How to Spatial Audio with the WebXR API: a comparison of the tools and techniques for creating immersive sonic experiences on the browser

Matteo Tomasetti
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
matteo.tomasetti@unitn.it

Alberto Boem
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
alberto.boem@unitn.it

Luca Turchet
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
luca.turchet@unitn.it

2023 Immersive and 3D Audio: from Architecture to Automotive (I3DA) | 979-8-3503-1104-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/I3DA57090.2023.10289525

***Abstract*—The WebXR Device API provides a powerful set of functionalities that can be used for creating immersive experiences that can be accessed directly from a web browser. However, while creating a compelling visual experience is easy to implement using the WebXR API and related tools, creating the audio part of the experience is less clear and with more open questions and possibilities. This paper presents a comparison of six tools oriented to practitioners and used for implementing and delivering interactive spatial audio experiences in XR applications running on a browser. These tools are selected and evaluated based on their compatibility with the WebXR Device API, ease of use, provided support and documentation, and specific features like reverberation, head-related transfer functions (HRTFs), sound scene rotation, and real-time microphone input. The paper then discusses the integration of these spatial audio tools into WebXR applications, highlighting their potential and limitations. It also addresses the absence of standardized spatial audio frameworks for the web. The challenges of choosing the right tool regarding resource allocation, learning curve, and compatibility with XR systems are acknowledged. This overview aims to provide insights into the current state of spatial audio tools for the web, making them more accessible to developers, musicians, and researchers seeking guidance on selecting suitable 3D spatial audio tools for experiences based on WebXR.**

***Index Terms*—WebXR, Spatial audio, Web audio, Binaural audio, Comparison, Browser, HCI, Internet of Sounds.**

I. Introduction

Over the past decades, technologies for Extended Realities (XR) have grown exponentially as devices such as Head-Mounted Displays (HMDs) and tracked controllers have become increasingly accessible to the general public. In addition, the availability of web browsers in most commercial XR systems (e.g., Meta Quest, Pico Neo, Magic Leap) creates new avenues for shared and more accessible interactive experiences.

The WebXR Device API [1] emerged as a suitable candidate for creating cross-platform virtual experiences, which are easy to deploy and, most importantly, not dependent on specific hardware or operating systems. While the predominant focus has been on enhancing the visual aspect of 3D web applications (e.g., leveraging WebGL™ [2]), it is essential to recognize that audio on the web also plays a vital role in engendering a profound sense of immersion and presence [3]. Thanks to a growing interest in multi-user and social XR applications, sound becomes an essential mode of communication [4] and expression [5]. Moreover, as highlighted by the emerging field of the Internet Of Sound, we are witnessing the emergence of devices capable of communicating sound-related information [6]. Hence, spatial audio becomes a critical component for constructing truly immersive applications [7], [8], [9].

This paper explores the existing range of tools that can be utilized to incorporate interactive spatial audio into immersive applications on web browsers, utilizing the WebXR API. We provide a comprehensive overview of six tools dedicated to spatial audio on the web, evaluating their ease of use, support, and documentation. Additionally, we delve into essential features specific to spatial audio systems, including reverberation, support for individualized head-related transfer functions, sound scene rotation, and real-time microphone input, while scrutinizing their implementation within the examined tools.

Furthermore, this paper delves into the utilization and integration of 3D audio tools within applications based on the WebXR API, exploring the possibilities and limitations they present. While numerous frameworks harness WebXR capabilities for crafting interactive 3D visual content, a standardized approach to spatial audio still needs to be developed. Existing tools are predominantly based on either the Web Audio API (WAA) [10] or third-party and proprietary systems, exhibiting variances in features and functionality.

For instance, Ambisonics has little support; only some tools include reverberation or compatibility with head-related transfer functions (HRTFs). Consequently, selecting the appropriate tool becomes challenging, necessitating careful consideration of resource allocation, learning curve, and compatibility with WebXR-based systems.

By providing an encompassing overview, this study aims to offer valuable insights into the current state of the art of 3D audio tools, empowering developers, musicians, and

researchers who may face uncertainty when deciding which spatial audio tools to employ in the context of web-based XR experiences.

II. Background

A. WebXR Device API

WebXR is a Device API developed and maintained by the Immersive Web Community Group and Immersive Web Working Group of the W3C [11] that enables developers to create immersive Virtual Reality (VR) and Augmented Reality (AR) experiences directly within web browsers. For developers, the WebXR Device API simplifies the development process of VR/AR applications by offering a unified set of functionalities that handle device compatibility, input controls, 6 Degrees Of Freedom (6DoF) tracking, and rendering of a 3D scene. Users can visit a website to engage with an immersive experience without downloading or installing any applications on a device. Another essential aspect of WebXR is interoperability, allowing different XR devices and browsers to work seamlessly and deliver a consistent user experience. From this point of view, WebXR API shares some similarities with the open OpenXR™ standard by the Khronos Group [12].

Despite these features, WebXR API is not supported by every browser. Table I summarizes the main browsers and devices that support WebXR. An important distinction must be made between the VR and the AR components of the API. The VR side is supported by browsers in devices such as the Meta Quest and Pico series of headsets and Android phones through Google Cardboard and XR-dedicated browsers such as Wolvic [13]. The AR has some more limitations. While basic functionalities are available on the Magic Leap browser or the Hololens 2, more advanced ones (such as depth sensing and light estimation) are available only on ARCore-supported Android devices [14]. Another aspect to consider is the availability of input and interaction techniques supported. For instance, hand input is not available in all browsers and hardware. Furthermore, it is worth noting that (as of the time of writing) a prominent limitation of existing systems is the inability to directly experience WebXR applications on the Safari web browser[1].

One prominent aspect of the WebXR Device API is its seamless integration with several web-based frameworks used for developing interactive experiences. Frameworks like Three.js [15], A-Frame [16], and Babylon.js [17] provide direct access to the WebXR functionalities, facilitating the creation of immersive web experiences with ease.

B. Binaural audio

Binaural audio is widely embraced as the primary choice for spatial audio due to the widespread availability of the necessary technology (headphones) for its reproduction [18]. Studies in psychoacoustics have revealed that the human auditory system employs different mechanisms to localize sounds [19]. Regarding sounds positioned in the horizontal plane, the angular direction is primarily identified using Interaural Time Differences and Interaural Level Differences. On the other hand, sound elevation relies primarily on spectral cues, which depend on the spatial position of a sound source relative to the listener's head. These spectral cues are generated using HRTFs, which encode the necessary acoustic information for sound localization in headphones and play a vital role in auditory perception [20].

HRTFs capture the modifications in the sound spectrum upon entering the ear canal, influenced by sound diffraction and reflection due to individuals' unique physical characteristics [21]. By conducting acoustic measurements, individuals can acquire individualized HRTFs, which can then be grouped in a database [22]. Nevertheless, the expensive nature of the measurement process, the lack of portability in 3D loudspeaker systems, and computational difficulties pose challenges for individualized binaural systems. As a result, nowadays, the binaural systems available often depend on generic HRTFs, leading to diminished accuracy in sound localization and a greater likelihood of errors [23]. In addition to localization errors associated with HRTFs, binaural systems also encounter challenges related to front-back confusion sound localization and externalization. The latter refers to the perception of sound sources as originating from outside the listener's head [24]. To enhance localization accuracy, one workaround involves employing external head tracking devices that dynamically adjust binaural systems in response to the user's head movements [25].

Higher Order Ambisonics (HOA) has become a prevalent technique contemporary musicians use to engage with spatial audio, encompassing multichannel loudspeakers array and binaural setups [26]. Ambisonics is a sound reproduction technique that facilitates generating a fully 3D virtual acoustic environment, incorporating numerous dynamic sound sources by employing a specific set of playback channels. A comprehensive review of Ambisonics can be accessed in the publication by Zotter and Frank [27]. In the initial implementation of Ambisonics by Gerzon [28], the technique utilized solely the 0th and 1st-order directional patterns, also called spherical harmonics. This configuration, known as B-Format, consisted of the omnidirectional component (W) and three dipole components (X, Y, Z). Nonetheless, the spatial resolution of the 1st-order in Ambisonics is limited, which confines precise sound field reconstruction to a limited listening area. HOA addresses the issue by incorporating spherical harmonic decomposition of the sound field at higher orders to overcome this constraint. This extension expands the reproduction area, albeit at the cost of significantly increasing the number of channels required [29].

Recording and reproducing 3D sound scenes involve well-established techniques. These methods typically employ a microphone array to capture the audio scene in three dimensions and subsequently map the recorded signals directly to the appropriate channels of the intended playback system, which could be stereo, headphones, or surround formats.

[1]However, the visionOS for the Apple Vision Pro HMD will include a version of Safari that supports WebXR.

However, these approaches often lack flexibility when it comes to reproducing recordings on different playback systems or accommodating variations in the user's head orientation, particularly in the case of binaural microphone array recordings [30].

Ambisonics can help reduce the aforementioned limitations thanks to its encoding and decoding processes while introducing more versatility [26]. To utilize Ambisonics, the audio signal originating from a particular sound source undergoes an initial transformation known as encoding. This step involves converting the signal to Ambisonics format. By employing the encoding process, choosing the Ambisonics order is possible, directly impacting factors such as the number of channels, spatial complexity of the reproduced sound field, and the computational load and memory requirements [31].

Commonly, encoding functions facilitate the conversion of the signal into intermediate spherical harmonics signals, enabling, in web architectures, precise placement of the sound signal within the 3D space using Cartesian coordinates. Reverberation can be added during the encoding process, as shown in [32]. However, reverberation can also be applied later in the process, depending on the use case [27]. Following this, dedicated functions within the spatial audio tool come into play to enable head tracking, leveraging the orientation data sent from HMDs. This process is called rotation of a sound scene, and is included in most tools for doing spatial audio for WebXR. Finally, the Ambisonics signal is decoded into binaural format through the process of binaural rendering.

While Ambisonics became widely adopted for 3DoF systems (such as the ones using 360° videos), it presents some limitations for 6DoF applications, where users can move in a three-dimensional space. For a detailed overview of such limitations and methods for overcoming them, see [33], [34].

III. Methods

A. Selection Criteria

We selected the spatial audio tools to compare based on the following criteria. First, we collected spatial audio tools suitable for web development by cross-referencing relevant papers in the field and keyword searches in Google Scholar and GitHub.

Second, we filtered out tools that are - at the time of this writing - not actively maintained and supported. This was done by history and upgrades reported on GitHub repositories, release history on official websites, as well as through personal correspondence with developers and companies' representatives. Third, since we are interested in immersive and 3D applications, we selected spatial audio tools that a) allow for binaural rendering; b) allow the sound source to be positioned in the 3D virtual space via a Vector representing three axis (e.g., Cartesian coordinates), which is the prevalent coordinate system used by WebXR frameworks such as A-Frame and Babylon.js; c) allow to simulate of the effect of proximity and distance of a sound with respect to the listener.

B. Evaluation Criteria

We identified five main criteria for evaluation. The first two pertain to the usage of a given tool. The others are related to four main characteristics identified for spatial audio systems in immersive applications.

*1) **Ease of Use**:* Developing immersive and interactive applications requires a set of knowledge and skills ranging from 3D modeling to interaction design and of course the use of audio and spatial audio in particular. However, not all practitioners are familiar with the theory behind binaural audio and how the different parameters and functionalities influence the final result. This very specialized domain of sound design can not be easily accessed, especially for beginners or developers with no previous knowledge of audio production. With "ease of use" we refer to the degree of accessibility of a tool from a very pragmatic point of view. We especially considered how a given tool can be easily integrated into frameworks supporting the WebXR API, such as Three.js, A-Frame, or Babylon.js.

*2) **Support and documentation**:* The second criterion for evaluation is closely related to the previous one. From the point of view of a developer, the "ease of use" of a tool is also influenced by the availability of clear documentation, which includes code examples and working demos, as well as active support available from the developers or a community of users (e.g., forums, blogs, Discord channels). Together with compatibility with programming languages and frameworks, the documentation and support provided are aspects that condition and orient the choice of spatial audio tools, especially for non-experts.

*3) **Real-time microphone input support**:* While spatial audio deals mostly with rendering an audio scene, the spread of multi-user immersive environments on the browser (e.g., Networked-Aframe [35]) makes supporting verbal communication among users increasingly important. In social virtual worlds, users talk with each other by speaking through their avatars' mouths. Speech is usually captured through microphones embedded in most HMDs and smartphones. Some 3D audio tools have the capability to incorporate microphones or line inputs directly, enabling the real-time spatialization of these audio sources within a 3D environment. In contrast, other tools do not support microphone input or do not provide any documentation about such a function. While this aspect is not crucial in binaural sound systems, we believe it helps practitioners better understand each tool, enabling them to make more informed choices concerning their specific project requirements.

*4) **Reverberation**:* The fourth criterion is reverberation, which plays a vital role in binaural audio systems. Reverberation, usually called room simulation, helps improve sound source localization, distance perception, and resolve externalization issues [36]. Therefore, it is crucial to understand whether a given tool supports reverberation, which types of reverberations it implements, and which control parameters. The most used techniques for reverberation are either based on convolution or algorithmic methods. The critical distinction between these two is that convolution-based techniques

TABLE I
SUMMARY OF THE MAIN FEATURES OF THE WEBXR API IN RELATION TO MOST DIFFUSED WEB BROWSERS AND THEIR CORRESPONDING XR DEVICES (ADAPTED FROM THE "SUPPORT TABLE FOR THE WEBXR DEVICE API" HTTPS://IMMERSIVEWEB.DEV).

	Web Browsers					
WebXR API Features	Google Chrome	Meta Quest Browser	Pico Browser	Microsoft Edge	Magic Leap Browser	Wolvic
WebXR Core	✓	Meta Quest 1, 2, Pro	Pico Neo 3, 4	Hololens 2	Magic Leap 2	Meta Quest 1, 2, Pro Pico Neo 3, 4
WebXR Gamepads Module	✓	Meta Quest 1, 2, Pro	Pico Neo 3, 4	Hololens 2	Magic Leap 2	Meta Quest 1, 2, Pro Pico Neo 3, 4
WebXR AR Module	Android w/ ARCore	Meta Quest 1, 2, Pro	X	Hololens 2	Magic Leap 2	X
Hit-test	Android w/ ARCore	X	X	Hololens 2	X	X
Hand tracking input	X	Meta Quest 1, 2, Pro	Pico Neo 4	Hololens 2	X	X
Anchors	Android w/ ARCore	Meta Quest 1, 2, Pro	X	Hololens 2	X	X
Depth sensing	Android w/ ARCore	X	X	X	X	X
Light estimation	Android w/ ARCore	X	X	X	X	X
Passthrough	X	Meta Quest 2 (b/w), Pro (color)	Pico Neo 4	X	X	X

involve convolving the sound signal with a multichannel impulse response (IR) recorded using specialized methods and microphone arrays from a real environment. In contrast, using algorithmic techniques simulates early reflections and the reverberant sound field.

*5) **Rotation of a sound scene**:* Our comparison focuses on spatial audio tools usable for VR/AR applications. It is important that users can locate exactly the different sound sources while moving their heads. The HMD tracks the position and rotation of the users' head and sends these information to the 3D audio tool, which in turn adapts the sound reproduction to maintain a constant relationship between the listener's position and the virtual position of the sound. Furthermore, this criterion is crucial for increasing realism and a sense of presence in immersive environments [37], [38].

*6) **Possibility of loading individualized HRTFs**:* The final criterion pertains to the tool's ability to load individualized HRTF profiles, which are fundamental aspects of binaural audio, as we described in Section II-B. Typically, HRTFs profiles are stored via the "Spatially Oriented Format for Acoustics" (SOFA) file [39], one of the most common data formats used for storing and sharing HRTFs [40]. Our comparison aims to determine which tools effectively support users in loading their own HRTFs or utilizing existing data sets and which tools either do not support this functionality or only provide partial support.

IV. SPATIAL AUDIO TOOLS FOR WEBXR

In this Section, we present the analysis of the six spatial audio tools we have selected, namely the WebAudioAPI and its Panner Node, jsAmbisonics, Google Resonance (Omnitone) for A-Frame, 3D Tune-In Toolkit JavaScript Wrapper, atmoky WebSDK, and Superpowered. The results of our comparison are summarized in Table II, where we present the different tools and the evaluation criteria for spatial audio.

A. Web Audio API (PannerNode)

The Web Audio API is a high-level JavaScript API that enables audio synthesis and processing directly in web browsers, from fundamental tasks such as equalization to advanced features like spatialization and real-time microphone input. The main strength of the WAA is the cross-platform nature and the support of most browsers, facilitating the development and delivery of interactive audio systems on the web [41]. The WAA is developed within the concept of nodes, fundamental building blocks representing different audio sources, processors, and destinations. They are used to create and manipulate audio signals within the audio graph. The latter is a visual representation of the audio processing flow, where nodes are connected to define the audio signal's path. Each node performs a specific function in the audio pipeline. Within the WAA, the node dedicated to audio spatialization is the PannerNode [42].

The PannerNode allows to place sounds in a 3D virtual space and control its sound spatialization parameters. This node uses two distinct panning techniques: the "equal-power panning law" and the "HRTF algorithm". While the former is the default mode of the PannerNode, in the context of our comparison, we consider only the latter, which employs binaural audio.

Ease of use: Thanks to its characteristics and widespread utilization, the WAA and its PannerNode have already been used to provide interactive audio in countless browser-based 3D applications using the WebXR API. Moreover, the PannerNode can be accessed directly through several frameworks used for developing audio on the browser, such as Howler.js [43] and Tone.js [44] (named Panner3D), as well as through WebXR-supported frameworks such as Babylon.js and Three.js. These audio frameworks are built upon the WAA, and are commonly used by developers of either sound on the web or 3D applications, making the PannerNode a tool very easy to integrate. Moreover, the PannerNode and WAA are distributed with the MIT Licence.

Support and documentation: Similar to other components of the WAA, PannerNode is accompanied by detailed online documentation. Its main reference page provides a clear guide and explanation of the different steps a developer has to follow to configure a spatial audio system. The theory behind

TABLE II
A SUMMARY OF THE COMPARISON RESULTS FOR THE AUDIO COMPONENTS OF EACH TOOL.

	Spatial Audio Tools for WebXR					
Audio Features	WAA PannerNode	GRAOAF	jsAmbisonics	3DTIT JSW	atmoky WebSDK	Superpowered
Real-time microphone input support	✓	✓	✓	X	✓	✓
Reverberation	✓	✓	✓	X	✓	✓
Rotation of a sound scene	✓	✓	✓	X	✓	✓
Possibility of loading individualized HRTFs	X	X	✓	✓	X	X

each parameter of PannerNode is also well-detailed. Moreover, thanks to a large and active community of users, several tutorials and examples can be sourced from websites, forums, and video tutorials.

Real-time microphone input support: Since PannerNode can spatialize incoming audio streams, it is possible to use the microphones embedded in headsets through the "MediaStreamAudioSourceNode" audio node interface of the WAA, and the consequent "getUserMedia()" method for capturing a sound signal from a microphone.

Reverberation: It is only possible through other WAA classes and methods, such as the "ConvolverNode", which operates on the input signal by convolving it with a maximum 4-channel IR that represents the acoustic characteristics of a given real environment. However, developers or users need to provide their own IRs files.

Rotation of a sound scene: Through the WAA, it is not possible to access the users' head orientation data. This feature must be implemented by using the orientation of the HMD acquired through the WebXR API and applied to the orientation of the audio listener positioned in the three-dimensional scene, which represents the user itself.

Possibility of loading individualized HRTFs: While the Panner Node's property "panningModel" allows the use of generalized HRTF (based on the described "IRCAM Listen HRTF Database" [45]), it is not possible to upload individualized HRTFs natively within the WAA.

B. *Google Resonance Audio (omnitone) for A-Frame*

Google Resonance Audio (omnitone) for A-Frame (GRAOAF) [46] is an open-source library based on the web SDK of Google's "Resonance Audio (Omnitone)" [47], a widely used spatial audio library available for several platforms such as Android, Wwise, and Unity. However, the official support from Google stopped in 2019. This "Resonance Audio (Omnitone)" wrapper is particularly interesting because it allows developers to manage spatial audio directly in A-Frame, one of the most popular frameworks used in developing immersive experiences on the web.

Ease of use: GRAOAF is provided as a component for A-Frame. Components are fundamental building blocks used in A-Frame. Developers can use custom JavaScript modules to implement and extend new functionalities. Therefore, GRAOAF can be easily integrated into WebXR applications developed with A-Frame, which is a very intuitive and robust framework. GRAOAF is subject to the MIT Licence.

Support and documentation: GRAOAF provides fairly detailed documentation on its GitHub repository, including demos and code examples.

Real-time microphone input support: GRAOAF leverages the "MediaStream" web interface [48] to capture the audio signal from an external microphone. This incoming signal can then be processed spatially within the virtual environment.

Reverberation: Within GRAOAF, it is possible to create algorithmic reverberation through the "resonance-audio-room" component, which can also control the Ambisonics order (up to the 3rd), and room width, height, and depth. Such parameters are all expressed in meters, which is particularly convenient since A-Frame (and most 3D frameworks for VR/AR) conventionally uses meters as measurement units. Through this library, it is also possible to specify different types of materials (e.g., grass, meta, marble, plywood-panel, fiber-glass-insulation, parquet-on-concrete) to be assigned to objects and elements that compose the virtual space.

Rotation of a sound scene: Since A-Frame gives direct access to WebXR API, it is then possible to update the orientation of the listener directly within the library.

Possibility of loading individualized HRTFs: To date, it is not possible for developers or users to load any files for individualized HRTFs.

C. *jsAmbisonics*

This library provides a set of JavaScript modules that extends the functions of the WAA to support first-order, and HOA since WAA does not natively support it [49]. It includes binaural decoding, the handling of rotation and direction of sound sources, and also supports the creation of new Ambisonics sound files [50].

Ease of use: The code base of jsAmbisonics is structured in a clear and intuitive way. Since this library is composed of a set of JavaScript modules, they can be easily integrated into the frameworks mentioned above, such as A-Frame or Babylon.js. However, to fully use this library, developers need a certain amount of understanding of the theory behind Ambisonics, its principles, and its components. The jsAmbisonics library is subject to the BSD 3-Clause License.

Support and documentation: The jsAmbisonics library is well-documented, and its GitHub page provides a step-by-step guide that explains how to integrate it within projects based on the Web Audio API.

Real-time microphone input support: to spatialize a real-time audio signal with jsAmbisonics, one must first acquire the audio input from the microphone, convert it to Ambisonics

format, and then process it with the spatialization classes provided by the library, such as the "monoEncoder".

Reverberation: For jsAmbisonics, reverberation is available as convolution through the "ConvolverNode" of the WAA.

Rotation of a sound scene: This library includes a "SceneRotator" class, which allows the rotation of the Ambisonics scene through yaw, pitch, and roll. Similarly to other tools, to use rotation data from an HMD, one must interface with WebXR and supported frameworks.

Possibility of loading individualized HRTFs: Users can load their individualized HRTFs via SOFA files, which must be first converted to JSON format. Such files can then be loaded through the "hrirLoader.load(url)" function developed within the "hrirLoader" of the "binDecoder" block, which is the component dedicated to performing binaural decoding.

D. *3D Tune-In Toolkit - JavaScript Wrapper*

This is a JavaScript port of the 3D Tune-In Toolkit (3DTIT) [51], an open-source C++ library designed for the development of immersive audio applications [52]. However, far from being complete, this JavaScript wrapper is only a partial port of the original 3DTIT project, done using Emscripten, an LLVM/Clang-based compiler that compiles C and C++ source code into WebAssembly primarily for execution in web browsers. Moreover, it is then possible for developers to contribute and extend functionalities of the 3DTIT JavaScript Wrapper through the "JsWrapperGlue.cpp" file by sending a pull request to the 3DTIT GitHub repository.

Ease of use: Even if it was never tested on immersive web applications, being this version of 3DTIT a JavaScript Wrapper (JSW), it is compatible with most frameworks that support WebXR API. 3D Tune-In Toolkit and the related JavaScript wrapper are distributed with a GPL-3.0 License.

Support and documentation: The documentation of the 3DTIT JSW is fairly detailed. It also provides code examples and a step-by-step guide for installation and use.

Real-time microphone input support: With the current version of this library, it is possible to spatialize incoming audio signals in real time. However, no specific functionalities are provided by the 3DTIT JSW for accessing and treating sound from microphones, and its system performances are unknown.

Reverberation: Unlike in the original 3DTIT release, reverberation has not yet been implemented in the JavaScript wrapper.

Rotation of a sound scene: We did not find any information in the documentation about this aspect, and there are no examples of how to implement it. While it might be possible to integrate 3DTIT JSW by interfacing with data from devices through the WebXR API, there is no information available on the performance of JSW in real-time with an external HMD.

Possibility of loading individualized HRTFs: This functionality is well supported by the 3DTIT JSW. HRTFs must be loaded through SOFA files or "3DTI-HRTF", a cross-platform portable binary format for handling HRTF data.

E. *Atmoky WebSDK*

The atmoky WebSDK [53] is an SDK explicitly designed for integrating spatial audio in 3D immersive environments. The core of this SDK is developed in WebAssembly and designed for seamless integration into the WAA.

Ease of use: The SDK can be easily integrated into WebXR-based applications since the atmoky WebSDK provides components that can be used directly in A-Frame, and in Three.js. The atmoky WebSDK is available with different types of licenses, from free to commercial.

Support and documentation: The website provides very detailed documentation that guides the developer step-by-step to set up and use the entire SDK.

Real-time microphone input support: It is possible to use the microphone input as the audio source for this WebSDK using the "source.setInput", which acts as the WAA's audio node interface.

Reverberation: The atmoky WebSDK uses an algorithmic reverberation. Only one reverberation parameter can be controlled: the "amount" (e.g., value = 0 means no reverberation, only dry signal; value = 100 means max possible reverberation, only wet signal). Moreover, from each sound source, one can control the "Reverb Send Level" (e.g., how much signal level of each sound source is sent to the reverb).

Rotation of a sound scene: This functionality can be implemented in two ways: if the WebSDK is used directly, it is only needed to set the listener's "setRotation" method. Otherwise, if the WebSDK's A-Frame integration is used, developers need to connect the "atmoky-camera-listener" component to the main camera in the scene, which is the user's HMD. Thus, any movement and rotation of the camera will be automatically transmitted to the virtual listener.

Possibility of loading individualized HRTFs: With the current version (v. 0.4.0), there is no way to import individualized HRTFs.

F. *Superpowered Web Audio JavaScript - WebAssembly SDK*

Superpowered is an API designed for low-latency audio, which provides a WebAssembly-based library [54] that can be used in audio applications running on the browser. This is done by leveraging the potential of the Web Audio API. Superpowered supports several functionalities, from audio synthesis and analysis, as well as audio effects. Regarding spatial audio, the Superpowered library for Web Audio provides a class named "Spatializer", which is the component that manages binaural audio, similar to what the PannerNode does in the WAA.

Ease of use: The Superpowered Web API is quite complex and is very dense regarding functions, methods, and classes. Compared to other tools, it might require a higher learning curve. However, at the time of writing, there are no examples showing how Superpowered can used in combination with the WebXR API. Superpowered is available through several licenses and price ranges depending on the intended use.

Support and documentation: The documentation provided on the GitHub repository and its website greatly helps developers understand the library's different functionalities. There

are also several examples of how to integrate the library into a web project and a step-by-step guide focusing especially on the spatialization part of Superpowered Web API.

Real-time microphone input support: Within the "Spatializer" class, it is possible to select a microphone as the sound source and then set the position of the audio source in the form of azimuth, occlusion, and elevation.

Reverberation: Through the use of the "Spatializer" class, it is possible to use a reverb named "global reverb", which exposes different parameters such as the "reverbDamp" for controlling high frequencies' absorption, the "reverbLowCutHz" for the low-frequency cutoff, the "reverbPredelayMs" a pre-delay expressed in milliseconds, the "reverbRoomSize" which is the size of the room a user wants to simulate, and the "reverbwidth" to control the global stereo width of the reverberation.

Rotation of a sound scene: The "Spatializer" allows performing the rotation of a sound scene relative to the position of the listener. However, the position of the HMD has to be transformed into position parameters through the WebXR API.

Possibility of loading individualized HRTFs: This function is not available at the time of this writing. It is only possible to apply custom filters to signal [55], but its implementation is not straightforward. Moreover, it is not clear how this will behave with real-time rendering.

V. Discussion

Through the results gathered from the comparison, we propose some general guidelines that can help practitioners (specialists in interactive applications or novices) make informed decisions on how to implement binaural audio in applications based on the WebXR API. Regarding the ease of use and available documentation, the most flexible tool appears to be the PannerNode, since it is part of Web Audio API. This can ensure high compatibility with the majority of modern browsers as well as the support of a very active community of users. However, if there is a need for different features not available in PannerNode, the WebSDK of atmoky and Google Resonance Audio (omnitone) for A-Frame (GRAOAF) provide high flexibility and detailed documentation, as well as they can be easily integrated into A-Frame, which makes developing VR/AR applications with the WebXR API smooth and solid.

Regarding the support of input from a microphone, the most intuitive tools to use appear to be the web SDKs of Superpowered and atmoky. However, except for 3D Tune-In Toolki JavaScript Wrapper (3DTIT JSW), all the other tools are equivalent since they interface with the functionalities provided by the WAA. It is essential to notice that most of the HMDs and AR/VR devices available in the market today provide audio input through arrays of four or more microphones. Therefore, it is unclear how well these spatial audio tools manage such kinds of inputs and whether they are easy to implement and process.

Notably, we developed this comparison focusing on operational and pragmatic aspects and did not focus on qualitative aspects of the spatial audio tools. As a future work, it will be essential to develop a more detailed understanding of the perceptual qualities of audio inputs acquired through the headsets and understand the impact of the latencies resulting between the acquisition and the rendering process.

The most versatile tool for reverberation is GRAOAF since it uses all the functionalities provided by Google's Resonance and Omnitone. Among many, this is the only tool that allows defining the environment's material properties to better simulate the acoustics of virtual spaces. GRAOAF can allow a very realistic and seamless integration of the visual and sonic components of an immersive experience. Nonetheless, suppose there is the need to create WebXR experiences that use $360°$ videos or a very accurate and photorealistic simulation of a real environment (e.g., a forest, a concert hall). In that case, we highly recommend PannerNode of WAA (with its "ConvolverNode") or jsAmbisonics, because they can load multichannel impulse responses captured in the real environment. While the tools mentioned above provide several parameters and models to create sophisticated reverberations, it is unclear which reverb types can increase the sense of realism and immersion in virtual environments. A comparison between convolutional-based reverbs (based on IR, such as PannerNode, jsAmbisonics) and ones using algorithmic techniques (e.g., atmoky, Superpowered) is needed. Additionally, it will be equally important to understand which of these types of reverberation produce better effects in VR and AR, where the types of immersion and interactions are entirely different.

Aside from 3DTIT JSW, all of the other tools allow the rotation of the sound scene, even if not natively. However, if using A-Frame, it will be easier to connect the rotation of the headset with the listener with GRAOAF, or through the A-Frame component of Atmoky [56]. Since the position of the HMD controls the rotation of the sound scene, it is not known the impact of the motion-to-sound latencies between the rotation of the headsets and the rotation delivered through the browser.

While these aspects can significantly help the general sense of immersion, using HRTFs can significantly reduce some of the challenges related to binaural audio. Even if, through the PannerNode, it is possible to use generalized HRTFs. Individualized HRTFs are preferred since they can significantly increase the sense of presence in virtual environments [57]. Among the applications we compared, only jsAmbisonics and 3DTIT JSW allow users to load individualized HRTFs, as JSON/SOFA files. The possibility of using HRTFs becomes crucial for experiences that require a high and fine rendering of the position of audio sources, especially regarding their movement in space and the perception of sound elevation.

Another issue of binaural audio that should be reduced is externalization [58]. Using spatial audio systems with HMDs that already provide a 6DoF helps to tackle this issue. A notable case is the Atmoky WebSDK, which provides a parameter named the "externalizer" that can produce the so-called "out-of-head" perceptual sensations. Atmoky is the only tool equipped with such functionality.

As already mentioned, the PannerNode of the WAA is

also used by Web Audio frameworks such as Howler.js or Tone.js and the spatialization functions found in Babylon.js and Three.Js. There is only a slight difference with Babylon.js. If developers want to manage the attenuation (or distance model in the WAA), they can bypass the native WAA attenuation using the Babylon.js custom function [59]. However, as reported in the official documentation, the use of such a function results in slower performances. Nevertheless, nowadays, most people use these described frameworks and libraries because it is possible to integrate all the audio features listed more intuitively and faster than WAA. Therefore, they are also widely used by people who are not experts in WebXR interactive web audio development.

There are other notable aspects to consider when developing spatial audio on the web. First, to ensure a smooth and realistic listening experience, there is a need for high-speed audio processing and reduced latency. Therefore, when implementing immersive applications, a powerful processing unit and a fast Internet connection is advisable. Lastly, we should notice that binaural audio systems are designed to deliver spatialized audio through headphones. However, most of the headsets for both VR and AR available today in the market are equipped with embedded speakers, and only a few provide a direct line input. This allows users to connect their own headphones directly to the device. However, since the choice is left to the user, there is a guaranteed baseline. Some devices allow the use of Bluetooth headphones, which might introduce additional latency, depending on the model and connection type. Therefore, developers must consider this aspect while developing consistent and realistic spatial audio experiences using commercial headsets and immersive systems.

VI. Conclusion

In this paper, we compared six tools for implementing interactive spatial audio in immersive applications based on the WebXR API. We analyzed them by looking from the perspective of practitioners interested in developing AR/VR experiences with WebXR. We first looked at very pragmatic aspects of each tool, such as their ease of use and the provided documentation. Moreover, we presented a detailed investigation based on four main components of spatial audio systems: reverberation, rotation of the sound scene, real-time microphone input support, and the possibility of using individualized HRTFs. Lastly, we discussed the pros and cons of the different tools compared and their limitations and use cases. While WebXR offers practitioners a standardized approach to developing immersive applications, selecting the appropriate spatial audio tool involves a series of considerations that primarily hinge on the desired user experience. We believe this work can help practitioners -being professional sound designers or beginners- to better understand the tools available and for researchers to suggest further topics for investigation.

References

[1] "Webxr device api," https://immersive-web.github.io/webxr/, Last Accessed: 2023-04-05.

[2] "WebGL," https://www.khronos.org/webgl/, Last Accessed: 2023-05-05.

[3] J. Jerald, *The VR book: Human-centered design for virtual reality*. Morgan & Claypool, 2015.

[4] J.-M. Jot, R. Audfray, M. Hertensteiner, and B. Schmidt, "Rendering spatial sound for interoperable experiences in the audio metaverse," in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*. IEEE, 2021, pp. 1–15.

[5] L. Turchet, "Musical metaverse: vision, opportunities, and challenges," *Personal and Ubiquitous Computing*, pp. 1–17, 2023.

[6] L. Turchet, M. Lagrange, C. Rottondi, G. Fazekas, N. Peters, J. Østergaard, F. Font, T. Bäckström, and C. Fischione, "The internet of sounds: Convergent trends, insights and future directions," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[7] C. Çakmak and R. Hamilton, "od: Composing spatial multimedia for the web," *Journal of the Audio Engineering Society*, vol. 68, no. 10, pp. 747–755, 2020.

[8] C. Rajguru, M. Obrist, and G. Memoli, "Spatial soundscapes and virtual worlds: Challenges and opportunities," *Frontiers in Psychology*, vol. 11, p. 569056, 2020.

[9] K. Sunder, "Binaural audio engineering," in *3D Audio*. Routledge, 2021, pp. 130–159.

[10] "Web Audio API," https://www.w3.org/TR/webaudio/, Last Accessed: 2023-05-05.

[11] "W3C - Immersive Web Working Group," https://www.w3.org/groups/wg/immersive-web, Last Accessed: 2023-05-05.

[12] "OpenXR," https://www.khronos.org/openxr/, Last Accessed: 2023-05-05.

[13] "Wolvic," https://www.wolvic.com/, Last Accessed: 2023-05-05.

[14] "ARCore Supported Devices," https://developers.google.com/ar/devices, Last Accessed: 2023-05-05.

[15] "Three.js," https://threejs.org/, Last Accessed: 2023-05-05.

[16] "A-Frame," https://aframe.io/, Last Accessed: 2023-05-05.

[17] "Babylon.js," https://www.babylonjs.com/, Last Accessed: 2023-05-05.

[18] T. Walton *et al.*, "The overall listening experience of binaural audio," in *Proceeding of the 4th International Conference on spatial audio (ICSA 2017), Graz*, 2017.

[19] J. Blauert, *Spatial hearing: the psychophysics of human sound localization*. MIT press, 1997.

[20] A. Kulkarni, S. Isabelle, and H. Colburn, "Sensitivity of human subjects to head-related transfer-function phase spectra," *The Journal of the Acoustical Society of America*, vol. 105, no. 5, pp. 2821–2840, 1999.

[21] P. Strumillo, *Advances in Sound Localization*. BoD–Books on Demand, 2011.

[22] W. G. Gardner and K. D. Martin, "HRTF measurements of a KEMAR," *The Journal of the Acoustical Society of America*, vol. 97, no. 6, pp. 3907–3908, 1995.

[23] N. Gupta, A. Barreto, M. Joshi, and J. C. Agudelo, "HRTF database at FIU DSP lab," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010, pp. 169–172.

[24] C. Faller and J. Breebaart, "Binaural reproduction of stereo signals using upmixing and diffuse rendering," in *Audio Engineering Society Convention 131*. Audio Engineering Society, 2011.

[25] M. Weitnauer, "Overview and status of binaural rendering in browsers," *Proceedings of the Annual German Conference on Acoustics*, 2018.

[26] M. Frank, F. Zotter, and A. Sontacchi, "Producing 3d audio in ambisonics," in *Audio Engineering Society Conference: 57th International Conference: The Future of Audio Entertainment Technology–Cinema, Television and the Internet*. Audio Engineering Society, 2015.

[27] F. Zotter and M. Frank, *Ambisonics: A practical 3D audio theory for recording, studio production, sound reinforcement, and virtual reality*. Springer Nature, 2019.

[28] M. A. Gerzon, "Ambisonics in multichannel broadcasting and video," *Journal of the Audio Engineering Society*, vol. 33, no. 11, pp. 859–871, 1985.

[29] S. Moreau, J. Daniel, and S. Bertet, "3d sound field recording with higher order ambisonics–objective measurements and validation of a 4th order spherical microphone," in *120th Convention of the AES*, 2006, pp. 20–23.

[30] L. McCormack, A. Politis, R. Gonzalez, T. Lokki, and V. Pulkki, "Parametric ambisonic encoding of arbitrary microphone arrays," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2062–2075, 2022.

[31] A. Avni, J. Ahrens, M. Geier, S. Spors, H. Wierstorf, and B. Rafaely, "Spatial perception of sound fields recorded by spherical microphone

arrays with varying spatial resolution," *The Journal of the Acoustical Society of America*, vol. 133, no. 5, pp. 2711–2721, 2013.
[32] T. Carpentier, M. Noisternig, and O. Warusfel, "Twenty years of ircam spat: looking back, looking forward," in *41st International Computer Music Conference (ICMC)*, 2015, pp. 270–277.
[33] A. Plinge, S. J. Schlecht, O. Thiergart, T. Robotham, O. Rummukainen, and E. A. Habets, "Six-degrees-of-freedom binaural audio reproduction of first-order ambisonics with distance information," in *Audio Engineering Society Conference: 2018 AES International Conference on Audio for Virtual and Augmented Reality*. Audio Engineering Society, 2018.
[34] E. Patricio, A. Ruminski, A. Kuklasinski, L. Januszkiewicz, and T. Zernicki, "Toward six degrees of freedom audio recording and playback using multiple ambisonics sound fields," in *Audio Engineering Society Convention 146*. Audio Engineering Society, 2019.
[35] "Networked-AFrame," https://github.com/networked-aframe/networked-aframe, Last Accessed: 2023-05-05.
[36] L. Picinali, A. Wallin, Y. Levtov, and D. Poirier-Quinot, "Comparative perceptual evaluation between different methods for implementing reverberation in a binaural context," in *Audio Engineering Society Convention 142*. Audio Engineering Society, 2017.
[37] A. Roginska and P. Geluso, *Immersive sound: the art and science of binaural and multi-channel audio*. Taylor & Francis, 2017.
[38] M. Tomasetti and L. Turchet, "Playing with others using headphones: Musicians prefer binaural audio with head tracking over stereo," *IEEE Transactions on Human-Machine Systems*, 2023.
[39] P. Majdak, Y. Iwaya, T. Carpentier, R. Nicol, M. Parmentier, A. Roginska, Y. Suzuki, K. Watanabe, H. Wierstorf, H. Ziegelwanger *et al.*, "Spatially oriented format for acoustics: A data exchange format representing head-related transfer functions," in *Audio Engineering Society Convention 134*. Audio Engineering Society, 2013.
[40] T. Carpentier, "Binaural synthesis with the web audio api," in *1st Web Audio Conference (WAC)*, 2015.
[41] A. McArthur, C. Van Tonder, L. Gaston-Bird, and A. Knight-Hill, "A survey of 3d audio through the browser: practitioner perspectives," in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*. IEEE, 2021, pp. 1–10.
[42] "Web Audio Api Panner Node," https://developer.mozilla.org/en-US/docs/Web/API/PannerNode, Last Accessed: 2023-05-05.
[43] "Howler.js," https://howlerjs.com/, Last Accessed: 2023-05-05.
[44] "Tone.js," https://tonejs.github.io/, Last Accessed: 2023-05-05.
[45] O. Warusfel, "Listen HRTF database," *online, IRCAM and AK, Available: http://recherche. ircam. fr/equipes/salles/listen/index. html*, 2003.
[46] "A-Frame Resonance Audio Component," https://github.com/mkungla/aframe-resonance-audio-component\#usage, Last Accessed: 2023-05-05.
[47] "Resonance Audio," https://github.com/mkungla/aframe-resonance-audio-component\#usage, Last Accessed: 2023-05-05.
[48] "Media Stream," https://developer.mozilla.org/en-US/docs/Web/API/MediaStream, Last Accessed: 2023-05-05.
[49] "jsAmbisonics," https://github.com/polarch/JSAmbisonics, Last Accessed: 2023-05-05.
[50] A. Politis and D. Poirier-Quinot, "Jsambisonics: A web audio library for interactive spatial sound processing on the web," in *Interactive Audio Systems Symposium*, 2016.
[51] "3D Tune-In Toolkit JSW," https://github.com/3DTune-In/3dti_AudioToolkit_JavaScript, Last Accessed: 2023-05-05.
[52] M. Cuevas-Rodríguez, L. Picinali, D. González-Toledo, C. Garre, E. de la Rubia-Cuestas, L. Molina-Tanco, and A. Reyes-Lecuona, "3d tune-in toolkit: An open-source library for real-time binaural spatialisation," *PloS one*, vol. 14, no. 3, p. e0211899, 2019.
[53] "Atmoky WebSDK," https://docs.atmoky.com/, Last Accessed: 2023-08-07.
[54] "Superpowered," https://docs.superpowered.com/?lang=js, Last Accessed: 2023-05-05.
[55] "Superpowerd Custom Filters," https://docs.superpowered.com/reference/latest/filter?lang=cpp, Last Accessed: 2023-05-05.
[56] "Atmoky A-Frame Component," https://cdn.atmoky.com/atmoky-aframe-1.0.1.js, Last Accessed: 2023-05-05.
[57] A. Väljamäe, P. Larsson, D. Västfjäll, and M. Kleiner, "Auditory presence, individualized head-related transfer functions, and illusory ego-motion in virtual environments," in *Proceedings of Presence 2004*, 2004, pp. 141–147.
[58] D. R. Begault, E. M. Wenzel, and M. R. Anderson, "Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source," *Journal of the Audio Engineering Society*, vol. 49, no. 10, pp. 904–916, 2001.
[59] "Babylon.js Distance Computation," https://doc.babylonjs.com/features/featuresDeepDive/audio/playingSoundsMusic, Last Accessed: 2023-05-05.