.e. up to 14OA) with optional video (web stream constrained to 4OA due to WAA channel count limitation)

+ Web interface for stream monitoring and debugging

+ Documentation clearly written and comprehensive

+ The Ambisonics stream produced on the server side using Earshot is stable

E. Acoustic Atlas

Acoustic Atlas is a real-time auralization application connected to a growing archive of room impulse responses (RIRs) from natural and cultural heritage sites. These sites are displayed on an interactive globe map user interface. Similar to browsing Google Earth, a user can search via the globe interface or also via text, for locations in the database. It functions on any smart mobile device or computer running a web browser. The built-in microphone (or selected microphone in the device soundcard settings) and headphone output of the device is utilized to transport the user to the selected heritage site via headphones and live microphone feed (which has its own UI, see Fig. 7).

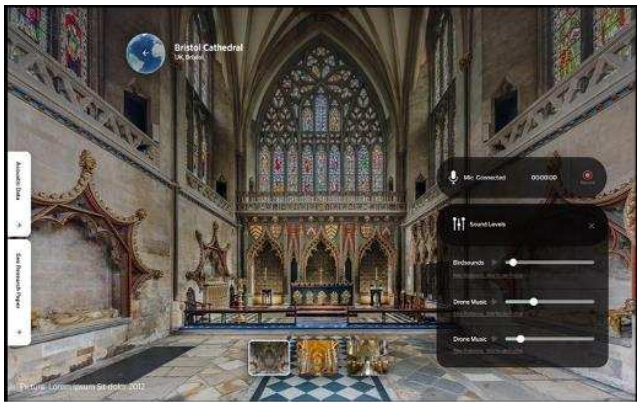

Fig. 7. UI inside a location in Acoustic Atlas

The system utilizes a document-oriented database that contains information such as geospatial position, room impulse response filenames, ambient field recording filenames, photography, archaeological text, acoustic text and analysis, technology used, and more. These are all loaded when clicking on the globe location.

The aims are to conserve acoustic heritage by creating audible interactive experiences of said spaces, with an equal mixture of acoustic scientific methods as well as creative artistic process.

The project aims to curate specialist RIR content as well as immersive field recordings and compositions. Such data is managed via user accounts directly embedded into the database, meaning said scientists and artists can manage their own content. The general public may record themselves and save their song or sound to their own devices. For legal reasons the Acoustic Atlas do not keep such files and the audio buffer clears on the client's device the moment the site is refreshed, or a new location is loaded. Any researcher or artist doing relevant work should get in touch to add their work to the archive [44].

The auralization process is enabled by WAA via connecting a number of node objects to create the desired signal chain. Tone.js [45] is a web audio framework, used as a wrapper around WAA for various parts of the audio signal flow and in particular for the convolution process. Tone.js (similar to WAA) allows for sample-accurate synchronization and scheduling of parameters. The simplified auralization signal flow of audio is as follows:

MIC_INPUT --> GAIN --> EQ --> CONVOLVER --> MASTER (see Fig. 8)

As a bridge, Tone.Context is used as AudioContext. Tone.Convolver, Tone.Filter, Tone.EQ3, Tone.Meter, Tone.Gain, Tone.UserMedia, Tone.Master enables the signal flow for the complete auralizations process. Additionally, the architecture includes Cesium.js for the globe interface.

Limitations (-) and affordances (+)

- The project is in early development and still needs to undergo robust user testing

- Documentation is largely unavailable at present

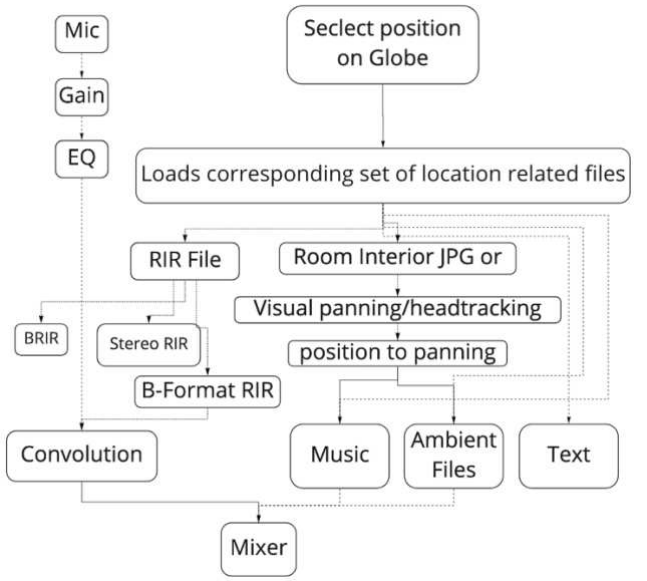

Fig. 8. Signal flow of UI & audio processing in Acoustic Atlas

- A plugin version that musicians can use as a VST or Max4Live plugin in musical DAWs would make more sense in the context of music production, the current WAA version has limited functionality in this respect

+ Supported on most devices

+ Easy to get started with and use

+ Great variety of potential future creative variations of this project and such an archive

+ Practitioners can refer to a URL of a selected location with its respective content, to enrich their work with real-time auralizations of said location

+ Version 2.0 (anticipated summer 2021) will support 360o headtracking/browser movement mapped to B-format RIR convolution reverb, thus adding to mobile and VR formats

+ Useful for heritage acoustics and sound ecology contexts

+ Useful for sound art and music projects dealing with acoustics and specific sites

V. WebRTC

WebRTC ('web real-time communication') is an open-source collection of JavaScript APIs affording direct, native connection between browsers and web-enabled devices. It provides high quality, low latency and low-cost video, audio and data communications, implemented as open standards for real-time, plugin-free applications. As such, it's particularly useful for computationally demanding experiences (for example, social VR). It can be integrated with WAA, which affords utilization of WAA's modular paradigm for audio processing and effects. It was developed by Google and is available on most browsers and native clients for major platforms. Its implementations vary across browsers, as we would expect.

"The guiding principles of the WebRTC project are that its APIs should be open source, free, standardized, built into web browsers, and more efficient than existing technologies" [46].

A. High Fidelity's Spatial Audio API

High Fidelity's Spatial Audio API is included here, though proprietary, as it does something unique relative to other tools. High Fidelity set out to have high quality audio with low 'mouth to ear' latency, available for live communication in online environments, with multiple sound sources and multiple listeners. This is no small undertaking. To achieve this, they use their own cloud servers to mix in real-time and send a single stereo stream to each listener in the online environment. Hundreds of listeners can occupy the same online environment, without compromising sound quality and intelligibility. In fact, using peer networked servers they have tested having 5,000 listeners within earshot of one another in a single environment, simultaneously. Each with their own real-time rendered HRTF, for binaural synthesis of a dynamic, high quality environment. It worked. In order for it to work, they needed to take a distinct design direction:

Use of WebRTC over WAA to overcome the > 100ms latency of WAA

Audio is processed in the cloud with no cost on the client side (making it mobile-friendly)

Use of full HRTF for each of the sources for each listener

The results can be experienced in online demos (see [47]). They achieved these by making their HRTF model fast (using a small audio framing window), paying attention to noise gating and peak limiting methods, and other mixing techniques which are not commonly employed by technologies using WebRTC (video conferencing software) where latency is prioritized, but spatialization is absent and mixing is largely ignored, save for (sometimes aggressive) ducking.

Limitations (-) and affordances (+)

- Paid-for product

+ High quality sound

+ Low latency makes it great for real-time applications

+ Low acoustic stress for listeners relative to the number of sound sources (sources do not compete for attention due to high-quality spatialization)

+ Accommodates a high number of sources and listeners in one space, simultaneously

+ Updated documentation, guides and examples [47] make it relatively easy to implement

VI. Further Tools

A. Google's Resonance Audio SDK for the web

Resonance SDK is a real-time JavaScript SDK (or suite of libraries) for encoding Ambisonics dynamically for the

browser [48]. It uses Google's Omnitone (and can export for this) and extends it to include customizable parameters (source directivity and spread, near-field effects) as well as simple room-model features (surfaces and dimensions as well as occlusions).

B. bogJS

bogJS [49] is a JS framework for object-based browser-rendered audio, which uses WAA's PannerNode (to be extended by SpatialPannerNode once this is implemented in the major browsers). Object based audio affords personalized, scalable, immersive audio through the browser. For an introduction to object-based audio see [50].

C. Invisio

Invisio allows expert and non-expert practitioners to design platform-agnostic complex sonic experiences through the browser, with dynamic 3D sound components [51].

VII. Concluding remarks

All but one of the tools outlined in this paper are free to use and/ or open source. This is a tremendous credit to the work being undertaken by individuals and organizations worldwide. High Fidelity's Spatial Audio API, the proprietary tool, undertakes all of its processing on the company's cloud servers, at cost to them. The investment in this infrastructure is evident in the quality of their tool. Some general notes about developing for the web follow:

- Keep file codecs and sizes as manageable as possible, given a project's demands
- Develop for a primary browser (often Chrome, as Google is heavily involved in developing web standards) and primary hardware configuration. It's just not possible to consider all combinations of hardware and software
- Web development is a specialism which should not be underestimated, particularly on the server-side (e.g. for live-streaming)
- Consider how powerful your server needs to be (for example, how many people will access it simultaneously) as this impacts project feasibility
- Invest in a good test setup to simulate live/ actual use

Acknowledgment

The authors would like to extend special thanks for input provided by the following contributors: Hongchan Choi (Chrome/ Omnitone/ W3C), Thomas Deppisch and Nils Meyer-Kahlen (HOAST), Roddy Lindsay (Envelop/ Earshot), Lorenzo Picinali (Pluggy) and Josh Reiss (Nemisindo). Cobi van Tonder thanks the European Union's Horizon 2020 research and innovation programme for support under the Marie Sklodowska-Curie grant agreement No 897905.

References

[1] "Web Audio API." [Online]. Available: https://www.w3.org/TR/webaudio/#Spatialization. [Accessed: 18-May-2021].

[2] A. Deveria and L. Schoors, "Can I use... Support tables for HTML5, CSS3, etc," 2021. [Online]. Available: https://caniuse.com/?search=web audio api. [Accessed: 27-May-2021].

[3] J. (ed. . Berkovitz and O. (ed. . Thereaux, "Web Audio Processing: Use Cases and Requirements," 2013. [Online]. Available: https://www.w3.org/TR/webaudio-usecases/. [Accessed: 27-May-2021].

[4] MDN contributors, "Basic concepts behind Web Audio API - Web APIs | MDN," 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Basic_concepts_behind_Web_Audio_API. [Accessed: 14-Jun-2021].

[5] Tom Söderlund, Marcus Krüger, Michael Stenmark, Oskar Åsbrink, and Robert Nyman, "Songs of Diridum: Pushing the Web Audio API to Its Limits - Mozilla Hacks - the Web developer blog," 2013. [Online]. Available: https://hacks.mozilla.org/2013/10/songs-of-diridum-pushing-the-web-audio-api-to-its-limits/. [Accessed: 20-May-2021].

[6] C. Pike, P. Taylour, and F. Melchior, "Delivering Object-Based 3D Audio Using The Web Audio API And The Audio Definition Model," in 1st Web Audio Conference (WAC-2015) (2015) 2-6, 2010.

[7] "The Chromium Projects." [Online]. Available: https://www.chromium.org/. [Accessed: 27-May-2021].

[8] "LISTEN HRTF DATABASE." [Online]. Available: http://recherche.ircam.fr/equipes/salles/listen/. [Accessed: 19-May-2021].

[9] T. Carpentier, "Binaural synthesis with the Web Audio API," in Proc. of 1st Web Audio Conference, 2015, pp. 0–7.

[10] P. Stitt, E. Hendrickx, J.-C. Messonnier, and B. Fg Katz, "The Role of Head Tracking in Binaural Rendering," in 29th Tonmeistertagung - VDT International Convention, 2016, pp. 1–5.

[11] "SOFA (Spatially Oriented Format for Acoustics) - Sofaconventions," 2021. [Online]. Available: https://www.sofaconventions.org/mediawiki/index.php/SOFA_(Spatially_Oriented_Format_for_Acoustics). [Accessed: 14-Jun-2021].

[12] "Web Audio Playground." [Online]. Available: https://webaudioplayground.appspot.com/ [Accessed: 07-Jun-2021]

[13] J. R. R. Lee, J. D. Reiss, J. Ryan, R. Lee, and J. D. Reiss, "'Real-Time Sound Synthesis of Audience Applause' Real-Time Sound Synthesis of Audience Applause," J. Audio Eng. Soc, vol. 68, no. 4, pp. 261–272, 2020.

[14] R. Selfridge et al., "Creating Real-Time Aeroacoustic Sound Effects Using Physically Informed Models," J. Audio Eng. Soc, vol. 66, no. 8, pp. 594–607, 2018.

[15] J. Zúñiga and J. D. Reiss, "Realistic Procedural Sound Synthesis of Bird Song Using Particle Swarm Optimization," in 147th Audio Engineering Society International Convention, 2019.

[16] J. Reiss, H. Tez, R. S.-A. E. S. C. 150, and U. 2021, "A comparative perceptual evaluation of thunder synthesis techniques," in 150th Audio Engineering Society Convention, 2021.

[17] "Nemisindo References - Google Docs." [Online]. Available: https://docs.google.com/document/d/1qGFCtd8drv0GBETCAptXJ2EbIkJtdwZ2gQ5rhUsIGc0. [Accessed: 10-Jun-2021].

[18] A. Farnell, Designing sound. Cambridge, Massachusetts: Mit Press, 2010.

[19] D. Rocchesso and F. Fontana, The sounding object. Firenze, Italy: Phasar Srl, 2003.

[20] P. R. Cook, Real Sound Synthesis for Interactive Applications, First. New York: A K Peters/ CRC Press, 2002.

[21] P. Bahadoran, A. Benito, T. Vassallo, and J. D. Reiss, "FXive: A Web Platform for Procedural Sound Synthesis," in 144th Audio

Engineering Society Convention, 2018.

[22] N. Jillings, Y. Wang, J. Reiss, and R. Stables, “Convention e-Brief 301 JSAP: A Plugin Standard for the Web Audio API with Intelligent Functionality,” in 141st Audio Engineering Society Convention, 2016.

[23] M. Comunità, A. Gerino, V. Lim, and L. Picinali, “Design and evaluation of a web-and mobile-based binaural audio platform for cultural heritage,” Appl. Sci., vol. 11, no. 4, pp. 1–22, Feb. 2021.

[24] M. Cuevas-Rodríguez et al., “3D tune-in toolkit: An open-source library for real-time binaural spatialisation,” PLoS One, vol. 14, no. 3, p. e0211899, Mar. 2019.

[25] Institute of Electronic Music and Acoustics, “IEM Plug-in Suite,” 2021. [Online]. Available: https://plugins.iem.at/. [Accessed: 15-Jul-2021].

[26] “SPARTA - Spatial Audio Real-Time Applications,” 2021. [Online]. Available: http://research.spa.aalto.fi/projects/sparta_vsts/. [Accessed: 15-Jul-2021].

[27] “Spat | Ircam Forum,” 2020. [Online]. Available: https://forum.ircam.fr/projects/detail/spat/. [Accessed: 15-Jul-2021].

[28] A. Politis and D. Poirier-Quinot, “JSAmbisonics: A Web Audio library for interactive spatial sound processing on the web,” Interactive Audio Systems Symposium, 2016.

[29] D. Poirier-Quinot and R. Vincent, “WebAudioAPI First and Higher Order Ambisonic Examples,” 2017. [Online]. Available: https://cdn.rawgit.com/polarch/JSAmbisonics/e28e15b384f2442a66fadc0035439c64ed65fa4d/index.html. [Accessed: 20-May-2021].

[30] J. Werle and A. Politis, “JSAmbisonics vs Omnitone · Issue #8 · polarch/JSAmbisonics · GitHub,” 2016. [Online]. Available: https://github.com/polarch/JSAmbisonics/issues/8. [Accessed: 20-May-2021].

[31] “SADIE | Spatial Audio For Domestic Interactive Entertainment.” [Online]. Available: https://www.york.ac.uk/sadie-project/GoogleVRSADIE.html. [Accessed: 27-May-2021].

[32] K. I. Brown, M. D. J. Paradis, and D. T. Murphy, “OpenAirLib - Research Database, The University of York,” 2017. [Online]. Available: https://pure.york.ac.uk/portal/en/publications/openairlib(60379d65-11fc-4478-8125-2406ea2b66c0).html. [Accessed: 19-May-2021].

[33] “ASSEMBLY 2020,” 2020. [Online]. Available: https://www.assembly2020.co/. [Accessed: 27-May-2021].

[34] “OMNITONE.” [Online]. Available: https://googlechrome.github.io/omnitone/#home. [Accessed: 21-May-2021].

[35] “Issues · GoogleChrome/omnitone · GitHub.” [Online]. Available: https://github.com/GoogleChrome/omnitone/issues. [Accessed: 27-May-2021].

[36] T. Deppisch, N. Meyer-Kahlen, B. Hofer, T. Latka, and T. Zernicki, “HOAST: A Higher-Order Ambisonics Streaming Platform,” in AES Engineering Brief, 2020.

[37] M. Narbutt, S. O’Leary, A. Allen, J. Skoglund, and A. Hines, “Streaming VR for immersion: Quality aspects of compressed spatial audio,” in Proceedings of the 2017 23rd International Conference on Virtual Systems and Multimedia, VSMM, 2017, pp. 1–6.

[38] C. Schörkhuber, M. Zaunschirm, and R. Höldrich, “Binaural Rendering of Ambisonic Signals via Magnitude Least Squares,” in Proceedings of the DAGA, vol. 44, 2018, pp. 339–342.

[39] “Loopback.” [Online]. Available: https://rogueamoeba.com/loopback/. [Accessed: 09-Jun-2021].

[40] Envelop, “Earshot,” https://github.com/EnvelopSound/Earshot. [Online]. Available: https://github.com/EnvelopSound/Earshot. [Accessed: 07-Jun-2021].

[41] “NGINX,” https://www.nginx.com/resources/glossary/nginx/. [Online]. Available: https://www.nginx.com/resources/glossary/nginx/. [Accessed: 08-Jun-2021].

[42] OBS Project, “Open Broadcasting Software (OBS),” https://obsproject.com. [Online]. Available: https://obsproject.com/wiki/. [Accessed: 07-Jun-2021].

[43] “Opus Codec Documentation,” https://opus-codec.org/docs/, 2017. [Online]. Available: https://opus-codec.org/docs/. [Accessed: 08-Jun-2021].

[44] C. van Tonder, “Acoustic Atlas,” https://acousticatlas.info, 2019. [Online]. Available: https://acousticatlas.info. [Accessed: 07-Jun-2021].

[45] Y. Mann, “Tone.js.” [Online]. Available: https://tonejs.github.io. [Accessed: 07-Jun-2021].

[46] S. Dutton, “Get Started with WebRTC - HTML5 Rocks,” 2012. [Online]. Available: https://www.html5rocks.com/en/tutorials/webrtc/basics/. [Accessed: 14-Jun-2021].

[47] “Real-time Spatial Audio API for Group Voice Chat | High Fidelity.” [Online]. Available: https://www.highfidelity.com/. [Accessed: 20-May-2021].

[48] “Resonance Audio - Resonance Audio SDK for Web.” [Online]. Available: https://resonance-audio.github.io/resonance-audio/develop/web/getting-started.html. [Accessed: 10-Jun-2021].

[49] M. Weitnauer and M. Meier, “GitHub - IRT-Open-Source/bogJS: JavaScript framework for object-based audio rendering in modern browsers from IRT,” 2016. [Online]. Available: https://github.com/IRT-Open-Source/bogJS. [Accessed: 10-Jun-2021].

[50] C. (BBC), Baume, A. M. (BBC), P. B. (BBC), M. L. (BBC), M. F. (BBC), and M. V. (MAGIX), “Object-based broadcasting – for European leadership in next generation audio experiences D3 . 4 : Implementation and documentation of a live object-based production environment,” 2016.

[51] A. Çamcı, K. Lee, C. J. Roberts, and A. G. Forbes, “INVISO: A Cross-platform User Interface for Creating Virtual Sonic Environments,” Proc. 30th Annu. ACM Symp. User Interface Softw. Technol. (pp. 507-518), 2017.