Comunità *et al. EURASIP Journal on Audio, Speech, and Music Processing* (2022) 2022:18
https://doi.org/10.1186/s13636-022-00250-x

EURASIP Journal on Audio, Speech, and Music Processing

SOFTWARE Open Access

PlugSonic: a web- and mobile-based platform for dynamic and navigable binaural audio

Marco Comunità[1*], Andrea Gerino[2] and Lorenzo Picinali[2]

Abstract

PlugSonic is a series of web- and mobile-based applications designed to edit samples and apply audio effects (PlugSonic Sample) and create and experience dynamic and navigable soundscapes and sonic narratives (PlugSonic Soundscape). The audio processing within PlugSonic is based on the Web Audio API while the binaural rendering uses the 3D Tune-In Toolkit. Exploration of soundscapes in a physical space is made possible by adopting Apple's ARKit. The present paper describes the implementation details, the signal processing chain and the necessary steps to curate and experience a soundscape. We also include some metrics and performance details. The main goal of PlugSonic is to give users a complete set of tools, without the need for specific devices, external software and/or hardware specialised knowledge, or custom development, with the idea that spatial audio has the potential to become a readily accessible and easy to understand technology, for anyone to adopt, whether for creative or research purposes.

Keywords: Binaural spatialisation, Spatial audio, Web audio, Augmented reality

Mathematics Subject Classification (2000): MSC code1, MSC code2, more

1 Introduction

In the recent years, the World Wide Web Consortium (W3C) and the Mozilla Foundation worked to set the basis to the development of modern web applications. The specification and release of the Web Audio (WAA) [1] and the WebGL (WGL) [2] APIs in 2011, and the later release of the HTML5 specification in 2014, aimed to satisfy the demand for audio and video processing capabilities needed to implement "sophisticated web-based games or interactive applications" [3]. To obtain performances comparable to modern digital audio workstations (DAWs) and games graphical engines, the decision was taken to move the burden of audio and video processing from the server to the client (i.e. the browser) side.

Examples of the impact of such technologies are visible in contexts like music composition and performance [4, 5], music information retrieval [6, 7], audio signal processing [8, 9], synthesis [10] and listening tests [11–13].

Similarly, research in 3D audio is investigating the potential of web technologies for loudspeaker-based and binaural (i.e. for headphones playback) rendering. Because of its intrinsically ubiquitous nature, the WAA—in conjunction with extensions being developed by the spatial audio community—offers the opportunity of a future in which spatial audio processing and rendering, and the services that make use of it, will be readily available and standardised across every browser, personal computer and mobile devices.

Therefore, we are observing major efforts on multiple aspects of web-based spatial audio, with examples in fruition [14, 15], rendering [16–18] and personalisation [19, 20].

The present work was undertaken within the PLUGGY project [21], which aimed at empowering European citizens to be actively involved in cultural heritage activities.

*Correspondence: m.comunita@qmul.ac.uk
[1]Centre for Digital Music, Queen Mary University of London, London, UK
Full list of author information is available at the end of the article

The *PlugSonic suite* consists of two main applications: PlugSonic Sample, to edit and apply audio effects, and PlugSonic Soundscape, to curate and experience binaural soundscapes in a virtual or physical space. The development of a series of complete and integrated tools could be of interest to both the spatial audio and web technologies communities.

A study has been recently published on PlugSonic focusing on the participatory design approach and the apps' integration within the PLUGGY social platform and presenting the results of an evaluation with human participants [22]. In an attempt to address requests from several researchers and professionals, and considering that [22] contains only a high-level description of the PlugSonic tools, this manuscript describes in detail the implementation and design of the developed tools and presents an evaluation of the computational performance of both web- and mobile-based implementations of PlugSonic.

The paper is organised as follows. Sections 2 and 2.1 provide a background about binaural web-based spatial audio and describe the aims of this paper. Section 3 describes all the apps that make up the *PlugSonic Suite*, main functionalities and implementation choices. In Section 4, we report on the apps' performance. Section 5 is devoted to the conclusions and potential future work.

2 Web-based spatial audio

The WAA is based on the idea of modular routing, where complex signal processing chains are created connecting many audio blocks. Built around the *AudioNode* object, this class defines audio blocks that are sources, destinations or processing units having inputs, outputs or both. The API also allows for the spatialisation of sound through an equal power panning and an HRTF (Head Related Transfer Function)-based convolution algorithm. For headphones-based applications, the equal power panning algorithm "can only give the impression of sounds located between the ears" [14] while, as reported in [20], the HRTF algorithm implemented in Google Chrome and Mozilla Firefox embeds only one set of HRTFs from the IRCAM Listen database [23]. Carpentier [20] discusses the limitations and potential drawbacks for users and developers of having only one choice of HRTFs (e.g. in-head localisation, inaccurate lateralisation, poor elevation perception and front-back confusion) and presents novel work to implement a *BinauralFIRNode* class. This class extends the WAA allowing to import custom HRTFs as FIR filters. Also Geronazzo et al. [19] discuss the limitations of the current WAA implementation and propose a framework to support personalised HRTFs. Another limitation of the WAA is in the method used for the simulation of distance; based on attenuation only, it does not account for frequency domain effects of waves propagation.

Even with its limitations, the WAA is having a great impact on web-based applications and research on spatial audio, audio narratives, games and immersive content broadcasting. Pike et al. [14] developed an object-based and binaural rendering player with head-tracking while RadioFrance nouvOson website [15] broadcasts 5.1 surround and binaural audio.

The WAA has also been used to develop high-order Ambisonics sound processing. Google Omnitone [24] implements decoding and binaural rendering up to the third order (16 channels) and uses eight virtual loudspeakers based on HRTF convolution to render binaural audio. This library is also used in the Google Resonance Audio SDK [25], which extends it with sound source directivity customisation, near-field effects, sound source spread, geometry-based reverb (room dimensions and wall materials) and occlusions. A relevant application of Omnitone in the field of soundscapes is the Storyspheres web app by the Google News Lab [26] which allows to create a soundscape for a 360° image placing sound sources in the scene and render the spatialised sound. In [16], Politis and Porier-Quinot present JSAmbisonics as a library that uses the WAA for interactive spatial sound processing on the web. This work is relevant for it supports Ambisonics of any order and allows to load custom HRTFs for Ambisonics to binaural decoding. Recent applications of JSAmbisonics are presented in [17] and [18]. In [17] Deppisch et al. present HOAST, a 360° video-platform that allows to render Ambisonics audio material up to the fourth-order, while in [18], van Tonder and Lopez describe AcousticAtlas, a web app that uses room impulse responses from natural and cultural heritage sites to render in real-time audio recorded through the user's device.

Great effort is also being put into investigating and developing web-based soundscape creation and experience tools with features and capabilities comparable to plug-ins that are used within DAWs by professionals to create immersive audio content. This trend exemplifies the interest and the aim to make 3D technologies open, familiar and available to a broad public.

The INVISO project [27] focused on the development of a web-based interface for "designing and experiencing rich and dynamic sonic virtual realities" suitable for both experts and novices. Here, the WGL was used to design a 3D interface for the creation of sound objects and navigation of the environment and the WAA for the audio rendering. The project presents interesting features like the multi-cone sound object to model complex directivity patterns, the control over sources' elevation and the possibility to define trajectories for moving sources or sound zones in which sounds are not spatialised to create ambient sounds. As discussed in previous paragraphs, the rendering quality was limited by the use of the WAA and the app did not offer control over other important

aspects of an interactive soundscape, such as the playback order or the areas within which sound sources are audible. Differently from the work presented here, which includes a mobile app, within INVISO it was not possible to experience the soundscapes in real environments.

To conclude this section, it is also worth citing other APIs available for the integration into web apps. Facebook recently released (2018) a JS implementation of their Audio360 [28] renderer to include Ambisonics decoding in the web to be used to play videos created using their Spatial Audio Workstation plug-ins.

It is evident from the literature collected above that no tool currently exists for naive users (i.e. with no or little experience in audio engineering and/or music technology) to create interactive high quality binaural audio (e.g. with distance simulation, HRTF choice, etc.) on a simple-to-use web-based interface. A recently published evaluation [22] supports our idea that web-based spatial audio tools can be designed to be easy to use also for inexperienced users. This idea is also discussed in a recent survey on web-based 3D audio technologies [29], where McArthur et al. recognise the potential of the *PlugSonic Suite* as a tool suitable for "novices with basic technical skills".

The underlying rendering engine adopted in this work [30] supports 6 degrees of freedom (DoF), allowing for translations along the 3 axes (X: front-back, Y: left-right and Z: up-down) as well as rotations around them (respectively: Roll/Tilt, Pitch and Yaw). Even though in the implementation presented here 3DoF were included (X, Y and Yaw), allowing for interactivity on both translations and rotations, the tools could be easily extended to more complex scenarios by including also Z (i.e. displacement in height), Roll and Pitch (i.e. additional freedom in terms of head movements, which in the current implementation are limited to horizontal rotations). Nevertheless, considering our purpose to strike a balance between ease and flexibility of use, these additional DoF were not considered as particularly relevant and therefore were not included.

2.1 Aims

The overall aim of this paper is to present the implementation details, design choices and some performance results of the PlugSonic Suite—with the hope these tools will be adopted for creative applications or web- and mobile-based spatial audio research.

3 PlugSonic suite

The *PlugSonic suite* is made of 4 web and mobile applications called Sample, Soundscape Create, Soundscape Experience Web and Soundscape Experience Mobile. We designed PlugSonic Sample to edit sound files and apply audio effects, PlugSonic Soundscape Create to curate interactive spatialised soundscapes and PlugSonic Soundscape Experience to explore them in a virtual (Web) or real space (Mobile).

In Fig. 1, we illustrate the creation and exploration processes and how each app contributes to the various steps towards the final result. First, a user selects the audio samples to be included in the soundscape and—if desired—they can modify and enhance the samples using PlugSonic Sample. The samples are then imported in PlugSonic Soundscape Create and the curation process can move forward by setting the options for room, sound sources, and interactions between sources and listener. During the creation of the soundscape, the user can explore the soundscape to verify the results. It is also pos-

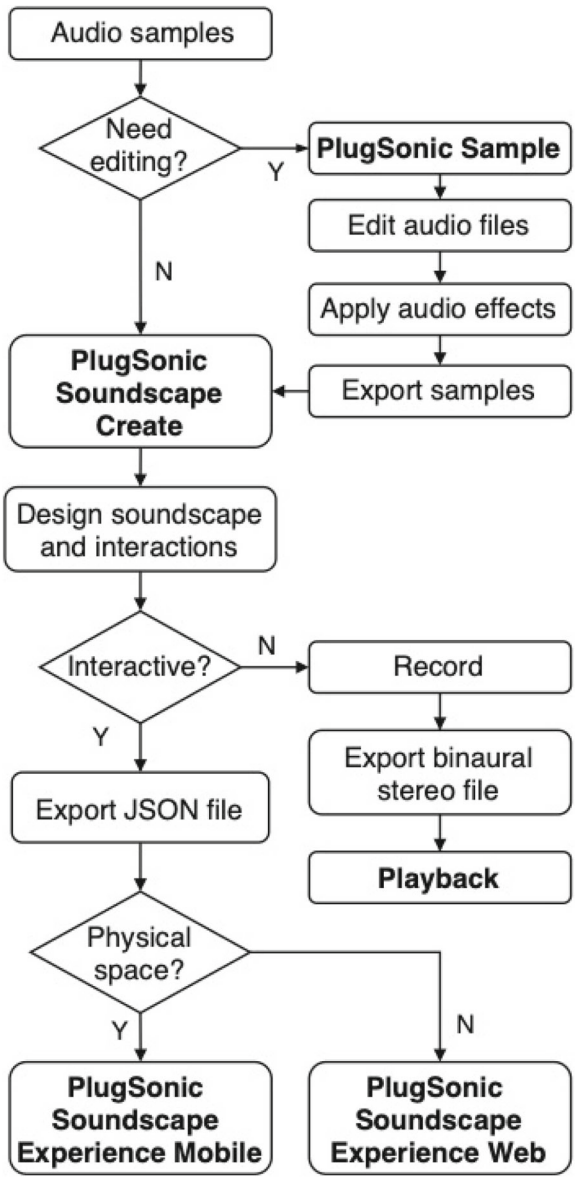


Fig. 1 Soundscape flowchart

sible to take a recording while navigating the soundscape in real-time and export it as a .wav audio file, which will include all the features of the 3D audio rendering in a standard stereo audio file. Otherwise, the full interactivity can be retained by exporting a JSON file containing all the properties of the soundscape.

At this point, the soundscape can be imported with Soundscape Experience Web, for navigation in a virtual environment within a browser; or opened with the Soundscape Experience Mobile app, to explore the soundscape through physical movements within a space.

A complete description of all the features and options available in the apps is presented in [22]

3.1 Implementation

Table 1 gives a complete overview of technologies, programming languages and libraries adopted, while a description of each app's user interface and features is given in the following sections. All the web apps in PlugSonic use established web development technologies. HTML and CSS languages are used to create the web page and define its aspect, while JavaScript (JS) to make the apps dynamic and interactive.

The user interface (UI) is built with the React [31] JS library. The main reason behind the use of React is its efficiency in managing the UI's rendering. With this library, the UI is assembled as a hierarchy of components—for which we define aspect, available interactions and behaviours. Whenever it is necessary to update the UI, e.g. as a consequence of user's actions, instead of re-rendering the whole web page, React will only re-render the components involved. In the case of PlugSonic Soundscape, because of the intensive audio processing associated with each change in the soundscape (e.g. listener's or sources' position change), it was extremely important to have an efficient and smooth UI rendering.

To manage the apps' state and properties, and therefore the apps' UI and audio processing, we use the Redux API [32]. Redux represents the whole app state as a single JS Object, updates the state as the user interacts with the app and triggers the re-rendering process when a state change occurs. Such a centralised approach simplifies the understanding and the update or extension of our web apps with new features—as well as the debug, since it allows to have an overview of all the properties determining the app's aspect at any given time.

In PlugSonic Soundscape, we also adopt the Redux-Saga JS library [33] to manage the side-effects of user's interaction on the sound rendering in an asynchronous fashion. The library allows to define event listeners on the Redux state. When events occur, separate threads execute the code that affects the sound rendering signal chain without interfering with the UI rendering. In our case, examples of actions that generate side-effects—and either change the audio signal chain structure and/or parameters—are as follows: change listener's/sources' positions, change listener's/sources' settings, add/delete or activate/deactivate sound sources and start/stop playback.

To give Sample and Soundscape a modern and familiar look, we adopt the Material design principles [34] through the use of the Material-UI library [35]. Material design was formalised by Google with the aim to synthesise "the classic principles of good design with the innovation of technology and science". The Material-UI library is essentially a set of React components that implement those principles; to put it into more tangible terms, these

Table 1 Technologies used by PlugSonic

App	Sample	Soundscape Create	Soundscape Experience Web	Soundscape Experience Mobile
Technology				
Prog. Language(s)	HTML, CSS, JS	HTML, CSS, JS	HTML, CSS, JS	Objective-C
React	✓	✓	✓	
Redux	✓	✓	✓	
Redux-Saga		✓	✓	
Material-UI	✓	✓	✓	
Material Icons	✓	✓	✓	
Responsive Design	✓	✓	✓	
Touch Enabled	✓	✓	✓	✓
Web Audio API	✓	✓	✓	
Wavesurfer.js	✓			
Recorderjs		✓	✓	
3D TuneIn Toolkit		✓	✓	✓
Apple AR Kit				✓
JSON	✓	✓	✓	✓

components are the ones used to design every Android mobile operating system and Android-based application and websites many users are already accustomed to. With the same idea of familiarity and immediate understanding, all our web apps use the Material Icons available from Google [36]. The apps also use responsive design—which allows to automatically scale the UI to different devices and screen sizes—and are *touch enabled*, to allow for their use on touch screens.

The Web Audio API is used to manage the sound processing chains in both PlugSonic Sample and PlugSonic Soundscape. In the Sample app, the WAA is used to implement all the filters and effects (using the *BiquadFilterNode*, *DynamicsCompressorNode* and *ConvolverNode* classes) as well as the audio export—which is obtained by rendering the signal chain in a memory buffer—ready to be downloaded as an audio file. The WAA is also used within wavesurfer.js [37], which handles the audio playback and waveform visualisation in the Sample app.

Within Soundscape (see Fig. 2), the WAA is only used to manage the connections between sound sources, 3D audio rendering engine and output using *AudioNode* objects. For each sound source, two objects are created: an input node, connected to an input buffer to read the sound samples, and a gain node, to control the volume. Each gain node is then connected to the mono input buffer of a binaural rendering processor. The spatialised stereo output is then connected to the master volume summing and gain node. The master volume gain node is finally connected to the *AudioContext* destination node which represents the actual audio-rendering device (speakers or headphones). The binaural rendering processor can also be bypassed since the users have the option to disable spatialisation for any sound source. Soundscape also allows to record the binaural signal in real-time and save it as a stereo wav file using the RecorderJS plugin [38].

For the binaural processing, we use the 3D Tune-In Toolkit [39]. The library is presented in [30] and gives control over full 3D listener's and sources' orientation and movement (6DoF). With respect to the WAA, the 3D Tune-In Toolkit allows for full customisation in terms of HRTFs selection, HRIR length and sampling rate. The interaural time difference (ITD) is computed from the listener's head circumference—which is passed to the rendering engine as a parameter—after convolution with the selected HRIRs. In the current implementation, the *PlugSonic Suite* allows to choose among 7 sets of HRIRs from the LISTEN dataset [23]. The HRIRs are stored as mono wav files and the rendering engine is reinitialised whenever a different set is selected. Furthermore, the head shadowing effect on the interaural level difference (ILD) is simulated for near-field sound sources. The library also implements a low-performance mode which adopts an IIR filter approximation of the HRTFs for a less demanding, albeit less realistic, spatial processing. Being natively written in C++, we use a JS compiled version [40] generated using the *emscripten toolchain* [41].

The exchange of information between the Soundscape web apps (Create and Experience) and the Soundscape Experience mobile app is possible using a JS Object Notation (JSON) file which contains all the relevant data about the soundscape (e.g. sources' position and settings, room size and shape, room floor plan image, URL links to the sound files, etc.). Although it does not follow any standard, in the future, the JSON file could be aligned with object-based 3D audio formats like BBC's Audio Definition Model [14].

3.2 PlugSonic Sample

As explained in the introduction, PlugSonic Sample can be used to edit and apply audio effects to an audio file. The modified file can then be exported for further use. Figure 3 shows the application's interface. The UI is divided into three main parts: (1) playback and edit controls, together with a button to export the modified audio file; (2) waveform display with mouse navigation and selection functions; and (3) filters and effects menu.

With exclusion of the allpass, the app implements all the filters available in the WAA through the *BiquadFilterNode* [42], which includes the following second order filters: low- and high-pass—with cutoff frequency and Q controls; band-pass—with centre frequency and bandwidth controls; low- and high-shelf—with cutoff frequency and boost/attenuation controls; peaking—with centre frequency, Q and boost/attenuation controls; notch—with centre frequency and Q controls. The dynamic range

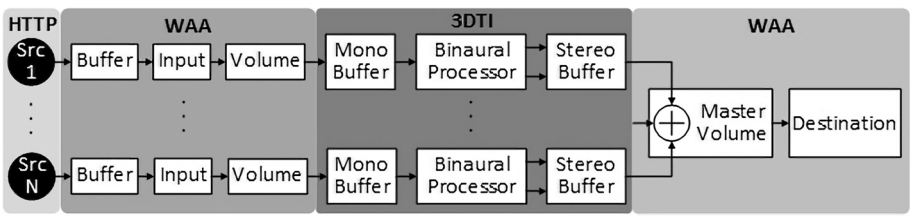


Fig. 2 PlugSonic Soundscape signal chain

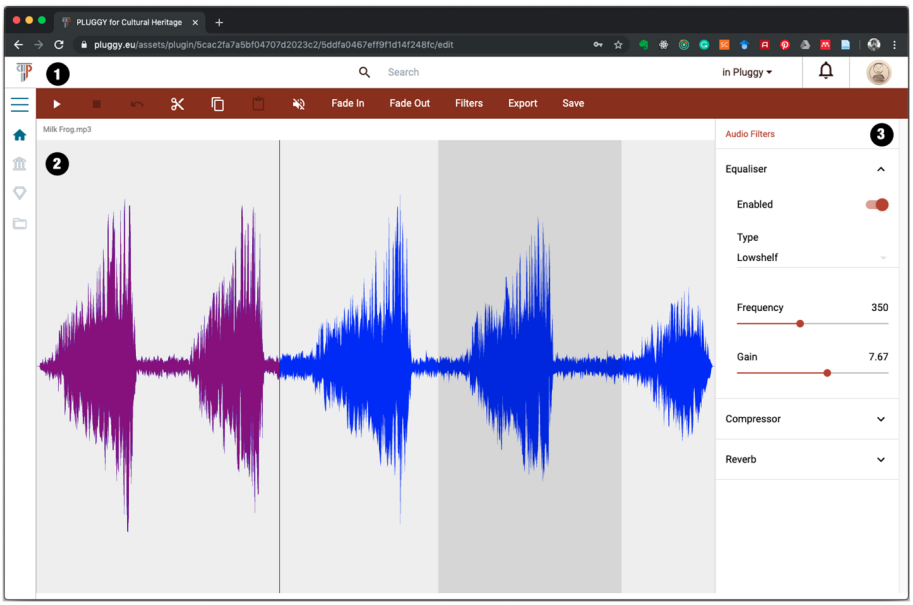


Fig. 3 PlugSonic Sample User Interface

compressor is implemented in the *DynamicCompressorNode* class [43] and includes controls for attack and release times, threshold, knee and compression ratio. The monophonic reverb uses the *ConvolverNode* class, which computes a linear convolution. The three impulse responses available for convolution are the ones used in the 3D Tune-In Toolkit, which correspond to small, medium and large rooms.

3.3 PlugSonic Soundscape Create

PlugSonic Soundscape Create is used for the curation of interactive 3D audio narratives and soundscapes. The UI, shown in Fig. 4, is divided into three main sections: (1) the top bar, hosting playback control buttons; (2) the "room", which shows the physical/virtual space described in the soundscape and includes icons that represent sound sources and listener; and (3) the dismissable side menu,

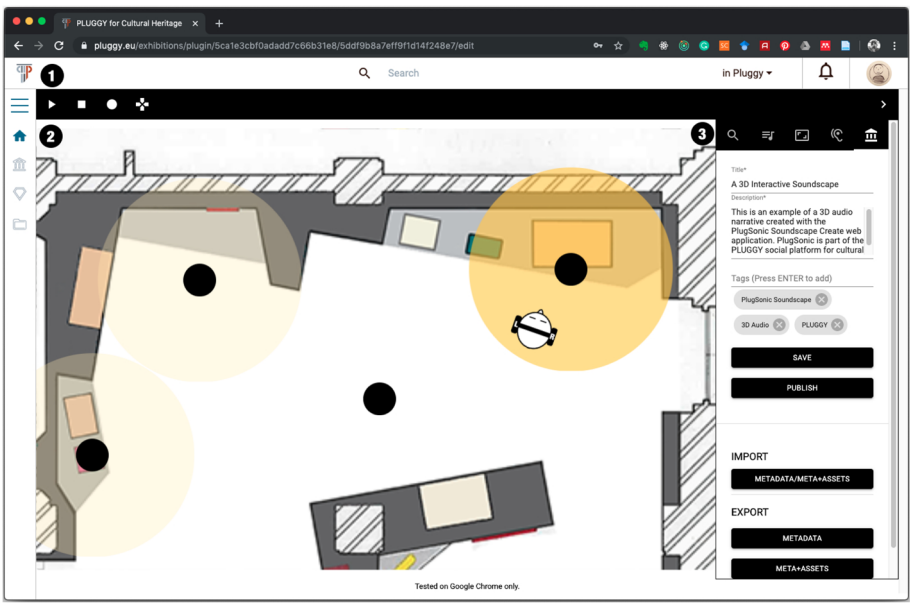


Fig. 4 PlugSonic Soundscape Create User Interface

containing all the controls and options to modify the soundscape.

The curation process would typically proceed as follows. The user starts by setting shape (round/rectangular) and size (width/depth/height) of the room and, if desired, chooses an image to be used as floor-plan. After importing the sound sources, the user can adjust individually each source's settings. A complete specification of all the options available for each sound source is available in [22]; here, we limit our description to what we believe are the most interesting and useful controls: *Position*—absolute or relative to the listener; *Loop*—to choose if the sound source will loop or play only once; *Reach*—to control the interaction between listener and sound source (when ON the listener will be able to hear the sound source only when inside the interaction area); *Reach radius*—to set the size of the interaction area; and *Timings*—to set an order in the reproduction of any sound source by constraining the playback to the reproduction of another sound source.

Once complete, the user can export the soundscape's metadata. The metadata can be imported into any application capable of interpreting it. In the case of PlugSonic, the metadata can be imported back into Soundscape Create or it can be opened with Soundscape Experience. There are two formats available to export the soundscape: the simple metadata, which requires an Internet connection to retrieve the audio files, or the metadata including the assets, in which case, the soundscape can be experienced off-line since the audio files' data is added to the exported soundscape file.

The UI presented here is the result of both: a complete redesign and extension, as well as the outcome of the expert's evaluation described in [44].

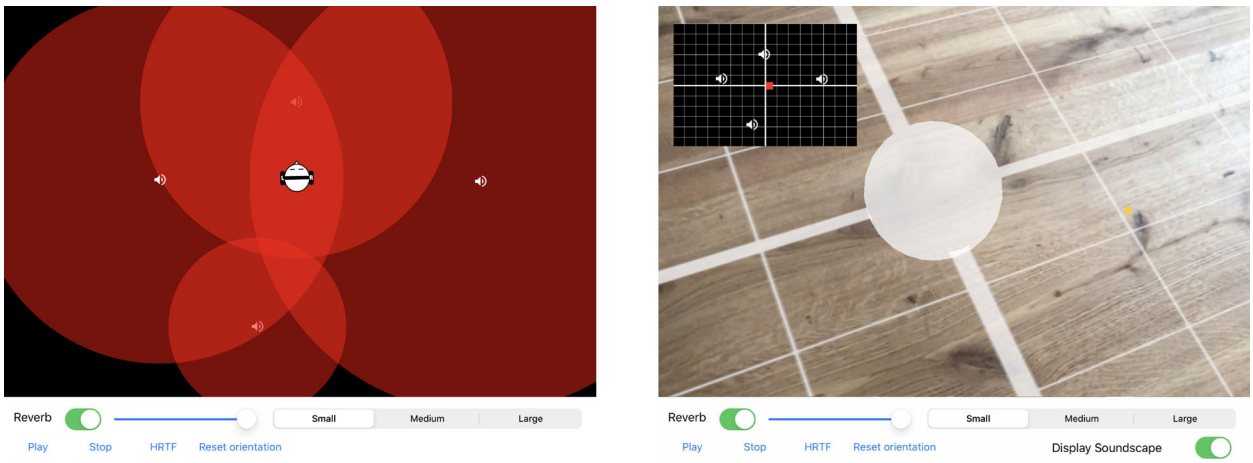
3.4 PlugSonic Soundscape Experience Web

The Soundscape Experience Web app was developed to allow the navigation of soundscapes using any device capable of running a web browser (PC, laptop, tablet or mobile). The app's UI is the same as Soundscape Create, stripped down of all the features that allow to modify the soundscape—the only controls available are the playback and record buttons, the access to the touch arrows controls and the listener's options.

3.5 PlugSonic Soundscape Experience Mobile

Soundscape Experience Mobile is designed to explore a soundscape on a mobile device and offers two separate interaction modes. In the first mode (Fig. 5a), users can explore the soundscape by moving the listener's icon using their finger and can change orientation by rotating the device. The second interaction mode provides an immersive experience by enabling users to explore a soundscape according to their movements in the real-world. In this case, we use ARkit [45], a technology developed by Apple to support detection and tracking of planar surfaces by analysis of video frames captured by a device's camera and data collected from inertial sensors. The framework provides anchors in a real-world environment that can be used to determine the relative position of the device with respect to the detected plane (Fig. 5b). In this mode, the interaction can also extend to the third dimension since the library detects planes at several elevations (e.g. steps in a staircase).

Soundscapes can be loaded using the QR code reader included in the app or importing the metadata. The app includes buttons to reset the listener's orientation, play and stop, and choose the HRTFs. It also allows some



(a) Touch-based

(b) Camera- and sensors-based

Fig. 5 PlugSonic Soundscape Experience Mobile UI

control over the convolution-based reverb settings. The reverb is based on the Reverberant Virtual Loudspeaker (RVL) method introduced in [46], and integrated within the 3D Tune-In Toolkit [30]. This method is based on virtual Ambisonics and Binaural Room Impulse Responses (BRIRs); users can import their own BRIRs (a minimum of 4 BRIRs is required for any given space, one every 90° of Azimuth on the horizontal plane), to simulate—for example—the reverb of a specific real space in which the navigation will take place; or they can use one of the three sets pre-loaded in the app, which allow to simulate a generic small, medium or large room.

4 Performance

To check for compatibility we tested the web apps both on MacOS and Windows using the following browsers: Google Chrome, Mozilla Firefox, Microsoft Edge and MacOS Safari. Both Soundscape Experience Web and Experience Mobile were tested for loading time and CPU/memory requirements. We tested Experience Web on a desktop pc with an Intel i7-8700 @ 3.20GHz, 16GB RAM, running Windows 10. Table 2 shows the average and standard deviation for loading times (in ms) when using Imperial College's Wi-Fi network. The quantities were calculated over 5 runs on Google Chrome. A good portion of the loading time (almost two seconds) is spent on scripting, due to the initialisation of the binaural processor.

To test CPU and memory requirements with the number of sources, we used a 30-s white noise excerpt. Figure 6 shows the percentage of CPU when rendering the soundscape in several listener's conditions: stationary and moving in circle (with and without the performance mode option). For each condition in Fig. 6, the lines show the maximum number of sources the browser was able to render without sound and graphical performance degradation. We can notice how there is a linear relation between number of sources and CPU use with a considerable difference between stationary and moving conditions. The graph also shows how the performance mode allows for the real-time rendering of up to 50 sources.

The same tests were performed to evaluate Experience Mobile on an iPad Pro (model A1701) with iOS 14.3. All metrics were measured with the Xcode profiler application and using the touch-based interaction mode only; the AR based one does not influence CPU usage. On average, 2.27 s are required to load the app and start playback (n=5, SD=0.007). Similarly to what was observed with

Table 2 Soundscape Create Web loading time (ms)

	Load	Script	Render	Paint	Other	Idle	Total
Avg.	1.2	1902.6	11.4	2	50.4	782.4	2445
SD	0.5	56.2	0.6	0	1.7	136.9	449.9

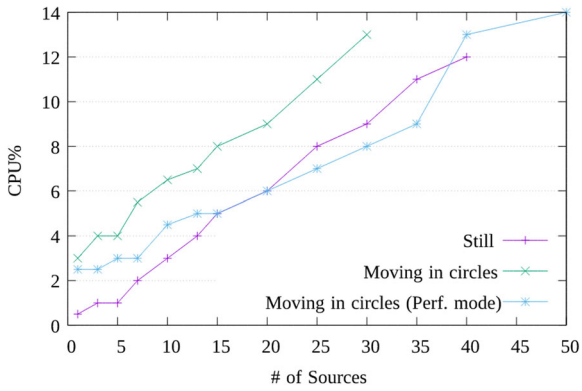


Fig. 6 Soundscape Experience Web CPU usage

the web application, CPU and memory usage grow linearly with the number of sources. However, compared to the former, memory and CPU usage of the latter is lower (see Figs. 7 and 8). Also, the mobile app has no hard limitation on the number of sources and could render 50 concurrently-playing sources also in the default high-quality mode.

When tested on an iPhone 12 Pro Max, the device was able to render up to 15 sources using Experience Web. We did not test Experience Mobile on iPhone although, based on the results above, we would expect better performance with respect to the browser-based version.

5 Conclusions and future work

PlugSonic aimed to demonstrate that spatial audio technologies and software can be designed to be accessible to anyone, without having to compromise on the binaural rendering quality or the flexibility of interactions available to the content creator, whether institutions or general public.

Due to their ubiquitous nature, web- and mobile-based applications have the potential to be exploited to ease the curation and experience of 3D interactive soundscapes and audio narratives. In this work, we used these

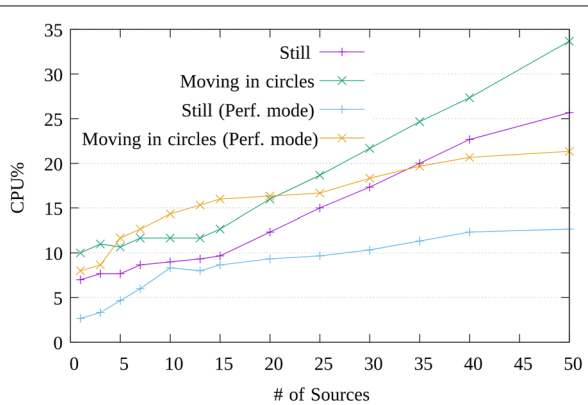


Fig. 7 Soundscape Experience Mobile CPU usage

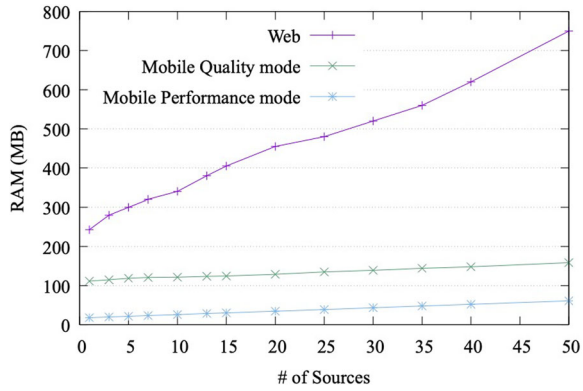


Fig. 8 Soundscape Experience memory usage

technologies to develop applications that can be used to edit sound files and create, test and experience such soundscapes in a low friction environment that allows to transition from a web browser to a physical space when desired.

PlugSonic could be integrated within an immersive music performance system, both real-time or offline. It could be used as audio renderer for a collaborative virtual reality music making experience, or again to allow individuals to remotely experience and interact with audio-based installations. Considering PlugSonic's mobile nature, its use could be extended to augmented reality scenarios, for example enhancing live concerts and/or allowing co-located and remote participatory performance applications.

There is also the potential for PlugSonic to become a web- and mobile-based 3D audio research evaluation tool. In fact, even if other web-based audio evaluation tools are already available [12, 13, 47], none of these focuses specifically on binaural audio, allowing the flexibility to select and individualise HRTFs, and opening up to online studies on, for example, HRTF selection and adaptation. Moreover, the hearing loss and hearing aid simulation algorithms available from the 3D Tune-In toolkit (not yet implemented in PlugSonic, but already available within the JS wrapper) would also allow for specific evaluations on hearing impairment. PlugSonic could be integrated in a listening test framework that lets researchers easily design different types of online tests (e.g. AB, ABX, MUSHRA). Both the web- and mobile-based apps could be exploited for localisation tests or games within a virtual or physical space.

6 Links

github.com/lpicinali/PlugSonic-soundscape
github.com/lpicinali/PlugSonic-sample
github.com/lpicinali/PlugSonic-experience-mobile

Acknowledgements
Not applicable

Authors' contributions
Conceptualisation, LP, MC and AG; methodology, MC, LP and AG; software development, MC and AG; validation, MC and LP; resources, MC and LP; writing—draft, MC; writing—review and editing, MC, AG and LP; supervision, LP; project administration, LP and MC; funding acquisition, LP. All authors read and agreed to the submitted version of the manuscript.

Funding
This work was supported by the PLUGGY project (https://www.pluggy-project.eu/), EU Horizon 2020 research and innovation programme under grant agreement No 726765.

Availability of data and materials
Not applicable

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Centre for Digital Music, Queen Mary University of London, London, UK. [2]Dyson School of Design Engineering, Imperial College London, London, UK.

Received: 28 November 2021 Accepted: 7 June 2022
Published online: 15 July 2022

References

1. Web Audio API. https://www.w3.org/TR/webaudio. Accessed 15 Nov 2021
2. WebGL. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API. Accessed 15 Nov 2021)
3. Web Audio API specification proposal. https://www.w3.org/2011/audio/drafts/1WD/WebAudio/. Accessed 15 Nov 2021
4. X. Favory, X. Serra, Multi web audio sequencer: collaborative music making. arXiv preprint arXiv:1905.06717 (2019)
5. Q. Lan, A. R. Jensenius, in *Proceedings of the International Web Audio Conference (WAC)*. Quaverseries: A live coding environment for music performance using web technologies (NTNU, 2019), pp. 41–46
6. A.A. Correya, et al., in *Proceedings of the 21st International Society for Music Information Retrieval Conference; 2020 Oct 11-16; Montréal, Canada*, ed. by J. Cumming, J. Ha Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zagerle, and T. de Reuse. Essentia. js: A JavaScript library for music and audio analysis on the web (International Society for Music Information Retrieval (ISMIR), Canada, 2020), pp. 605–612
7. H. Rawlinson, N. Segal, J. Fiala, in *1st Web Audio Conference (WAC). Paris, Fr.* Meyda: an audio feature extraction library for the web audio api, (2015)
8. M. Buffa, J. Lebrun, in *Web Audio Conf 2018*. WebAudio virtual tube guitar amps and pedal board design, (2018)
9. N. Jillings, Y. Wang, J. D. Reiss, R. Stables, in *Audio Engineering Society Convention 141*. JSAP: A plugin standard for the web audio API with intelligent functionality (Audio Engineering Society, Suite, 2016)
10. P. Bahadoran, A. Benito, T. Vassallo, J. D. Reiss, in *Audio Engineering Society Convention 144*. FXive: A web platform for procedural sound synthesis (Audio Engineering Society, Suite, 2018)
11. L. Blin, O. Boeffard, V. Barreaud, in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, ed. by N.(.Conference.C.hair). Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, and D. Tapias. WEB-Based Listening Test System for Speech Synthesis and Speech Conversion Evaluation (European Language Resources Association (ELRA), Marrakech, 2008). http://www.lrec-conf.org/proceedings/lrec2008/
12. N. Jillings, B. Man, D. Moffat, J. D. Reiss, in *Proceedings of the 12th International Conference on Sound and Music Computing*. Web Audio Evaluation Tool: A browser-based listening test environment, (2015), pp. 147–152
13. M. Schoeffler, F.-R. Stöter, B. Edler, J. Herre, in *1st Web Audio Conference*. Towards the next generation of web-based experiments: A case study

assessing basic audio quality following the itu-r recommendation bs. 1534 (MUSHRA), (2015), pp. 1–6
14. C. Pike, P. Taylour, F. Melchior, in *Proceedings of the 1st Web Audio Conference*. Delivering object-based 3D audio using the web audio api and the audio definition model, (2015)
15. H. Dejardin, E. Ronciere, in *AES Conference: 57th International Conference: The Future of Audio Entertainment Technology - Cinema, Television and the Internet*. nouvOson website: how a public radio broadcaster makes immersive audio accessible to the general public, (Suite, 2015)
16. A. Politis, D. Poirier-Quinot, in *Interactive Audio Systems Symposium*. JSAmbisonics: a web audio library for interactive spatial sound processing on the web, (2016). https://hal.archives-ouvertes.fr/hal-01790246
17. T. Deppisch, N. Meyer-Kahlen, B. Hofer, T. Latka, T. Zernicki, in *Audio Engineering Society Convention 148*. HOAST: A higher-order ambisonics streaming platform (Audio Engineering Society, Suite, 2020)
18. C. Van Tonder, M. Lopez, in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*. Acoustic Atlas–auralisation in the browser (IEEE, New York, 2021), pp. 1–5
19. M. Geronazzo, J. Kleimola, P. Majdak, in *Proc. 1st Web Audio Conference*. Personalization support for binaural headphone reproduction in web browsers, (2015)
20. T. Carpentier, in *1st Web Audio Conference (WAC)*. Binaural synthesis with the web audio API, (2015)
21. V. Lim, N. Frangakis, L. M. Tanco, L. Picinali, in *Advances in Digital Cultural Heritage*. PLUGGY: A pluggable social platform for cultural heritage awareness and participation (Springer, Cham, 2018), pp. 117–129
22. M. Comunità, A. Gerino, V. Lim, L. Picinali, Design and evaluation of a web-and mobile-based binaural audio platform for cultural heritage. Appl. Sci. **11**(4), 1540 (2021)
23. Ircam LISTEN. http://recherche.ircam.fr/equipes/salles/listen. Accessed 15 Nov 2021
24. Google Omnitone. https://googlechrome.github.io/omnitone/. Accessed 15 Nov 2021
25. Google Resonance. https://resonance-audio.github.io/resonance-audio/. Accessed 15 Nov 2021
26. Google Storyspheres. https://storyspheres.com/. Accessed 15 Nov 2021
27. A. Çamcı, K. Lee, C. J. Roberts, A. G. Forbes, in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. INVISO: a cross-platform user interface for creating virtual sonic environments (ACM Siggraph, New York, 2017), pp. 507–518
28. Facebook Audio 360. https://facebook360.fb.com/spatial-workstation/. Accessed 15 Nov 2021
29. A. McArthur, C. van Tonder, L. Gaston-Bird, A. Knight-Hill, in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*. A survey of 3d audio through the browser: practitioner perspectives (IEEE, New York, 2021), pp. 1–10
30. M. Cuevas-Rodríguez, L. Picinali, D. González-Toledo, C. Garre, E. de la Rubia-Cuestas, L. Molina-Tanco, A. Reyes-Lecuona, 3D Tune-In Toolkit: An open-source library for real-time binaural spatialisation. PloS ONE. **14**(3), e0211899 (2019)
31. ReactJS. https://reactjs.org. Accessed 15 Nov 2021
32. Redux. https://redux.js.org. Accessed 15 Nov 2021
33. Redux Saga. https://redux-saga.js.org. Accessed 15 Nov 2021
34. Material Design. https://material.io/design. Accessed 15 Nov 2021
35. Material UI. https://material-ui.com. Accessed 15 Nov 2021
36. Material Icons. https://material.io/resources/icons. Accessed 15 Nov 2021
37. WavesurferJS. https://wavesurfer-js.org. Accessed 15 Nov 2021
38. RecorderJS. https://github.com/mattdiamond/Recorderjs. Accessed 15 Nov 2021
39. 3D Tune-In Toolkit - Repository. https://github.com/3DTune-In. Accessed 15 Nov 2021
40. 3D Tune-In Toolkit - JS package. https://www.npmjs.com/package/@reactify/3dti-toolkit. Accessed 15 Nov 2021
41. Emscripten. https://emscripten.org. Accessed 15 Nov 2021
42. WAA - Biquad Filter Node. https://developer.mozilla.org/en-US/docs/Web/API/BiquadFilterNode. Accessed 15 Nov 2021
43. WAA - Dynamic Compressor Node. https://developer.mozilla.org/en-US/docs/Web/API/DynamicsCompressorNode. Accessed 15 Nov 2021
44. M. Comunità, A. Gerino, V. Lim, L. Picinali, in *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. Web-based binaural audio and sonic narratives for cultural heritage, (Suite, 2019)
45. Apple AR Kit. https://developer.apple.com/augmented-reality/. Accessed 15 Nov 2021
46. I. Engel, C. Henry, S. V. Amengual Garí, P. W. Robinson, L. Picinali, Perceptual implications of different ambisonics-based methods for binaural reverberation. J. Acoust. Soc. Am. **149**(2), 895–910 (2021)
47. S. Kraft, U. Zölzer, in *Linux Audio Conference*. BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality, (2014)

Publisher's Note