Interactive Audio Sculpting: Plugin Customization and UI Affordances in Immersive Environments

Michel Buffa, Marco Winckler, Quentin Escobar, Samuel Demont, Ayoub Hofr, Adam Mir-Sadjadi
SPARKS Research Group, I3S Lab.
University Côte d'Azur
Sophia-Antipolis, France
firstname.lastname@univ-cotedazur.fr

***Abstract*—WAM Jam Party is a collaborative WebXR platform for creating 3D musical installations using Web Audio Modules (WAM) plugins. Expanding on earlier work, this paper introduces major improvements focused on usability, expressiveness, and interaction design. A new 3D GUI editor allows users to attach interactive, custom graphical interfaces to existing WAM plugins without modifying their source code, enabling representations such as synths with distinctive textures, labels, 3D knobs etc. or physics-based instruments like a guitar with a body shaped like a real acoustic guitar. The design offers a "3D Plugin Shop" and floating menus with preview thumbnail images as alternative means to select and assemble audio components. The system allows dynamic code injection for advanced behaviors, including animated parameter modulators, piano roll sequencers or an animated drum kit. User studies report higher engagement, faster onboarding, and creative workflows. This study also gives some practical insights for building collaborative audio tools in immersive web-based environments.**

***Index Terms*—Musical Metaverse, WebXR WebAudio, WebMIDI, Web Audio Plugins, Web standards, Web Audio Modules, CRDT, Collaborative Experiences**

I. Introduction

The convergence of Web Audio, immersive 3D environments, and real-time collaboration opens up new paradigms for musical creation and interaction. As browser-based technologies mature, most notably the Web Audio API, WebXR, and emerging standards such as Web Audio Modules (WAM, web audio plugins for the web [15], [16]), we are witnessing a fundamental shift in how musical experiences can be authored, shared, and performed on the web. In this context, the Musical Metaverse is not merely a speculative vision, but a tangible, evolving ecosystem in which music technologies are spatialized, embodied, and collaborative by design [36].

This paper introduces WAM Jam Party, a collaborative WebXR platform that allows users to create and interact with 3D musical installations built using WAM plugins. These installations resemble setups typically found in a music studio, consisting of MIDI note generators, virtual instruments, and audio effects connected together, each represented by a WAM plugin. To make the sound audible within the virtual environment, the audio graph ends with spatialized (binaural) virtual speakers that serve as the system's output (Figure 1).

This work aligns strongly with the foundational themes of the Internet of Sounds conference, specifically, networked audio systems, multi-user collaboration, web-based infrastructures, and interactive audio in distributed environments [37]. One of the distinctive aspects of WAM Jam Party is that *everything runs entirely within the web browser of the VR headset*, including full WebAssembly recreations of instruments like the Prophet V and Oberheim OB-Xd synthesizers, complex audio effects, and MIDI sequencers, all processed locally on the immersive device. *Multi-user collaboration is also seamless*, as participants can join simply by entering the WAM Jam Party URL in their browser or scanning a QR code with any XR-capable device.

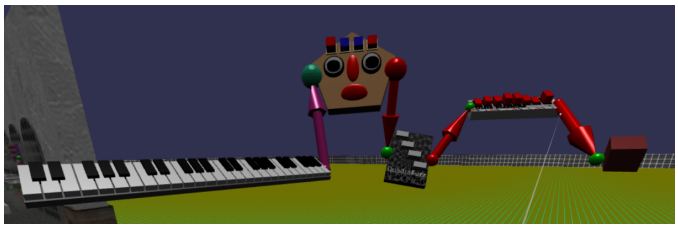

Fig. 1. A typical Music Installation graph: a note generator (here a virtual MIDI keyboard) connected to an instrument (a synthetic voice generator), then to some effects and finally to a spatialized audio output (the red cube on the right).

Earlier phases of the WAM Jam Party project demonstrated the feasibility of these distributed audio graphs and their manipulation and synchronization across WebXR clients [12]. However, those early efforts mainly focused on building the core architecture needed for a stable multi-user system. The 3D interfaces were basic and automatically generated from plugin metadata, which often made it hard to tell plugins apart or interact with them in a meaningful way.

This new version of WAM Jam Party introduces a 3D graphical editor that brings the focus to expressive interaction and immersive design, allowing users to create fully customized, more recognizable, plugin interfaces with unique 3D visuals, animations, and physics-driven behaviors.

Additionally, the system's architecture now supports injecting custom code into plugin interfaces, enabling rich 3D controls like animated modulators and sequencers that enhance visual feedback and interactivity. This code can also interface with a 3D physics engine—such as Havok, used in many AAA games. For example, a virtual drum kit was implemented as

a MIDI note generator, allowing users to strike drums with virtual sticks and see realistic physical responses. This marks an initial step toward embedding more embodied instruments and interactive interfaces into the platform. A new 3D Plugin Shop complements these features by presenting plugins as 3D objects on virtual shelves, offering a more intuitive and engaging alternative to earlier floating text-based menus.

The remainder of this paper is organized as follows: Section II reviews related work on modular XR music systems, with a focus on WebXR-based and multi-user environments, as well as research on virtual instruments. Section III presents the WAM Jam Party system architecture and implementation details. Section IV describes the new features introduced since our previous publication, including the 3D GUI editor, Plugin Shop, and scripting support. Section VI reports on our user study and evaluation, and Section VI discusses broader implications and future directions.

II. Related Work

A. Modular and Embodied Instruments in XR

XR musical instruments have emerged in waves: early HMD/data-glove performances in the 1990s [26], CAVE/stereoscopic systems in the 2000s [28], [30], and a return to affordable HMDs in the 2010s, such as Hamilton's *Coretet* that recreates bowed string quartet performance in VR [24], or Bell and Carey's AR/VR who developed cello animated notation and spatial cello interaction for AR/VR settings [3]. Boem et al. [6] introduced the concept of networked Virtual Musical Instruments (VMIs) as a framework for the Musical Metaverse. The following authors report comprehensive surveys of musical instruments in VR [33] [36] [23].

Immersive applications may also adopt a modular scene-based approach. Wakefield and al. [38] and Çamcı and al. [18] analyzes VR-specific affordances for modular music and shows how modular synthesis practices translate into VR, identifying spatial patching, embodied manipulations, and constraints such as occlusion/scale. More recently, *PatchXR*[1] [2], *SynthVR*[2], and *SynthSpace*[3] allow users to assemble instruments, sequencers, effects, and spatialized audio outputs as audio graphs in 3D. *Virtuoso*[4] offers five gesture-friendly instruments, including a volumetric theremin-like controller and drum pads playable with virtual sticks. Both *Virtuoso* and *PatchXR* map notes to predefined scales, helping users create harmonically coherent performances. A similar approach is seen in the 2D mobile synthesizer *BeBot*[5], which maps finger position to pitch within user-defined scales, offering accessibility and expressive potential.

[1]https://patchxr.com/

[2]https://www.meta.com/fr-fr/experiences/synthvr/3748465338566486/

[3]https://store.steampowered.com/app/1355640/SYNTHSPACE/

[4]https://www.meta.com/experiences/virtuoso/4705981139481778

[5]http://www.normalware.com/

B. Plugin Architectures and MIDI support

Most of existing immersive music systems are native applications (built with Unity and Unreal) that quite often embed monolithic audio pipelines, making it difficult to integrate external plugins or community-authored tools. There is little support for dynamic loading, third-party reuse, or standard plugin formats[6]. Integrating 2D plugin standards like VST requires significant effort, particularly for 3D GUI adaptation.

WAM Jam Party takes a different approach by adopting Web Audio Modules (WAMs) [15], [16], a standard plugin architecture for the Web. WAMs allow for modular composition, GUI separation, and dynamic loading using URIs. More than 100 plugins are currently available, mainly through the WAM-Community initiative[7], many compiled from FAUST or CMajor DSLs to WebAssembly. The system uses the Web Audio and Web MIDI APIs to handle sequencing and control locally, avoiding MIDI-over-network challenges [5]. This is especially useful for standalone headsets, where MIDI support is limited unless tethered to a host machine, as stated by Boem et al. [7]. In this paper, titled "Issues and Challenges of Audio Technologies for the Musical Metaverse", they note that despite advances in 5G and Wi-Fi technologies, real-time audio and MIDI streaming still lack the reliability required for precise musical interaction. WAM Jam Party addresses this challenge by using local plugin clocks combined with state-based synchronization, providing a more stable and portable alternative for collaborative music making.

C. Collaborative and Distributed Musical XR Systems

Projects like *Sequencer Party* [10] and *Orchestra* [21] allow users to perform together over the Web using shared virtual spaces. *Musical Metaverse Playgrounds* [8] explore sonic co-creation in social XR settings. Live coding platforms such as *Estuary* [29], *Glicol* [25], as well as [32] and [20] also rely on local clock synchronization and shared actions (e.g., triggering notes or updating parameters) across the network.

WAM Jam Party employs a state-synchronization approach, augmented with advanced change-tracking mechanisms similar to those used in collaborative editors like Google Docs. Each participant runs a local clock, while only control data—such as sequencer patterns, parameter changes, plugin positions, orientations, interconnections, and host tempo—is exchanged over the network. This architecture enhances robustness in bandwidth-constrained or mobile environments and aligns well with MIDI-based instruments and control-driven workflows. Although real-time audio streaming is not supported—aside from limited scenarios like synchronized sample playback—this design is particularly well suited to electronic music practices, where sequenced and parameter-driven interactions are more typical than low-latency audio exchange.

[6]Note that Unity provides its own SDK for developing custom audio plugins, but integrating existing 2D audio plugin standards, such as VSTs, remains a complex process: https://docs.unity3d.com/Manual/AudioMixerNativeAudioPlugin.html

[7]https://www.webaudiomodules.com/docs/community

More recently, CSound Meta [27] and PdXR [19] have explored immersive, multi-user music systems. CSound Meta can be used within native Unity applications to embed Csound "sound objects" in collaborative VR scenes, while PdXR maps Pure Data patches into A-Frame/WebXR environments with networked 3D GUIs. In contrast, WAM Jam Party is a web host based on the open Web Audio Modules (WAM) standard, supporting dynamic loading of diverse instruments and effects directly in the browser. Moreover, our 3D GUI editor (section IV-C and CRDT-based state synchronization III distinguish it from these pipelines, offering a more general and extensible framework for collaborative musical creation on the Web.

D. Interaction Design and Immersive Usability

Current immersive technology allows to create high quality sound experiences and custom sound placement in the 3D environments. For example, Quere et al. [31] and Berthaut [4] investigate the use of 3D interaction techniques for information placement and musical expression.

WAM Jam Party applies gesture-based control and spatial layout to support expressive and discoverable interaction within collaborative audio workflows in 3D environments. Interaction features, such as deletion gestures and joystick remapping, are designed to align with users' expectations from gaming and performance contexts.

Research into music education and collaborative performance using immersive environments (e.g., PatchWorld and Fortnite) also shows that presence and engagement depend strongly on the modality (VR/XR/AR) used [9]. Projects like *Holodeck* [22] and remote orchestral conducting setups [34] further illustrate the potential of XR for streaming, gesture capture, and distributed rehearsals.

III. System Overview

A. Layer based architecture

The architecture of WAM Jam Party is organized into three modular layers, each responsible for a specific set of functionalities: immersive 3D rendering and interaction, audio/MIDI rendering, and network synchronization. This separation into layers allows the system to clearly decouple 3D interaction, audio rendering, and network communication, making it easier to develop, maintain, and extend each part independently.

B. 3D Interaction and Scene Management layer

The first layer handles the rendering and interaction logic of the immersive environment. It is responsible for visualizing the 3D scene using Babylon.js, including the graphical representations of all WAM plugin instances and their 3D user interfaces. Through the WebXR API, supported by Babylon.js, users can interact with objects in the scene by selecting, positioning, rotating, or connecting them, using VR controllers or hand tracking. Two complementary mechanisms are provided for instantiating plugins and building musical installations. The first is a system of dynamically generated floating menus, built using the MRTK Toolkit[8], which also enables less frequent operations such as controlling transport events sent to WAM plugins (e.g., play, pause, tempo, record) similar to what digital audio workstations (DAWs) do with native plugins. The second is the 3D Plugin Shop, a spatial interface that presents available plugins as 3D objects on virtual shelves, allowing users to browse and select components directly in the scene.

Once a plugin is added, users can adjust its parameters through 3D GUIs that are either automatically generated from metadata or fully customized. Interactions on these GUIs trigger updates in the underlying audio layer to ensure that visual changes are reflected sonically.

C. Audio and MIDI Rendering layer

The second layer manages the instantiation and execution of audio and MIDI logic. Each WAM plugin is identified by a unique URI and can be used either with its original 2D interface (based on HTML/CSS/JavaScript) or in headless mode. In WAM Jam Party, plugins are loaded in headless mode, and a 3D user interface is generated or attached dynamically. Each plugin is associated with a JSON configuration file that defines its URI, parameter structure, default values, and intended 3D representation. This configuration may also include optional JavaScript or TypeScript files to enable custom interface behaviors and interactive logic.

This layer also supports the management of plugins through a dedicated folder within the web application's directory structure. This folder contains one configuration file per WAM plugin, each describing how the plugin should be loaded and represented. When the application is launched, the contents of this folder are parsed at runtime, and the available plugins are reflected in both the floating menus and the 3D Plugin Shop. In future versions, a resynchronization feature or periodic automatic scanning could be introduced to detect newly added or updated plugins without requiring a page reload.

D. Network Synchronization and Collaboration layer

The third layer manages multi-user synchronization, ensuring consistency across distributed sessions. Rather than streaming audio or MIDI data between participants, the system implements a state-based synchronization model. All client-side audio engines run locally; only structural and parameter changes in the shared scene are broadcast across clients. Changes are tracked and compared at regular intervals, and only delta updates (diffs) are transmitted over the network. Each event is timestamped, and a history of changes is maintained to support disconnection and reconnection scenarios without data loss. This approach significantly reduces bandwidth requirements, improves resilience on unstable connections, and aligns with best practices for networked music systems operating under limited or variable latency conditions.

E. Implementation details and performances

The first description of the WAM Jam party reports its core architecture [12]. The current description of the system focuses on the user interaction and the integration of custom 3D GUIs for WAM plugins, MIDI note generators, and instruments.

[8]https://doc.babylonjs.com/features/featuresDeepDive/gui/mrtk/

WAM Jam Party uses Babylon.js for 3D rendering and WebXR integration, the MRTK v2 toolkit for spatial UI components, and Havok Physics (in an experimental repository branch) for embodied instruments. WAM Jam Party is open source[9], the GitHub repository includes a Docker configuration that makes it straightforward to build and deploy for web developers with basic experience. However, for ease of access, a hosted version is available at https://wamjamparty.i3s.univ-cotedazur.fr/, where users are immediately placed into a shared VR session upon visiting the site. Section VI outlines future plans to replace the current direct entry point with a dedicated landing page, featuring user and session management functionalities inspired by Sequencer Party [10], which originally served as a foundation for WAM Jam Party.

The network layer is based on a Conflict-free Replicated Data Types (CRDT) [35] algorithm, implemented using the Yjs JavaScript library[10] in WebRTC mode, while a websocket-based configuration can also be used. Yjs handles real-time state synchronization by maintaining a full history of changes and transmitting only diffs when state updates occur[11]. Session persistence is handled via JSON snapshots, rather than complete history logs.

Most WAM plugins are not bundled with the system but are dynamically imported by the host using TypeScript/JavaScript dynamic imports. Each plugin is referenced by a URI through its configuration file, which may also include optional custom GUI code, all stored in a dedicated configuration folder.

Performances: Although formal performance benchmarks are still forthcoming, the core system architecture has remained stable since the previously reported user evaluation in 2024 [12]. In that study, six participants connected under varying network conditions using Meta Quest 2 headsets with the default web browser (Chrome) and collaboratively instantiated up to 40 3D plugins. The system maintained a consistent frame rate above 40 FPS, with only minor slowdowns observed during floating menu interactions. Although latency was not quantitatively measured, participants reported no noticeable delays affecting usability.

In the current version, the 3D GUIs of WAM plugins have become significantly more advanced. Although this added complexity could potentially impact performance, internal tests using Meta Quest 3 headsets, benefiting from improved hardware, demonstrated smoother interactions in general (above 70 FPS). These improvements also came from recent optimizations in the Babylon.js adapter, which now emphasizes efficient rendering by generating low-polygon 3D meshes.

However, more rigorous performance evaluations are planned for the near future to validate the system's behavior under greater load, particularly when multiple instances of plugins with complex 3D GUIs, such as the 3D piano roll presented in Section IV-C, are used simultaneously.

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

[10]https://github.com/yjs/yjs

[11]https://blog.kevinjahns.de/are-crdts-suitable-for-shared-editing/ gives more insight about the performance characteristics in Yjs.

IV. New Features and Enhancements

A. Toward more stability and usability

Anyone can join a session simply by opening the URL of WAM Jam Party in the browser of a VR headset or using a standard web browser with a WebXR extension, though the latter may offer a less optimal experience. The system is fully compliant WAM host. Most plugins used in the platform have their DSP cores compiled to WebAssembly, typically from audio-specific DSLs like FAUST or CMajor, and are also released as open source.

Current development efforts have focused on four main areas: (i) implementing full MIDI support through a range of note generators and virtual instruments, (ii) developing a 3D editor and an accompanying API for building custom 3D graphical interfaces, (iii) creating a 3D plugin shop to improve the discovery and selection of plugins for assembling musical installations, and (iv) exploring advanced behaviors such as parameter modulators with animated 3D visuals and embodied instruments like a drum kit driven by a physics engine.

B. Full featured instruments and MIDI support

This new version introduces MIDI support through a collection of dedicated MIDI note generator WAMs, including a step sequencer, piano roll, random note generator, and a Turing machine–inspired generator that creates melodies based on user-defined scales, modes, octaves, and pitch ranges. These generators can be freely connected to instrument WAMs to produce sound within modular audio graphs.Timing is handled locally on each client: all WAMs within a user's session share a common local MIDI clock, ensuring tight synchronization between note generators and virtual instruments.

The platform includes a wide range of virtual instruments, such as high-fidelity recreations of analog synthesizers like the Prophet V and Oberheim OB-Xd, as well as instruments with physics-based modelization including flute, clarinet, maracas, djembe, guitar, and electric piano. Many of these are compiled from FAUST to WebAssembly, allowing compact, efficient deployment directly in the browser. A General MIDI–compatible synthesizer is also available, weighing just 24kB, as well as the SPECTRUM[12], a synthesizer based on popular eurorack modules made by Mutable Instruments.

In addition, several sample-based instruments are provided, including a general-purpose sampler that integrates with the FreeSound database, allowing users to search for samples online and automatically generate playable note sets by adjusting the playback rate to match the pitches of any chosen scale or mode [15]; a dedicated drum sampler; and a looper capable of recording audio from the headset's microphone. The platform is further enriched by a large suite of audio effects [11], [13], [17], including advanced modules such as a FAUST port of the Eventide Blackhole reverb and distortions/tube guitar amplifier simulations, including a faithful model of a Marshall JCM 800 amp [14].

[12]The SPECTRUM WAM plugin is a WebAssembly port of an iOS synthesizer available at https://apps.apple.com/us/app/spectrum-synthesizer-bundle/id1467384251

C. 3D GUI Editor for WAM plugins and API for custom code injection

User feedback from earlier versions of WAM Jam Party [12] highlighted a significant usability issue: the 3D GUIs for plugins were automatically generated from parameter metadata, leading to visually uniform controls that made it difficult to distinguish between plugins during interaction (Figure 2).

Fig. 2. Auto-generated 3D GUI compared to a custom 3D GUI made with the new 3D editor.

To address this, the latest version shifts the emphasis toward a more expressive 3D GUI design workflow. At the core of this enhancement is a newly introduced 3D graphical editor, available online[13], which allows users to build fully customized and visually distinctive interfaces for WAM plugins, without requiring access to their original source code (only the WAM plugin URI is sufficient), see Figures 3 and 4. These new 3D GUIs can include personalized meshes, textures, labels, borders, animation behaviors, interaction logic, and optional physics-based dynamics, significantly improving plugin recognition, usability, and overall immersion. A demonstration of the editor's flexibility is shown in the accompanying video[14], where it is used to create, for example, a vocal synthesizer shaped like a human head, or a guitar plugin with a curved hollow body and neck modeled in 3D. More than 40 plugins with custom 3D GUIs have been created (Figure 4) and are included in the editor as ready-to-use examples.

In addition, the system architecture has been redesigned to support the integration of custom JavaScript or TypeScript code directly within the plugin interfaces *without requiring access to the original 2D plugin source code*. This makes it possible to build highly personalized 3D controls, with custom interaction logic, animations, and visual elements, enabling more dynamic and responsive user interfaces around existing WAM plugins. For example, developers can build 3D modulation orbiters: animated spheres that move along curves in space and visually control other WAM plugin parameters at sample rate, like a filter cutoff in a synthesizer, see the accompanying video[15].

[13]https://jempasam.github.io/3d_wam_editor/app/

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

[15]Video of a parameter modulator WAM using 2D orbiters: https://youtu.be/uvhjJ7NzIUA, and here is a video of its 3D port in WAM Jam Party: https://youtu.be/G5ooMWxwn04

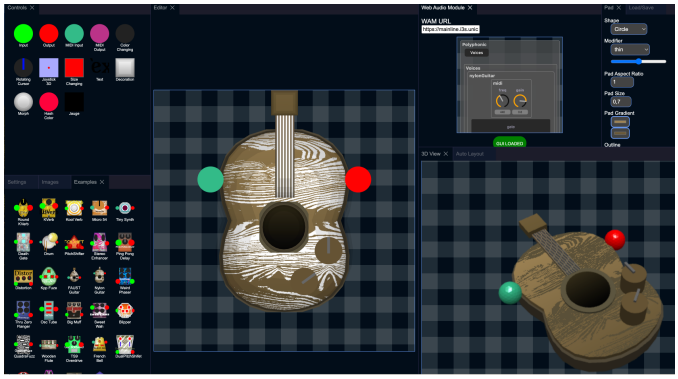

Fig. 3. The new 3D GUI editor. Start by entering the URI of an existing WAM plugin, then shape its 3D GUI, map controllers to the WAM parameters, export as a JSON configuration file.

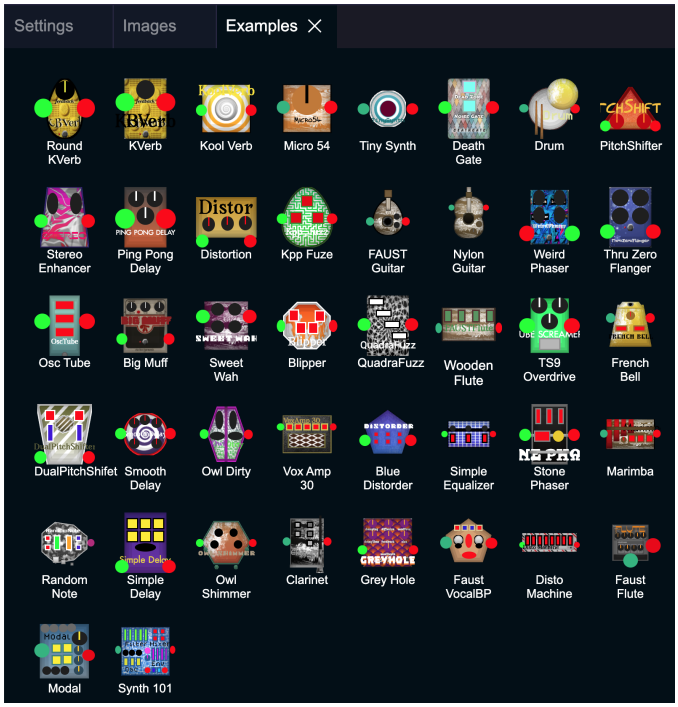

Fig. 4. About 40 WAM instrument and effect plugins with a 3D GUI have been created so far and are included in the editor as ready-to-use examples.

One notable example of a WAM requiring a fully custom 3D interface is the Piano Roll 3D sequencer. Built on top of Tom Burns' original 2D Piano Roll WAM (running in headless mode), this version faithfully reproduces its 2D behavior in a fully interactive 3D environment. The interface features a vertical 88-key piano keyboard and a horizontal note grid: blue cells represent inactive notes, red marks single-click active notes, and purple indicates long notes. A green playhead moves across the grid in sync with MIDI playback or recording, visually aligned with the Web Audio API's high-resolution AudioContext time. See the accompanying video[16].

User interaction is mainly based on pointer and ray-based

[16]https://www.youtube.com/watch?v=O4c1RaNclXo

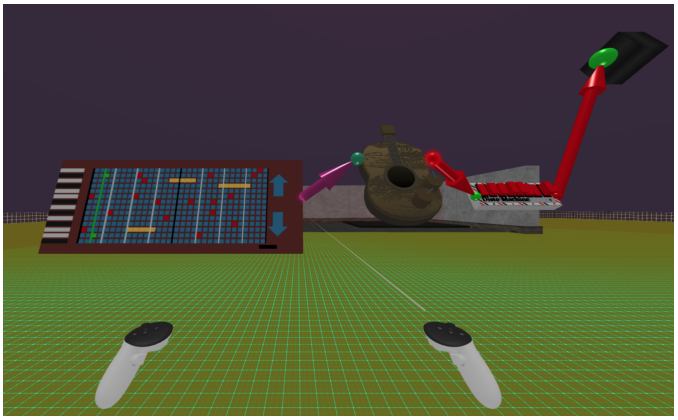

Fig. 5. A Custom 3D GUI example: the 3D piano roll sequencer.

input. Clicking cells toggles notes, while holding the X button enables long-note creation with adjustable yellow borders for resizing or deletion. The 3D GUI acts as an adapter to the original plugin, sharing context, accessing its API, and dispatching events, illustrating a robust integration model that aligns well with the WAM architecture. When the piano roll receives transport events from the host (e.g., when the tempo is changed or playback/recording is started), the custom code in the 3D GUI first relays these events to the original WAM instance (which handles MIDI playback and recording). It then calculates constants from the tempo `bpm` and the 3D grid configuration (e.g., mesh width/height, number of displayed bars, time signature), to perform accurate 3D animation of the playhead and to visually highlight notes during playback or recording. This method ensures real-time synchronization between the visual playback and the audio output (provided the piano roll is connected to a WAM instrument).

D. 3D Plugin Shop

Another key innovation introduced in this work is the *3D Plugin Shop*, a dedicated spatial interface within the VR environment where users navigate and interact, see Figure 6. In earlier versions of the system, selecting plugins from a large library was cumbersome and unintuitive, relying on floating textual menus that lacked spatial affordance and visual clarity [12]. Plugins are now shown as unique 3D objects placed on virtual shelves in the 3D plugin shop, making it easy for users to browse, choose, and connect them by simply interacting in the scene.

This immersive system helps users find and connect audio components directly in the 3D space, making the process more playful and hands-on. It works a bit like setting up gear in a music studio: you can pick a note generator, connect it to a virtual instrument, add audio effects, and send the sound to a 3D speaker in the scene. Just like in a physical studio, where you might connect a MIDI keyboard to a synth module, then to a mixer with effects, and finally to speakers in the room, this system lets you build and hear your setup in a virtual space.

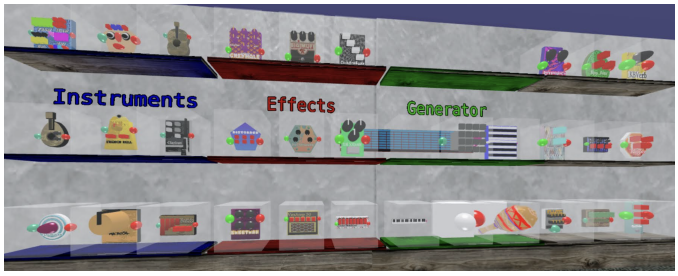

Fig. 6. 3D plugin shop: WAM intruments, note generators or effects can be clicked and dragged, then assembled into music installations.

E. Reworked floating menus

In earlier versions of WAM Jam Party, plugin selection was handled through floating menus built using MRTK v2 for Babylon.js[17]. User feedback indicated that these menus were difficult to read and cumbersome to navigate [12], particularly due to nested submenus, poor readability, and unintuitive backtracking mechanisms. To address this, the current version introduces a redesigned menu system featuring preview thumbnails, icons, and an improved hierarchical structure, enhancing usability and visual clarity, see Figure 7. Unlike the previous static implementation, these menus are now dynamically generated at runtime, based on plugin configuration files located in the application's plugins directory. This makes the system more flexible and easily extensible: new plugins can be added or updated without requiring changes to the interface logic.

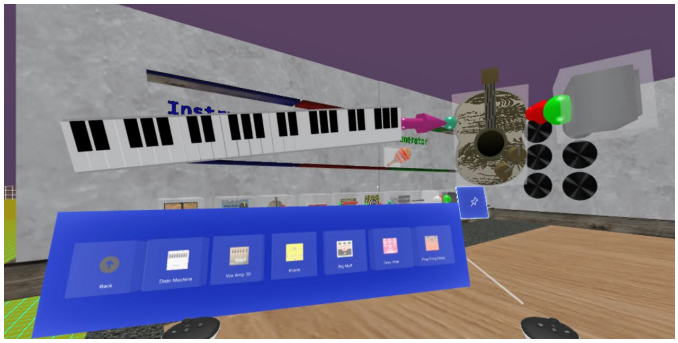

Fig. 7. New floating menus include thumbnails/icons of the proposed WAM plugins.

While the 3D Plugin Shop now serves as the primary, immersive method for browsing and selecting plugins, presenting them as tangible, spatially arranged 3D objects, these updated floating menus continue to play an important complementary role, particularly for quick filtering, precise search, or access in contexts where spatial navigation may be less efficient (e.g., seated VR use or 2D browser mode for debugging sessions). Together, these two interaction modes —spatial and symbolic— ensure that users can choose the most suitable interface for their workflow, balancing immersive exploration with practical control and accessibility.

[17]https://doc.babylonjs.com/features/featuresDeepDive/gui/mrtk

F. Towards embodied instruments

While most instruments and effects in the current system are 3D adaptations of existing 2D WAM plugins, recent developments in experimental branches of the source code repository, have begun exploring more embodied instruments. One such example is a fully modeled drum kit that closely resembles a real-world setup and can be played using hand gestures or virtual drumsticks attached to the 3D controllers visible in the immersive scene. The kit reacts dynamically to user input thanks to integration with the Havok physics engine[18]: toms vibrate on impact, and cymbals tilt and spin based on the hit position and applied force. It also produces haptic feedback through the VR controllers, enhancing the physical realism of interaction by simulating tactile response on each strike.

This drum kit is implemented as a WAM plugin with MIDI input and output, making it compatible with other MIDI generators such as a piano roll or step sequencer. It can be used both for recording timed sequences of drum patterns and for replaying them with synchronized physical animation.

G. Minor interaction enhancements

Several usability enhancements have been implemented to improve interaction within the 3D scene. Most notably, a gesture-based deletion mechanism allows users to remove plugins by selecting their 3D mesh and performing a shaking motion with the controller. This triggers a red halo and a circular progress indicator around the object, providing visual feedback and a brief confirmation delay to avoid accidental deletion. Users can now also move freely while holding a selected object, reducing the need to constantly drop and reselect items, and enabling smoother placement workflows. Finally, joystick controls have been made configurable, allowing users to swap the roles of the left and right joysticks, aligning with familiar gamepad conventions where the left stick handles movement and the right adjusts the view.

H. BabylonJS audio engine version 2

The new version 2 of the BabylonJS Audio Engine was integrated, providing native support for any Web Audio node as an input source. This eliminates the need for low-level handling when inserting WAM chains into the existing audio graph.

V. User study and Evaluation

In order to identify the limitations and drawbacks of the system, we conducted *a formative evaluation* aimed at detecting usability problems that should be addressed during the design and development of the tools. The formative evaluation was implemented as a user study with 6 participants that were invited to interact with the tool and complete basic tasks such as choosing plugins, placing them in the 3D space, connecting them to create a music installation, producing sound, and adjusting the parameters of an existing installation. A recurrent problem when creating immersive installation is to decide where to place the assets and menus in the virtual scenario. To explore interaction design alternatives for plugin selection, two methods were tested: (i) a floating menu accessible from anywhere in the scene, displaying thumbnail images of WAM plugins (Figure 7), and (ii) a virtual shop where users can browse and select 3D plugins directly from shelves (Figure 6).

[18]Havok is used by many AAA video games, and is now included in Babylon.js, see https://doc.babylonjs.com/features/featuresDeepDive/physics/havokPlugin.

A. Measures and Methods

The user study was conducted as a trial of the tool. During the session, data was collected including audio recordings of participant comments, task completion times, user preferences, and demographic information. A questionnaire captured the backgrounds of the participants and their experience with virtual reality and music. Task difficulty was rated using a 5-point Likert scale and overall usability was assessed through the System Usability Scale (SUS). For the qualitative analysis, we relied on participants' comments, collected through the use of a thinking aloud protocol during the sessions and the subsequent debriefing.

B. Tasks

The tasks presented during the trial consisted of a sequence of actions aimed at completing a scenario whose ultimate goal was to build and play a musical installation. Participants were instructed to use both the floating menu (condition "menu") and the virtual shop (condition "shop") across different tasks. The order of these two interaction methods was alternated and randomized across participants: those assigned even numbers began with the floating menu, followed by the virtual shop, while odd-numbered participants followed the reverse order. The list of tasks included the following:

- **T1** *Place a 3D guitar instrument into the scene using the menu / shop.*
- **T2** *Place a 3D MIDI piano keyboard into the scene using the shop / menu.*
- **T3** *Place a spatialized audio output into the scene using the method of your choice.*
- **T4** *Connect the 3D piano keyboard to the guitar.*
- **T5** *Connect the guitar to the audio output and play some notes with the piano keyboard, using the controllers.*
- **T6** *Modify the volume parameter of the guitar.*
- **T7** *Move every element in a way you like.*
- **T8** *Delete every element you added to the scene.*
- **T9** *Recreate an installation using this time the piano roll sequencer, a guitar instrument and a distortion effect named "disto machine".*
- **T10** *Free will, you are free to experiment with every proposed note generator, instrument or effect.*

Following the completion of each task, participants were asked to answer the following questions:

- **Q1** *Rate the difficulty of this task on a scale of 1 to 5.*
- **Q2** *How could you make this task easier ?*

C. Protocol

The experiments were conducted in a controlled environment within a space of approximately 2 square meters. Participants were seated in a chair, allowing them to rotate as needed while completing the tasks. The setup was designed to ensure participant safety throughout the trial. To minimize the effects of VR fatigue, each session lasted no longer than 30 minutes, and participants were free to take breaks at any time. They were also informed that they could stop the test at any point if they felt uncomfortable or experienced cybersickness during the trial.

Before the experiment, participants signed a consent form authorizing data collection and were informed about study conditions, including their right to withdraw at any time. They then completed a demographic questionnaire. Additionally, if a participant, despite not previously reporting any susceptibility, experienced motion sickness during the experiment, the session was immediately stopped and their participation withdrawn, along with any data collected up to that point.

The experimenter prepared the equipment and introduced the study by demonstrating a tutorial scenario. This tutorial covered essential interactions, including player movement, plugin creation, object manipulation within the 3D environment, plugin removal, and parameter adjustments. During this phase, participants were also introduced to the VR headset and virtual environment, with explanations of the controller buttons and available interaction methods. After assisting participants in putting on the VR equipment, the experimenter launched the immersive experience, and the participant began the scenario. Following the trial, participants completed the SUS questionnaire and were asked to identify three aspects they liked and disliked about the experience, along with any additional comments or suggestions. The experiment concluded with the open-ended question: *If you had to give a purpose for the application, what would it be?*

To preserve data anonymity, no personally identifiable information was collected. Each participant was assigned a unique identifier, allowing their responses across questionnaires and system logs to be linked without revealing their identity.

D. Participants

Using a convenience sampling method, six volunteers (one female), aged between 20 and 54, were recruited to participate in the experiment. Most participants were researchers, including PhD students and research engineers, as well as a musician and a software engineer. All participants were native French speakers.

Participants reported having normal or corrected-to-normal vision. They also indicated that they had only rarely, if ever, experienced motion sickness. In terms of VR experience, all participants had used virtual reality before, some frequently, others occasionally, none were complete novices. The most common reason for VR use was work-related tasks, followed by gaming. When asked about their experience with 3D video games, most participants reported regular usage, while two indicated they played only rarely. Regarding musical background, three participants reported playing a musical instrument, with skill levels ranging from intermediate to advanced.

E. Results

Completion Time and Accuracy There were no differences in accuracy, each participant successfully completed each task without failure.

Regarding completion time (i.e. the time between starting a task and providing an answer), participants took an average of 11 minutes to complete the entire scenario. Tasks completed using floating menus generally resulted in faster performance compared to the 3D shop, with an average of 6.5 to 12 seconds for tasks **T1, T2**. The final task, **T10**, that consisted in free experimentation, lasted an average of 4 minutes and 51 seconds. This task was particularly appreciated by the musicians, who engaged deeply with the system to create interesting musical results, sometimes spending up to 7 minutes before completion.

SUS The overall System Usability Scale (SUS) score obtained from the responses of the participants was 68.75, which, according to Bangor et ali. [1], falls within the range considered ”acceptable” to ”good”. t is interesting to note that participants with regular VR experience (P1, P2, P4) achieved higher SUS scores (85–92.5) compared to occasional users. = *Task Difficulty* The perceived difficulty of the tasks ranged from very easy to hard, with this variation largely explained by participants' backgrounds. Those who work with VR on a daily basis consistently rated the scenario steps as easy or very easy. In contrast, participants less familiar with VR reported task difficulty ranging from easy to hard. However, these users also provided valuable feedback and insights that are particularly helpful for improving usability and onboarding for novice users.

The most frequently reported issues were related to a lack of feedback and guidance throughout parts of the scenario, particularly in tasks Q4 and Q5. As noted by participant P5: “Making connections between elements (in 3D music installations) is tedious and not intuitive for me.” VR novices also commonly requested reminders for controls ”I would have liked a HUD display with the list of interactions and gestures, buttons...”, especially for actions like moving and rotating objects. Regarding the comparison between the floating menu and the 3D shop, participants pointed out interaction difficulties with both interfaces. Nonetheless, the overall difficulty of tasks **T1** and **T2** was generally rated as easy to very easy.

User Satisfaction and Preferences Participants consistently highlighted three key strengths of the system: the clear visual representation of musical installations through connected elements, the realism and quality of the spatialized sound, and the intuitive, hands-on concept of constructing music within a 3D environment.

The most frequent complaints concerned the lack of general feedback (reported by 5 out of 6 participants), difficulties with object manipulation controls (3 out of 6), and the absence of sufficient visual cues, particularly problematic for VR novices

(3 out of 6). Participants P2, P4, and P5 specifically requested enhanced visual cues when moving or rotating objects, such as custom cursors, 3D representations of rotation paths, or similar indicators to improve spatial interaction clarity.

Participants also noted difficulties in connecting two nodes, primarily due to the absence of visual feedback during the drag-and-connect process. For better guidance, they suggested adding a contextual menu displaying the functions of the controller buttons. Additionally, they recommended including tooltips for the plugin shop elements to help users better understand each component's purpose and functionality. Regarding the two interaction methods tested, participants expressed mixed preferences. P1 favored the 3D shop approach, noting that it felt more natural and took better advantage of VR's spatial interaction capabilities compared to traditional menus.

In contrast, P3 and P4 preferred the Floating Menu for its efficiency and immediate accessibility. Behavioral observations indicated that P6 remained mostly stationary throughout the experiment, relying almost exclusively on the Floating Menu rather than navigating through the virtual space. P6 also suggested introducing a circular 'pie menu' attached to the wrist as a potential enhancement to the current interaction methods."

P5 highlighted that too many steps were required, which creates arm fatigue when creating an installation, and that coming from a musical background, what he wanted to recreate should take a maximum of 30 seconds. He made a suggestion to have 'presets', a suggestion that P3 also supported, "I would like to choose a guitar preset and have a piano roll sequencer connected to a virtual guitar instrument connected to a set of 5 classic audio effects connected to an audio output, with one click! Then I can customize".

When asked about the potential purpose of the application, participants suggested entertainment/gaming (3/6), educational use (2/6), and live music performance or jam sessions (1/6).

VI. Discussion and Conclusion

This paper presented the evolution of *WAM Jam Party*, a browser-based, collaborative platform for immersive music creation using Web Audio Modules plugins in WebXR. The new version significantly improves on previous prototypes by introducing spatial interaction metaphors (e.g., the 3D Plugin Shop), enhanced usability features (e.g., redesigned menus, new gestures), and support for visually distinctive, customizable 3D GUIs for audio plugins, thanks to a new 3D editor and API.

The user study validated several design choices. In particular, it highlighted users' appreciation of the spatial clarity of the audio graphs, while also pointing to ways it could be improved. Participants further praised the realism and quality of the spatialized sounds, as well as the engaging, hands-on interaction metaphor. However, it also highlighted important usability challenges, particularly for novice users, including a lack of visual feedback, difficulty with spatial manipulation, and a need for clearer guidance and controller affordances. The observed preference split between the Floating Menu and 3D Shop indicates the value of offering multiple interaction modalities, depending on user preference or context (e.g., standing vs. seated use). Participants' suggestions, such as adding pie menus, tooltips, or preconfigured presets, point toward future improvements in accessibility and workflow speed, especially for experienced musicians seeking efficiency.

From a broader perspective, *WAM Jam Party* demonstrates how web-based immersive environments can support not only expressive music creation with plugins comparable in quality to native ones, but also collaborative workflows in real time. The ability to co-build modular audio graphs in a shared virtual space represents a concrete step toward the vision of a Musical Metaverse, a networked, spatial, and social space for sonic interaction. Future work will focus on addressing the usability issues identified, expanding the plugin ecosystem with additional embodied instruments and modulators, implementing session management, presets, and simplified onboarding tools, and conducting further studies with experienced composers and performers to explore the system's potential for collaborative composition and live performance.

Compared to native solutions built with Unity, Unreal Engine, or similar platforms, which typically require installing applications directly on VR headsets, managing platform-specific permissions, and handling complex software updates, WAM Jam Party presents a more accessible alternative. In contrast to certain research projects such as the MIDI instruments developed by Berthaut [5], which rely on tethered desktop machines for audio and MIDI processing, WAM Jam Party runs entirely within the browser, using only the computational resources available on standalone VR headsets. Despite this constraint, the system supports a wide range of high-quality WAM plugins, many of which are on par with native audio tools. These include synthesizer ports such as the Prophet V and Oberheim OB-Xd, as well as faithful recreations of iconic hardware audio effects like the Big Muff and Eventide Blackhole reverb. This demonstrates that professional-grade audio quality can be achieved without native installation, using web technologies alone. The result is a platform that lowers the entry barrier for immersive music creation while maintaining the expressive and sonic depth expected by musicians and composers.

To conclude, while the system introduces visually distinctive 3D GUIs, we acknowledge that many current interactions remain 2D affordances transposed into 3D space. For example, the Piano Roll sequencer adapts mouse interaction into a controller ray: notes are set with trigger clicks, long notes with click-and-drag, and viewport navigation via scrolling gestures. Similarly, most parameter changes rely on virtual knobs/sliders/switches. Beyond the drum kit and early experiments with 3D modulation orbiters, we have not yet fully exploited affordances unique to WebXR, such as embodied gestural mapping, spatial metaphors, or constraints for richer haptic play. Addressing these gaps, by designing interaction models native to immersive environments, forms a key direction for future development.

ACKNOWLEDGMENT

This work was supported by the France 2030 investment plan managed throught the "UCA DS4H" and ANR-17-EURE-0004 projects.

REFERENCES

[1] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[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] F. Berthaut. 3D interaction techniques for musical expression. *Journal of New Music Research*, 2020.

[5] F. Berthaut. Gdpd/ivmi-builder: A libre software framework for extended reality musical instruments and sonic installations. In *Proceedings of the 19th Linux Audio Conference*, 2025.

[6] A. Boem, D. Dziwis, M. Tomasetti, S. Etezazi, and L. Turchet. "it takes two"-shared and collaborative virtual musical instruments in the musical metaverse. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–10. IEEE, 2024.

[7] A. Boem, M. Tomasetti, and L. Turchet. Issues and challenges of audio technologies for the musical metaverse. *J. Audio Eng. Soc*, 73(3):94–114, 2025.

[8] 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.

[9] L. Bruns, B. Saurbier, T. M. Voong, and M. Oehler. Presence and flow in virtual and mixed realities for music-related educational settings. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–7. IEEE, 2024.

[10] 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,.

[11] M. Buffa and S. Demont. Can you DAW it Online? In *IS2 2024 - IEEE International Symposium on the Internet of Sounds 2024 / 1st IEEE International Workshop on the Musical Metaverse (IEEE IWMM)*, Erlangen, Germany, Sept. 2024.

[12] 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.

[13] M. Buffa, P. Kouyoumdjian, Q. Beauchet, Y. Forner, and M. Marynowic. Making a guitar rack plugin -WebAudio Modules 2.0. In *Web Audio Conference 2022*, Cannes, France, July 2022.

[14] M. Buffa and J. Lebrun. Rocking the web with browser-based simulations of tube guitar amplifiers. *Journal of the Audio Engineering Society*, 71(11):753–768, 2023.

[15] 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.

[16] 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.

[17] M. Buffa and A. Vidal-Mazuy. WAM-studio, a Digital Audio Workstation (DAW) for the Web. In *ACM digital library*, WWW '23 Companion: Companion Proceedings of the ACM Web Conference 2023, Austin TX USA, United States, Apr. 2023.

[18] A. Çamcı, M. Vilaplana, and R. Wang. Exploring the affordances of vr for musical interaction design with vimes. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 121–126, 2020.

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

[20] 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.

[21] 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.

[22] A. F. Genovese, Z. Nguyen, M. Gospodarek, R. Pahle, C. Brenner, and A. Roginska. Holodeck: A research framework for distributed multimedia concert performances. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–10. IEEE, 2024.

[23] J. L. Gómez-Sirvent, F. López de la Rosa, R. Sánchez-Reolid, R. Morales Herrera, and A. Fernández-Caballero. Musical instruments in extended reality: A systematic review. *International Journal of Human–Computer Interaction*, pages 1–20, 2024.

[24] 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.

[25] Q. Lan and A. R. Jensenius. Browser-based collaborative live coding with glicol: A graph-oriented live coding language written in rust. In *Proceedings of the Web Audio Conference (WAC)*, 2021.

[26] J. Lanier. Virtually there. *Scientific American*, 284(4):66–75, 2001.

[27] P. LUCAS, S. FASCIANI, and K. GLETTE. Csound vs. chuck: Sound generation for xr multi-agent audio systems in the meta quest 3 using the unity game engine. *Perspectives on immersion through laser doppler vibrometry*, page 301, 2024.

[28] T. Mäki-Patola, J. Laitinen, A. Kanerva, and T. Takala. Experiments with virtual reality instruments. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 11–16, 2005.

[29] D. Ogborn, A. F. Briones, L. N. del Angel, A. Roberts, D. A. Stewart, J. Beverley, B. Gray, K. Oduro, J. Rodríguez, C. Testa, et al. Estuary 0.3: Collaborative audio-visual live coding with a multilingual browser-based platform. In *2022 Web Audio Conference*, 2022.

[30] R. Polfreman. Frameworks 3d: composition in the third dimension. In *Proceedings of the international conference on new interfaces for musical expression*, pages 226–229, 2009.

[31] 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.

[32] C. Roberts, I. Hattwick, E. Sheffield, and G. Smith. Rethinking networked collaboration in the live coding environment gibber. In *Proceedings of the International Conference on New Interfaces for Musical Expression NIME, Waipapa Taumata Rau, Aotearoa.*, 2022.

[33] S. Serafin, C. Erkut, J. Kojs, N. C. Nilsson, and R. Nordahl. Virtual reality musical instruments: State of the art, design principles, and future directions. *Computer Music Journal*, 40(3):22–40, 2016.

[34] L. Severi, M. Sacchetto, A. Bianco, C. Rottondi, G. Abbate, A. Paolillo, and A. Giusti. Remote orchestral conduction via a virtual reality system. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–6. IEEE, 2024.

[35] D. Sun, C. Sun, A. Ng, and W. Cai. Real differences between ot and crdt in correctness and complexity for consistency maintenance in co-editors. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):1–30, 2020.

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

[37] 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*, 10(13):11264–11292, 2023.

[38] G. Wakefield, M. Palumbo, and A. Zonta. Affordances and constraints of modular synthesis in virtual reality. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 547–550, 2020.