Using Web Audio Plugins in 3D environments

Michel Buffa, Marco Winckler, Quentin Escobar, Samuel Demont, Ayoub Hofr, Adam Mir-Sadjadi

Using Web Audio Plugins in 3D environments

Michel Buffa, Marco Winckler
University Côte d'Azur
firstname.name@univ-cotedazur.fr

Quentin Escobar, Samuel Demont,
Ayoub Hofr, Adam Mir-Sadjadi
University Côte d'Azur
firstname.name@etu.univ-cotedazur.fr

ABSTRACT

This article explores the frontiers of interactive, modular music creation by presenting how existing WAM plugins, available on the Web in their 2D version, can be deployed in real-time 3D collaborative environments without any modification of their existing source code. We present a system in which users build and play with modular audio graphs, which we call *music installations*, using both audio processors and note generators, all instantiated as headless WAM plugins, with additional, dynamic 3D user interfaces that replace their 2D counterparts. These 3D interfaces are either generated automatically by introspection from the plugins' metadata, or created manually using a dedicated interactive 3D GUI editor.

The entire system is hosted in WAM JAM Party, a multi-user 3D application where participants assemble, spatialize and co-edit real-time musical installations for performance or educational use.

1. INTRODUCTION

From hardware modular synthesizers to virtual plugin chains in digital audio workstations (DAWs), musicians and sound designers have long worked with modular systems to create complex, expressive audio. However, these tools remain largely confined to 2D interfaces, siloed workflows, and single-user environments. Meanwhile, emerging technologies in web audio, real-time graphics, and multi-user networking are rapidly reshaping the creative landscape.

In this paper, we demonstrate how Web Audio Modules (WAMs) [14, 8, 7] —a browser-native plugin format— can form the foundation of immersive, collaborative, and modular audio environments built entirely in the browser. We describe a system that allows users to connect modular WAMs into spatially arranged audio installations, where each module can be interacted with in 3D, modulated at audio rate, and collaboratively edited by multiple participants in real time.

Our contributions include:

- A system for automatic and manual construction of 3D GUIs for WAMs based on plugin metadata or user custom design.
- The development of an API enabling deeper customization of 3D graphical user interfaces —supporting features such as animation and advanced event handling— has been initiated. A 3D piano roll WAM and an interactive virtual drumkit were implemented as test cases to validate and demonstrate the API's capabilities.
- A multi-user, real-time immersive host application, WAM JAM Party, where users assemble, spatialize, and perform with modular audio systems in shared 3D spaces. It was conceived to foster creativity and support collaboration among multiple players.

This work operates at the intersection of networked performance, modular synthesis, embodied interaction and web technology. We propose a new model for collaborative sound-making on the Web that is sough to foster the UX and creativity among multiple players. It reframes audio plugins not as desktop-bound utilities, but as spatial, sculptural, and performative tools

The remainder of this paper is structured as follows: Section 2 reviews related work on modular audio systems and interactive 3D environments, highlighting both native and web-based approaches. Section 3 presents our methodology for extending Web Audio Modules (WAMs) with 3D graphical interfaces—either automatically generated from plugin metadata or custom-designed—without requiring access to the plugin's source code. In Section 4, we describe how these 3D-enabled plugins are integrated into WAM Jam Party, a multi-user, immersive host environment that enables collaborative construction and real-time manipulation of interactive music installations. Section 5 discusses prospective enhancements to the system, including new interaction models, user interface strategies, and instrumentation paradigms adapted to immersive contexts. Section 6 concludes with a summary of contributions and outlines future research directions.

2. RELATED WORKS

Although there is a considerable amount of research on sound spatialization[15, 9] and numerous examples of 3D audio visualization[1], there is relatively little investigation

[1]https://github.com/willianjusten/awesome-audio-visualization

Web Audio Conference WAC-2025, November 19–21, 2025, Paris, France.

into the representation and use of 3D plugins (instruments, audio effects, others...), particularly in web-based environments or when accessed via virtual reality headsets.

However, since the resurgence of virtual reality technologies around 2020, researchers and musicologists have explored the design and performance of virtual instruments controlled through gestural interaction. Notable examples include Rob Hamilton's Coretet project, which models bowstring interactions and idiomatic performance techniques of traditional string instruments such as violins, violas, cellos, and double basses in real-time VR environments [13]. Another example is the work of Jonathan Bell and Benedict Eris Carey's TENOR 2019 performance, where musicians engaged with virtual cellos and animated notation through AR/VR headsets [3]. These systems are typically developed as native applications developed with Unity or Unreal Engine, paired with custom real-time audio synthesis engines. Beyond classical music paradigms, platforms like PatchXR/PatchWorld [1][2] offer mixed-reality environments for building, performing, and sharing 3D modular instruments, where users manipulate and interconnect spatialized audio components. PatchXR has also been used by Bell in real-time VR performances [2]. Other immersive platforms include Synthspace[3] and SynthVR[4], which replicate Eurorack-style modular synthesis within virtual reality. Music-focused VR applications also appear in gaming contexts, such as SoundStage[5] (2016), which features a virtual theremin and drum kit for HTC Vive, or more recent platforms like Tranzient[6] and TribeXR[7], which enable live looping, music production, and DJ performance in VR.

although 3D graphics and real-time audio have been fully available over the Web platform for more than a decade, examples of virtual instruments or audio effects designed for interactive 3D environments remain relatively rare. Following the introduction of the Web Audio and WebGL APIs, some first experiments have emerged, including a 3D reproduction of the MiniMoog[8] (2015) and Jean-Michel Jarre's laser harp, presented by David Rousset as part of the WebXR demonstrations at the Web Audio Conference 2022[9]. Most recently, Google's "AR Synth" experiment (2025)[10] allowed users to explore photorealistic 3D models of iconic electronic instruments from the Swiss Museum of Electronic Musical Instruments (SMEM), such as the Memorymoog, ARP synthesizers, Roland CR-78, Akai S900, and Fairlight CMI. These models can be accessed via standard web browsers and XR headsets. However, these visualizations typically offer minimal interactivity, typically allowing only a single parameter to be manipulated (e.g., a filter cutoff), and are not intended to emulate the full audio or functional behavior of the original hardware. Furthermore, they rely on specific implementations and do not conform to standardized, interoperable plugin architectures.

[2]https://patchxr.com/
[3]https://store.steampowered.com/app/1355640/SYNTHSPACE/
[4]https://www.meta.com/experiences/synthvr/3748465338566486/
[5]https://github.com/googlearchive/soundstagevr
[6]https://www.aliveintech.com/
[7]https://www.tribexr.com/
[8]http://aikelab.net/websynthv2/
[9]https://www.youtube.com/watch?v=izYsT2_sKzM
[10]https://artsexperiments.withgoogle.com/ar-synth/

Recent work within the domain of immersive music interaction includes contributions to the notion of the "Musical Metaverse" [18], with investigations into multi-user music experiences in XR. Boem et al. [4] explored collaborative interaction within WebXR environments, while Dziwis et al. developed Orchestra [12], an open-source framework for live performance within browser-based metaverse contexts. This includes capabilities such as live streaming of spatialized audio and video, generative algorithmic music performance, live coding in multiple languages [11], and the execution of PureData-based instruments via PDXR [10].

However, none of these systems, to the best of our knowledge, support interoperable, extensible audio plugin formats. As of 2025, Web Audio Modules (WAMs) remain the only actively maintained community-driven standard for web-based audio plugins and the work presented in this paper is, to our knowledge, the first to demonstrate their integration within interactive 3D environments, including immersive contexts enabled by the WebXR API.

Table 1: Comparison of immersive modular audio environments

System	Web-based	Plugins	Multi-user	3D GUI / XR Integration
WAM Jam Party	Yes	Yes (WAM URIs, no source code needed)	Yes	Fully immersive 3D interfaces; auto/manual GUI generation
PatchXR	No	Yes (custom patch format, limited plugin integration)	Yes, experimental	Full VR support; node-based editing in 3D space
SoundStage VR	No	No	No	VR musical sandbox with drum kits and synths, but limited modularity

Table 1 compares WAM Jam Party with other immersive audio systems based on key features.

Relying on WAMs offers immediate access to a growing ecosystem of over a hundred plugins, including note generators (e.g., step sequencers, random and Turing machine-based generators), instruments (synthesizers, samplers), real-time effects, parameter modulators, and extensions that enable interaction with video streams or 3D geometry—such as WebGL shader-based animations. This approach renders host applications inherently extensible, removing the need for local installation: plugins can be invoked directly via URI.

In the following sections, we detail the architecture and methodology used to generate interactive 3D GUIs for WAMs and describe their integration into the multi-user host environment WAM Jam Party.

3. ADDING 3D GUIS TO EXISTING WAM PLUGINS

3.1 WAM in 3D scenes

Thanks to the standardized nature of web technologies, the Web Audio API can run seamlessly in any 3D web ap-

plication modern web browser, including those embedded in immersive headsets such as the Meta Quest series and the Apple Vision Pro. Naturally, Web Audio Modules (WAMs) plugins can also be executed in a headless mode, without their traditional 2D GUIs, and integrated directly into any WebGL or WebGPU-based 3D scenes. Recently, the authors of the WAM standard have collaborated with core developers from the Babylon.js project —a high-level 3D engine built on top of WebGL, WebGPU, WebXR, and Web Audio— to better integrate WAM plugins within Babylon.js applications. One illustrative example, available online[11], involves loading an Oberheim Ob-Xd synthesizer WAM into a Babylon.js scene and triggering notes by clicking on the keys of a 3D-modeled piano keyboard. The implementation is concise and publicly available, showcasing how Web Audio plugins can now be embedded in fully immersive, web-native musical interfaces.

However, the dandling question is : *how to provide effective 3D user interfaces to support interactions with WAMs?*

3.2 Challenge: there is no equivalent to the DOM API for 3D environments

One of the biggest challenges is the absence of conventional web interface technologies such as HTML, CSS, and Web Components. Unlike 2D web applications, where graphical elements can be easily styled and composed using declarative markup and standard DOM APIs, 3D environments require UI elements to be constructed and managed entirely within the rendering context of a graphics engine. To address this, we introduced an intermediate representation based on a structured JSON configuration file that defines the layout, appearance and parameter bindings of a 3D GUI independently of the rendering engine. This format serves as a declarative intermediate representation, allowing us to interpret and render plugin interfaces through an adapter layer tied to a specific 3D graphics library. For our implementation, we selected Babylon.js, due to prior experience and its mature GUI framework, advanced WebXR support and scene management features. However, the adapter model is intentionally designed to be modular, and could be extended to support other rendering libraries in the future, such as Three.js, A-Frame, or custom WebGL engines, ensuring portability and long-term flexibility.

The configuration file includes the URI of the WAM plugin, along with global properties such as size, color, texture, and labeling. It also specifies a list of GUI elements (e.g., knobs, sliders, pads), each mapped to one of the plugin's parameters via its identifier.

This JSON file can be automatically generated for simple plugins using default conventions, or manually authored using an interactive 3D GUI editor for more complex instruments. At runtime, a rendering adapter (implemented with Babylon.js in our case) interprets this configuration to instantiate the corresponding objects as 3D meshes, bind their behaviors to WAM parameters, and render them in the 3D scene. A special class implements this rendering adapter and is used by the host application that will manage the WAM plugins.

3.3 Automatic 3D GUI Generation

For simple audio effects—such as delay, reverb, overdrive, or distortion—that typically expose a limited number of parameters, we designed an automatic GUI generation process. Upon loading a plugin via its URI and instantiating it without its 2D interface, we invoke its getParameterInfo() method. This method returns a set of parameter metadata including: parameter name and label, default value, minimum and maximum range, step size and optional descriptions and units, see an example in Figure 1. Using this metadata, it is then possible to generate automatically a generic 3D interface composed of simple 3D geometric controllers (e.g., sliders, knobs, buttons) mapped to plugin parameters. Each parameter is initially rendered using a default shape and interaction model (e.g., a draggable cylindrical mesh for "gain" or "frequency"). This process follows a "convention over configuration" paradigm: default mappings are usable as is, but fully overridable. A corresponding JSON configuration file (using the intermediate representation discussed in the previous section) is associated with each auto-generated GUI, allowing developers or designers to override: controller shape and geometry, color and texture (e.g., apply a metallic or wood-like surface), label placement and font, interaction behavior (e.g., linear drag, rotation, two-handed scaling in VR). Parameter controls can also be marked as hidden to avoid generating overly complex GUIs, which would otherwise require a more carefully designed and readable layout.

This solution enables functional, coherent 3D interfaces, with minimal effort. You can see examples of auto-generated 3D Interfaces for existing WAMs in Figures 1 and 2.

Figure 1: *Automatic generation of a 3D GUI by introspecting the parameter set of a WAM. In this case, the 3D GUI uses the default internal parameter names, which may differ from those used in the 2D version where an image is set as a div background via a CSS property. These names can be customized if needed.*

This approach was previously applied to a variety of effects and virtual instruments in earlier experiments [6]. However, user testing revealed that participants had difficulty distinguishing between different plugins, as their visual appearances were largely uniform—despite minor variations in labels and color schemes (Figure 2). Users expressed a desire for a more distinctive "visual signature," similar to what can be seen in VST plugins, which often vary by vendor or plugin type (textures, shapes of the plugin, buttons, sliders, etc.). Moreover, in the case of more complex plugins

[11] https://playground.babylonjs.com/#QO4OTN#10

such as synthesizers with dozens of parameters/controllers (knobs, sliders, switches), the automatically generated GUIs suffered from oversimplification. In practice, to maintain clarity, many instrument plugins required most parameters to be hidden, exposing only the essential controls—such as a preset menu mapped to a draggable vertical cylinder and a few key parameters.

These limitations highlighted the need for a more flexible solution and ultimately motivated the development of a dedicated 3D GUI editor.

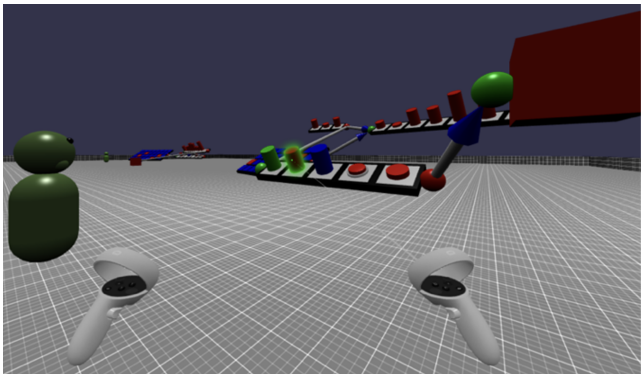

Figure 2: *WAMs with auto-generated 3D GUIs can be hard to tell apart as they tend to all look similar.*

3.4 Custom GUIs via a 3D GUI Editor

For more complex WAMs—such as synthesizers or multi-effect units featuring dozens of parameters and sophisticated 2D interfaces—automatically generated 3D user interfaces are often insufficient. To address this limitation, we designed a dedicated 3D GUI Editor, implemented as a web-based application. Several editing sessions demonstrating its functionality are available in an accompanying video[12].

The editor allows users to instantiate any existing WAM plugin by providing its URI. Upon loading, the plugin is initialized, and its 2D interface is rendered for reference; this interface remains fully functional. Once the plugin is active, two additional, synchronized views become available: (1) a top-down orthographic view for positioning and scaling 3D interface elements, and (2) an adjustable 3D perspective view that provides a real-time preview of the rendered 3D plugin (Figure 4). A contextual menu facilitates the placement of various elements in the top view, including controllers, text labels, and plugin input/output nodes —visually differentiated by color coding for audio and MIDI I/O. It also provides automatic alignment and centering tools for organizing groups of elements.

As in traditional GUI editors, selected objects can be repositioned, resized, and edited via a property panel. Users can add extruded 3D text, borders, and textures for annotation or stylistic purposes. Additionally, decals and icons can be applied to represent logos, labels, or interactive hotspots, giving a more distinguishable appearance. Figures 3 and 5 shows some 3D WAMs forming a music installation, in the WAM Jam Party collaborative, immersive host application. They are organized into a "plugin chain" and are easily identifiable.

[12]https://www.youtube.com/watch?v=4JOE3F579kI

The current version of the editor supports a variety of controller types, including sliders, rotary knobs, pads, toggle switches, XY pads, draggable meshes (e.g., cylinders, boxes, polygons), and user-defined geometries.

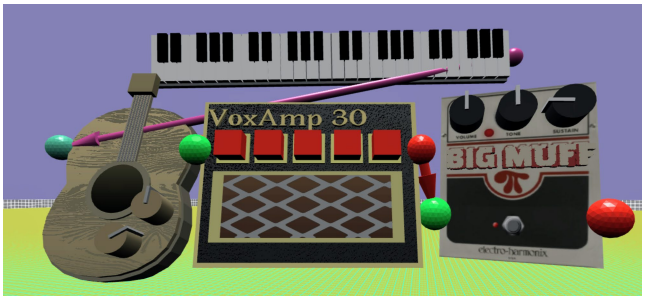

Figure 3: *A chain of identifiable WAM plugins: a MIDI keyboard, a guitar instrument, a guitar amplifier simulator and the Big Muff fuzz pedal WAM plugin already seen in Figure 1, this time with a dedicated GUI.*

Figure 4: *A reverb plugin named KVerb can be textured and get an identifiable look'n'feel.*

A flexible "body" model allows shaping the GUI into recognizable forms, such as a face, a guitar body, or a futuristic console, with support for textures like wood grain, metal, or glow shaders. Along our experiments, it appeared that it is possible to make plugins with a look'n'feel that could appeal to young children, this can be interesting for educational purposes. Figure 6 shows a FAUST WAM plugin that generates singing voices, rendered in 3D with the shape of a toy face, for example.

MIDI and audio inputs/outputs can be visually represented as connector spheres embedded in the 3D interface with color schemes to disinguish between audio and MIDI I/O. These connectors are designed to be interactively manipulated by users within the host application, allowing them to establish connections between 3D WAMs using VR controllers.

By allowing designers to customize the geometry, layout, colors, textures, and interaction elements of each plugin, the editor enables the emergence of strong visual identities across a modular VR environment. This results in rich, in-

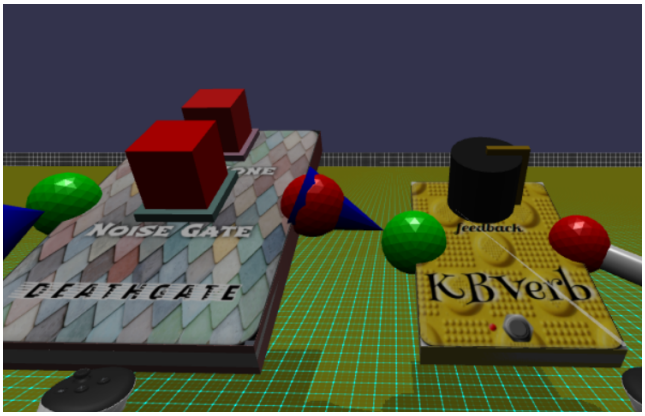

Figure 5: *Two audio effect pedals in the WAM Jam Party immersive host, including the Kverb WAM from the previous figure.*

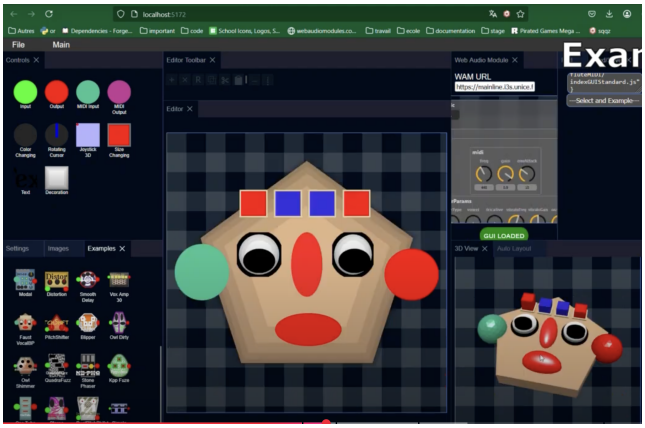

Figure 6: *The 3D GUI Editor interface displays multiple views: the original 2D WAM plugin is running in the top-right corner, a top-down layout view on the left, and an interactive 3D representation in the bottom-right. In this example, a WAM plugin developed in FAUST for choir synthesis is visualized using a form resembling a human face.*

teractive, and spatially coherent user interfaces that preserve the modular architecture of traditional plugin systems while fully embracing the affordances of immersive 3D environments.

3.5 Towards more personalized 3D GUIs

Some plugins, such as a piano roll, require even more interactive user interfaces: individual cells in the piano roll must be clickable to indicate which notes will be played; the playhead needs to be animated and synchronized; and during playback, active notes should be visually highlighted, among other behaviors (Figure 7).

Another example involves plugins that modulate the parameters of other plugins and require custom 3D animations. For instance, an “orbiter modulator” plugin features 3D spheres following Lissajous curves, with their XYZ positions used to modulate at sample rate parameters of other WAMs. An example of such a plugin, along with its associated 3D orbiters in motion, can be seen in the accompanying

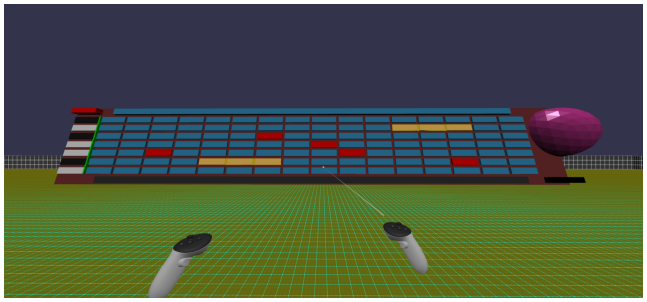

Figure 7: *A piano roll 3D WAM.*

video[13].

Yet another case is a 3D drum kit designed for interaction via gestures simulating virtual drumsticks, where striking the virtual drums generates MIDI notes and the drum elements are expected to visually respond to the impacts, depending on strike velocity and force. Custom code must be written to handle the physics engine[14], the interaction between the plugin’s logic and its 3D user interface. This WAM is meant to be connected to a drum sampler plugin for sound generation.

Figure 8: *A drum kit / MIDI note generator WAM with a dedicated 3D GUI loaded by custom code for animating its elements depending on the velocity (3D model file created with Blender). Here with a debug panel showing the measured velocity at impact and the generated MIDI events/notes.*

To allow such a level of customization, an API has been developed that extends the established plugin GUI default implementation. This API manages the interaction logic between the 3D GUI and the WAM core. The ”PianoRoll” WAM serves as a pertinent example of this API’s application. The interface designated for the 3D GUI facilitates creation, animation and event handling of its visual elements, such as rendering the piano roll notes and handling 3D-optimized controls like a scroll mechanism for note navigation (the piano roll displays only a fragment of the total virtual keyboard). At construction time, it receives a reference to the core component of the WAM, and can be synchronized with it at initialization and running time, managing its input/output configurations, even implement-

[13] https://youtu.be/G5ooMWxwn04

[14] babylonJS comes with the Havok physics engine used in many AAA video games.

ing some WAM extensions such as the "pattern extension" that allows loading and saving note sets, or the "note extension" that allows displaying specific note names in place of the traditional piano keyboard (when connected to a drum sampler, for example).

4. USING 3D WAMS IN THE WAM JAM PARTY HOST

4.1 Description

Since the early stages of this project, the development of 3D graphical user interfaces (GUIs) has progressed in parallel with the design and implementation of a 3D host environment: the real-time collaborative application WAM Jam Party. while the core concepts and system architecture have been detailed in previous work, the platform continues to evolve. For a comprehensive overview of its main features and interaction model, readers are referred to our earlier publication [6]. Accordingly, this paper provides only a brief summary to contextualize the integration of 3D-enabled Web Audio Modules (WAMs).

WAM Jam Party is a 3D Web host that uses Web Audio Modules (WAMs) in a multi-participant, real-time, virtual environment. It is part of the emerging field of the musical metaverse[18], an immersive virtual space dedicated to musical activities. The project aims to transpose the concepts of 2D applications such as Sequencer Party, a web-based platform for collaborative music creation [5], into an immersive, shared 3D universe.

The project is open source[15], available online[16] and current version can be seen in an accompanying video[17].

4.2 Typical sessions

In WAM Jam Party, users can join online sessions to collaboratively create interactive musical installations and perform live by interconnecting WAM plugins. Sessions are typically initiated by a primary organizer, who invites (by sharing an URI) a group of collaborators to contribute in real time within a shared, browser-based environment. All connected participants in a WAM Jam Party session share a common tempo and have the ability to start or stop the audio output of the collective musical environment. A typical music installation consists of a chain of interconnected plugins—note generators linked to instruments, which are then routed through audio effects and ultimately to a spatialized audio output. Once a complete subgraph is assembled, the resulting sound becomes audible to all participants, provided the host is in "play" mode.

4.3 State synchronization-based networking

Importantly, no audio or MIDI data is transmitted over the network. Instead, synchronization relies on sharing the internal states of each WAM (e.g., parameter values, piano roll patterns), the global host state (e.g., tempo), and the spatial states of other participants (e.g., position and orientation). This synchronization is based on a CRDT (Conflict-free Replicated Data Types) implementation using the Yjs JavaScript library. It uses a peer-to-peer (P2P) approach via WebRTC in its current version. Synchronization sends only incremental changes ("diffs") to optimize data exchange.

This approach ensures that all installations remain locally synchronized across users. For example, a drum sampler installation will stay rhythmically aligned with a bass generator, even though no audio/MIDI stream is exchanged.

In a typical session, participants begin by constructing their own musical installations. Once connected to an audio output, they enter a phase of "audio sculpting," adjusting parameters, toggling audio effects, and fine-tuning their setup. Users may also interact with installations created by other participants, fostering a collaborative and dynamic creative process (Figure 9).

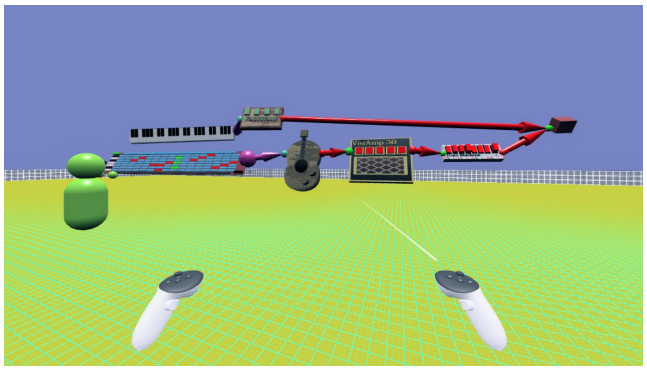

Figure 9: *Multiple participants build music installations collaboratively. We can see the avatar of another connected person in green.*

4.4 User interactions

Although WAM Jam Party was initially conceived as a collaborative platform for live electronic music performance, it has also demonstrated strong potential in educational settings—particularly for training students in music composition. It enables hands-on exploration of audio effects, modulation techniques, and digital sound design within a multi-user environment, making it well-suited for preparing group-based electronic performances and collaborative composition projects.

From an ergonomic standpoint, each WAM operates as an audio node that can be interconnected with others to form a dynamic audio graph. A key component provided by the host is a dedicated node representing a 3D spatialized audio output. Within WAM Jam Party, dozens of WAMs have been enhanced with interactive 3D graphical interfaces and made accessible to users. A "virtual plugin shop" (Figure 10) allows participants to browse a collection of available plugins, each represented by a 3D model. Once selected, a WAM can be positioned and rotated in 3D space, and connected to other nodes within the environment, allowing for flexible and intuitive construction of complex audio systems. For a detailed description of the various manipulation methods available for 3D elements, refer to [6].

WAM Jam Party is designed for easy extensibility through its host/plugin architecture. To add a new WAM plugin and make it accessible to all participants, one simply needs to place the corresponding 3D GUI configuration in a desig-

[15]https://github.com/doriangirard9/musical-multiverse-vr

[16]https://wamjamparty.i3s.univ-cotedazur.fr/, this URL is meant to be opened in a VR headset's web browser.

[17]https://www.youtube.com/watch?v=zZdZef6XB_I, (multi-participant VR host, version at the time of writing this paper.)

Figure 10: *A virtual shop with all available WAM 3D models displayed. Once picked, the shop disappear from the 3D scene of the current user, and the WAM can be positioned in space.*

nated folder, as detailed in Section 3. This typically involves a mapping file, with optional supporting JavaScript files.

4.5 Preliminary Performance Observations

Although formal performance testing is still pending, the core system architecture has remained stable since the previously published user evaluation in [6]. In that study, six participants connected under varied network conditions using Meta Quest 2 headsets, collaboratively instantiating up to 40 3D plugins. Frame rates remained smooth (above 40 FPS), with only minor slowdowns during floating menu interactions. While latency was not precisely measured, users reported no disruptive delays.

However, usability issues emerged with fully auto-generated GUIs: as already said, users struggled to distinguish between plugins due to visual uniformity and hard to read labels, and found the plugin selection process unclear in the 3D interface. These observations led to the development of two key features presented here: a 3D plugin "shop" for clearer module selection, and a custom 3D GUI editor enabling more expressive and identifiable plugin interfaces. While these addition with enhanced 3D representations could affect the frame rate, preliminary internal testing using Meta Quest 3 XE headsets —whose improved hardware enables better rendering performance— showed noticeably smoother interaction. This gain also reflects recent optimizations to the BabylonJS adapter, which now prioritizes efficient rendering through the generation of low-poly 3D meshes.

5. FUTURE ENHANCEMENTS

As future work, we aim to explore the design of virtual instruments specifically tailored for immersive environments, moving beyond the replication of traditional instruments toward forms that make use of the unique affordances of spatial computing. This includes developing 3D volumetric interfaces that support rich, embodied interaction—such as manipulating sound through gesture within a defined spatial volume—alongside multi-participant interaction, enabling collaborative performance in shared virtual spaces. Inspiration for this direction comes from iconic spatial instruments like Jean-Michel Jarre's virtual laser harp, where sound is triggered by intersecting invisible beams, and Jonathan Bell's virtual cello [3], which reimagines bowed string interaction through gestural and spatial metaphors within VR. These approaches open up new possibilities for expressive control, spatialization, and performance aesthetics in virtual music-making. Refining the balance between automated GUI generation and the need for distinct, expressive interfaces in XR environments seems important.

Future improvements to the host environment include enhancing its affordances, introducing a variety of customizable virtual spaces (as the current environment consists only of a minimal green playfield under a dark blue sky), and implementing support for larger user groups through an instance-based architecture—similar to multiplayer games—where users join dedicated rooms, each accommodating up to 16 participants, for example.

Furthermore, we plan to investigate the use of standard WebXR hand tracking as a complementary or alternative input method to traditional VR controllers. Given its support in BabylonJS, this integration could offer a natural and controller-free interaction model, particularly suited for lightweight or mobile XR setups. Also, to extend the range of gestural input and enhance expressive interaction with virtual instruments, additional motion-sensing devices—such as the SOM-1[18] sensor, that resemble a wristwatch and can send MIDI events over bluetooth, could be integrated alongside standard VR controllers to capture complementary movement data.

In addition to audio interaction, we also aim to enhance the immersive quality of the experience by incorporating real-time visual feedback during performance. This includes for example the animation of waveforms, frequency spectra, reactive 3D shapes synchronized with audio, video streams. The WAM standard already comes with a set of extensions [7] for 3D visualization, shaders, video streams, that could be adapted to immersive 3D, enabling mixed-media experiences with expanded visual interaction.

Moreover, we also aim to improve the logic of creation and modification of different WAMs, as demonstrated by the studies of Clément Quere [16]. Finding appropriate control methods to reduce articulatory distance is essential for sustained creative interaction. During extended musical creation sessions, users may experience physical fatigue, particularly in the arms and shoulders. To address these ergonomic challenges at the individual interaction level, we can envision body-anchored menus or WAM instantiation through predefined gestural interactions that minimize repetitive strain.

Building upon individual ergonomic considerations, effective system-level affordance design becomes crucial for guiding users regarding available interactions and limitations. For instance, the influence sphere of an audio output could be visualized through a dispersion dome or fallback indicators, providing clear spatial feedback about sound propagation. Proactive affordances can further enhance this guidance—when no audio output is detected while a user attempts to produce sound, the system could automatically suggest adding one, reducing cognitive load and maintaining creative flow.

Building on this system's intelligence, we could extend it

[18]https://instrumentsofthings.com/

into collaborative learning by integrating LLMs or AI agents that interact through the Model Context Protocol[19] (see [17] for a survey of its use). These agents would connect via an MCP server to access and control virtual instruments or audio effects in real time. For example, if the MCP server indicates that no instrument or effect parameters have been changed since initialization, the AI could gently intervene with visual suggestions or automated tweaks to encourage exploration. A concrete illustration of this approach has been explored by Yann Orlarey[20], who demonstrated how an LLM can control a polyphonic synthesizer written in FAUST through an MCP server.

At the collaborative interaction level, user representation becomes paramount—both from the user's own perspective and how they appear to others. Currently, users lack virtual "hands," seeing only VR controller representations. From an external viewpoint, they are represented by simplistic round avatars. We could envision more realistic avatars similar to those in Meta's Metaverse [21].

6. CONCLUSIONS

This work illustrates a web-native approach to building immersive, interactive audio systems by integrating Web Audio Modules plugins into real-time 3D multi-user environments. In contrast to immersive music platforms built on native applications and proprietary engines, the proposed system is developed entirely with standard web technologies —including Web Audio, WebGL, WebXR, and WebRTC— and requires no installation or platform-specific dependencies.

A key strength of this approach lies in its ability to reuse a large and growing ecosystem of existing WAM plugins, ranging from simple effects to complex synthesizers, without needing access to their source code. Any plugin can be instantiated via its URI, including high-fidelity emulations of classic, complex hardware : synthesizers such as the Prophet V or the Oberheim OB-Xd and a large palette of audio effects. These modules can be embedded in interactive 3D scenes with either automatically generated or manually crafted graphical interfaces.

The WAM Jam Party host platform demonstrates that advanced features —such as low-latency audio processing, spatialized interaction, physics-based control, and synchronous multi-user collaboration— can now be realized within an immersive, purely web-based system. To the best of our knowledge, this is the first work published to showcase Web Audio Modules operating within fully interactive 3D and XR applications on the web.

Future work will explore enhanced embodied interaction techniques, adaptive user interfaces, and intelligent assistance features. In addition, upcoming efforts will include performance evaluations focusing on CPU and GPU usage, rendering efficiency, and audio latency, along with extended user testing involving composers, musicians, and developers to assess usability, expressiveness, and creative potential in real-world contexts.

[19]MCP is an open standard designed to connect language models to external tools, data, and services, proposed by Antropic in 2024. See https://www.anthropic.com/news/model-context-protocol

[20]https://github.com/orlarey/McpUI

[21]https://www.meta.com/en/avatars/

Another important direction involves the design of instruments and audio effects specifically tailored for XR environments. Rather than replicating existing hardware paradigms, these tools will explore novel interaction models that take advantage of spatial input methods such as gesture tracking, volumetric sensors, and body-aware controls. The goal is to develop interfaces that are natively expressive in immersive settings, offering new affordances for creative performance and composition.

7. ACKNOWLEDGMENTS

This work was supported by the French government, through the France 2030 investment plan managed by the Agence Nationale de la Recherche, as part of the UCA DS4H project, reference ANR-17-EURE-0004.

8. REFERENCES

[1] V. Bauer and T. Bouchara. First steps towards augmented reality interactive electronic music production. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 90–93. IEEE, 2021.

[2] J. Bell. Networked music performance in PatchXR and FluCoMa. In *International Computer Music Conference (ICMC) 2023*, 2023.

[3] J. Bell and B. E. Carey. Animated notation, score distribution and AR-VR environments for spectral mimetic transfer in music composition. In *TENOR*, 2019.

[4] A. Boem and L. Turchet. Musical metaverse playgrounds: exploring the design of shared virtual sonic experiences on web browsers. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–9. IEEE, 2023.

[5] M. Buffa and T. Burns. Real-Time Collaborative Music Creation on the Web: exploiting Web Audio Modules for Interactive Performance and Composition. In *30th IEEE Symposium on Computers and Communications (ISCC), workshop Next-Generation Multimedia Services at the Edge (NGMSE): Leveraging 5G and Beyond*, Bologna, Italy, July 2025. IEEE,.

[6] M. Buffa, A. Hofr, and D. Girard. Using Web Audio Modules for Immersive Audio Collaboration in the Musical Metaverse. In *IS2 2024 - IEEE International Symposium on the Internet of Sounds 2024*, Erlangen, Germany, Sept. 2024.

[7] M. Buffa, S. Ren, T. Burns, A. Vidal-Mazuy, and S. Letz. Evolution of the web audio modules ecosystem. In *Web Audio Conference 2024*. Zenodo, 2024.

[8] M. Buffa, S. Ren, O. Campbell, T. Burns, S. Yi, J. Kleimola, and O. Larkin. Web audio modules 2.0: An open web audio plugin standard. In *Companion Proceedings of the Web Conference 2022*, pages 364–369, 2022.

[9] T. Carpentier. Binaural synthesis with the web audio api. In *1st Web Audio Conference (WAC)*, 2015.

[10] D. Dziwis. Pdxr-the evolution of pure data into the metaverse. In *Proceeding of the International Computer Music Conference (ICMC)*, pages 1–8, 2023.

[11] D. Dziwis, H. von Coler, and C. Porschmann. Live coding in the metaverse. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–8. IEEE, 2023.

[12] D. Dziwis, H. Von Coler, and C. Pörschmann. Orchestra: a toolbox for live music performances in a web-based metaverse. *Journal of the Audio Engineering Society*, 71(11):802–812, 2023.

[13] R. Hamilton. Coretet: a 21st century virtual interface for musical expression. In *14th International Symposium on Computer Music Multidisciplinary Research*, pages 1010–1021. Springer, 2019.

[14] J. Kleimola and O. Larkin. Web audio modules. In *Proc. 12th Sound and Music Computing Conference*, 2015.

[15] N. Polys and N. Bendelac. Spatial audio designer. In *Proceedings of the 27th International Conference on 3D Web Technology*, pages 1–4, 2022.

[16] C. Quere, A. Menin, R. Julien, H.-Y. Wu, and M. Winckler. Handynotes: Using the hands to create semantic representations of contextually aware real-world objects. In *IEEE VR 2024 - The 31st IEEE Conference on Virtual Reality and 3D User*, 2024.

[17] P. P. Ray. A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. *Authorea Preprints*, 2025.

[18] L. Turchet. Musical metaverse: vision, opportunities, and challenges. *Personal and Ubiquitous Computing*, 27(5):1811–1827, 2023.