OPEN ACCESS Freely available online
D. Dziwis, H. V. Coler and C. Pörschmann,
"Orchestra: A Toolbox for Live Music Performances in a Web-Based Metaverse"
J. Audio Eng. Soc., vol. 71, no. 11, pp. 802–812, (2023 November).
DOI: https://doi.org/10.17743/jaes.2022.0096

Orchestra: A Toolbox for Live Music Performances in a Web-Based Metaverse

DAMIAN DZIWIS,[1, 2*] *AES Student Member*, **HENRIK VON COLER,**[2] **AND**
(damian.dziwis@th-koeln.de) (voncoler@tu-berlin.de)

CHRISTOPH PÖRSCHMANN,[1] *AES Associate Member*
(christoph.poerschmann@th-koeln.de)

[1]*Institute of Computer and Communication Technology, TH Köln - University of Applied Sciences, Cologne, Germany*
[2]*Audio Communication Group, Technische Universität Berlin, Berlin, Germany*

As the potential of networked multiuser virtual environments increases under the concept of the metaverse, so do the interest and artistic possibilities of using them for live music performances. Live performances in online metaverse environments offer an easy and environmentally friendly way to bring together artists and audiences from all over the world. Virtualization also enables countless possibilities for designing and creating artistic experiences and new performance practices. For many years, live performances have been established on various virtual platforms, which differ significantly in terms of possible performance practices, user interaction, immersion, and usability. With Orchestra, we are developing an open-source toolbox that uses the Web Audio Application Programming Interface to realize live performances with various performance practices for web-based metaverse environments. Possibilities vary from live streaming of volumetric audio and video, live coding in multiple (including audiovisual) programming languages, to performing with generative algorithms or virtual instruments developed in Pure Data. These can be combined in various ways and also be used for telematic/networked music ensembles, interactive virtual installations, or novel performance concepts. In this paper, we describe the development and scope of the Orchestra toolbox, as well as use cases that illustrate the artistic possibilities.

0 INTRODUCTION

The term metaverse originated in the dystopian science fiction novel "Snow Crash" [1], in which it describes a virtual world interwoven with people's physical reality. In our current technology landscape, it has come to serve as an umbrella term for a specific category of virtual environments. In contrast to standalone, offline virtual- or augmented-reality (VR/AR) applications, metaverse environments are understood as "an interconnected web of social, networked immersive environments in persistent multiuser platforms" [2]. Although there is some consensus on this definition, the term metaverse is used to market all kinds of VR/AR applications, virtual multiplayer games, or social experiences. In particular, multiplayer games like Fortnite [3] are often used as examples of metaverse environments [4]. However, strict to the definition given, these environments do not meet the metaverse requirements, because they are self-contained environments, which also have a domain-specific purpose due to the respective game mechanics. Although virtual social worlds such as VRChat [5], Horizon Worlds [6], or the pioneering Second Life [7] are much more diverse in their possibilities, social interactions, and immersion, they are still self-contained applications that cannot connect to each other.

Therefore, it is primarily web-based multiuser virtual environments that can be experienced through browsers on immersive VR/AR devices, in addition to computers or smartphones, that meet the definition of metaverse environments. Depending on the implementation, these applications can include a wide range of social and cultural interactions, and because they are internet-based, enable connectivity between these environments. Collectively, these web-based environments can be seen as the current state of the metaverse. Platforms that allow the creation of such metaverse environments are, for example, STYLY [8] or Mozilla Hubs [9].

There are several approaches to virtual live music performances on the aforementioned platforms and beyond. A particular advantage of virtualization is that artists and audiences are independent of location. In addition to solo artists, ensembles and groups of geographically distributed

*To whom correspondence should be addressed: damian.dziwis@th-koeln.de.

performers can also realize so-called telematic/networked music performances (NMPs) [10]. In the same way, audiences can also participate in such performances regardless of their location. Virtual live performances may involve different concert formats or performance practices, depending on the technological implementation used.

Audio streaming is probably the most common technology. Local audio streams from artists can be transmitted, often using streaming technology optimized for musical performances [11, 12]. In this way, signals from acoustic or electronic instruments, as well as speech or vocals, can be used.

Virtual instruments are another suitable form of technology. Depending on the implementation, virtual instruments in a network can replace the need to transmit audio between participants with the ability to transmit control data and perform sound synthesis locally at each site [13].

Live coding is a growing form of performance practice and technology [14]. It is a performance practice where the on-the-fly programming of algorithms for the generative composition of music and/or visuals is performed live in front of an audience [15]. The reproduction of sound through loudspeakers, usually in stereo, and the projection of the programming code, optionally with visual elements, on a screen behind the performer and their laptop on stage, is a common form of presentation in a musical context. As with virtual instruments, in suitable networked music or virtual live performance environments, instead of audio, only the source code for local audio rendering can be transferred between participants [16].

Using these technologies, a variety of virtual live performances can be realized. However, the crucial factor here is how these are presented and how the audience participates, as well as the interactions that are made possible between each other and the performers. In contrast to 2D, in audio and video, virtual live performances using video conferencing and/or streaming platforms [17], performances and composition in 3D virtual (especially metaverse or metaverse-like) environments can offer crucial advantages [18–21]. Here, immersion and a sense of presence can be enhanced for both audience and performers, especially in conjunction with VR/AR devices [22]. Direct and augmented social (as well as cultural) interaction and participation can be enabled, allowing users to interact with the environment and each other [23]. Text, voice, and video chats enable social interactions for and between the audience and performers. As will be shown throughout the paper, the potential of 3D virtual environments opens up a wide range of possibilities for environmental and stage design, allowing the development and realization of unique artistic concepts and novel performance practices.

For this purpose, we have developed the open-source Orchestra toolbox for live performances in web-based metaverse environments. The meaning of the name is ambiguous: like the large instrumental ensemble, the toolbox can be used to bring together different instruments and performers. The word originated in ancient Greek, meaning a part of the stage of a Greek theatre, just as the toolbox can be seen as a part of the novel stages in the form of metaverse environments. Orchestra consists of individual components that cover the aforementioned virtual live performance formats: volumetric audio and video streaming, virtual instruments, and live coding. The components are compatible with each other and have been developed for widely used web-based frameworks for metaverse environments. This allows Orchestra to be used in arbitrarily designed, networked, social, and immersive multiuser metaverse environments to create diverse live performances and musical art experiences.

In the following, we describe the development of the audio-based components for live music performances included in the Orchestra toolbox. We show the integration of the selected frameworks for web-based metaverse environments (SEC. 2) and spatial audio (SEC. 2.2). Furthermore, the development of the Web Audio-based components (SEC. 3) and the resulting artistic possibilities and use cases (SEC. 4) will be presented.

1 MOTIVATION AND RELATED WORK

There is a long history of numerous realizations of music performances on the platforms mentioned in SEC. 0 [24]. Very common is the playback of audio to performing virtual avatars in a virtual environment [25], sometimes with highly detailed animations based on the performer's motion capture [26]. In more advanced realizations, high-quality (but usually pre-recorded) live performances with volumetric audio and video are integrated into the virtual environments [27]. A simple form is to integrate 2D audio and video live streams from video-streaming platforms into virtual environments. This approach has been particularly popular in the live coding community. However, the result here is more like a public screening than a live performance, in that the audience watches a large virtual screen instead of the performer.

Not all the presented examples are in metaverse environments by the definition given. They differ in the degree of possible immersion, e.g., using VR/AR devices and spatial audio, or in social interaction and participation. Preproduced performances, whether volumetric audio/video or motion capture, exclude the artists from the virtual environment and don't allow interaction between the audience and performers. This also applies to integrated live streaming using video-streaming platforms.

There is no open and customizable toolbox for web-based metaverse environments that combines the possibilities of live music performances and other realizations with volumetric audio streaming, virtual instruments, and live coding in multiuser metaverse environments. Therefore, Orchestra's live music performance components are developed for implementation in web-based metaverse environments to allow the creation of live music performances with a high degree of immersion by integrating VR/AR devices and spatial audio.

The focus is on the realization of real-time performances inside these metaverse environments to enable participation and social interaction between the audience and artists. The components cover the possibilities of real-time volumetric audio and video streaming, performances with virtual instruments and generative algorithms, as well as live coding. Intercompatibility and multiuser interaction allow for the arbitrary combination of components and resulting performance practices (for example, for NMP ensembles) or interactive virtual installations. The use of the Web Audio Application Programming Interfaces (API) enables integration into web-based metaverse environments with local audio rendering in the browser, without the need for plugins or other software installations.

2 METAVERSE ENVIRONMENT DEPENDENCIES

Web-based implementations are particularly suited for creating an interconnected network of metaverse environments. Being built on the Internet, these are networked by default and allow exchange between each other. There are also available web technologies for browser-based implementations to meet the remaining requirements for metaverse environments (see SEC. 0). JavaScript-based APIs such as Web Audio [28] and WebGL [29] allow the rendering of audiovisual virtual 3D environments directly in the browser without the need for additional software. The WebXR API [30] enables the use of VR/AR systems with stereoscopic rendering on head-mounted displays (HMDs), roaming in 6 degrees of freedom (6-DoF) through head and room tracking, as well as controllers and hand tracking for interaction.

There are several ways to create web-based metaverse environments using the APIs mentioned here. Game engines such as Unity [31], especially in combination with special add-ons such as the Spatial Creator Toolkit [32], enable the high-level design of web-based multiuser virtual environments using graphical interfaces. Far more control over code and implementation is offered by low-level programming of such applications. In addition to using the APIs with plain JavaScript programming, frameworks such as Three.js [33] facilitate development. A balance between low-level JavaScript programming, a higher-level markup language, and a graphical editor is provided by the A-Frame framework [34].

2.1 A-Frame and Networked-Aframe

A-Frame is a higher-level framework based on Three.js. Although it is also JavaScript-based at its core, it abstracts the description of 3D virtual worlds into an HTML-like [35] markup language. A-Frame also includes the A-Frame Inspector, a visual editor that allows the design of 3D scenes using graphical user interfaces similar to those of other 3D software and game engines. The software architecture of A-Frame corresponds to an entity component system (ECS) [36]. The JavaScript-based components allow the programming of 3D entities such as geometries and models, materials, light, shadows, and multimedia content such as images, videos, or sounds. The scope of the components can be extended by integrating third-party or custom-developed components in JavaScript.

Although virtual environments developed in A-Frame can also be used with conventional PCs or mobile devices with interfaces such as a mouse, keyboard, or touch screen, there is a special focus on VR/AR integration in A-Frame. The integrated WebXR API also allows the use of current VR/AR systems with associated hardware such as HMDs, tracking, or controllers. By allowing the use of compatible hardware of your choice, the virtual environments can be accessed by as many users as possible.

For extending A-Frame worlds to shared multiuser environments, as required for the metaverse, the Networked-Aframe (NAF) library is available [37]. Also written in JavaScript, NAF provides a set of adapters for exchanging data over WebRTC [38] or WebSockets [39] to synchronize user interactions between each other or the environment. Using the supplied adapters, data are transferred from user to user in a peer-to-peer (P2P) network, ensuring interactivity and persistence in resulting metaverse environments. It enables the implementation of shared objects such as avatars and the synchronization of attributes of all A-Frame components, including custom-developed ones. The integration of WebRTC also allows low-latency broadcasting of audio and video streams. Using NAF, any interaction with objects in the environment, as well as user interactivity such as voice, video, or text chats, can be implemented.

Just as the popular metaverse platform Mozilla Hubs is based on A-Frame/NAF, the Orchestra toolbox was also developed for this combination. Orchestra includes components and entity primitives for the integration in A-Frame environments. For multiuser interactivity using NAF, templates are included. NAF's WebRTC real-time audio implementation is used for the audio streaming component. Interactions with virtual instruments, as well as the source code of the live coding components, are shared over data broadcast. The Orchestra components can thus be integrated into any kind of metaverse environment that is designed and implemented on the basis of A-Frame/NAF.

2.2 Resonance Audio

In addition to the visual reproduction with HMDs, as well as the tracking of movements, the auralization of spatial audio using headphones is another aspect of the immersive experience of 3D environments. A 3D sound experience over headphones is achieved through dynamic binaural reproduction of sound sources, which are rendered by real-time convolution with head-related transfer functions (HRTFs) depending on the user's head orientation and room position [40]. By default, the A-Frame framework uses the Three.js implementation of the Web Audio PannerNode object [41] with the HRFT panner model to provide binaural audio for headphones.

Instead of using this standard spatialization, a spatializer with room simulation is implemented to add reverberation to the acoustic scene, increasing the audiovisual coherence by acoustically counteracting the visually designed environment [42]. For this purpose, the Resonance Audio spatializer [43] was integrated as a component in Orchestra. The Resonance Audio Web SDK is based on the Omnitone [44] spatial audio rendering framework, written using the

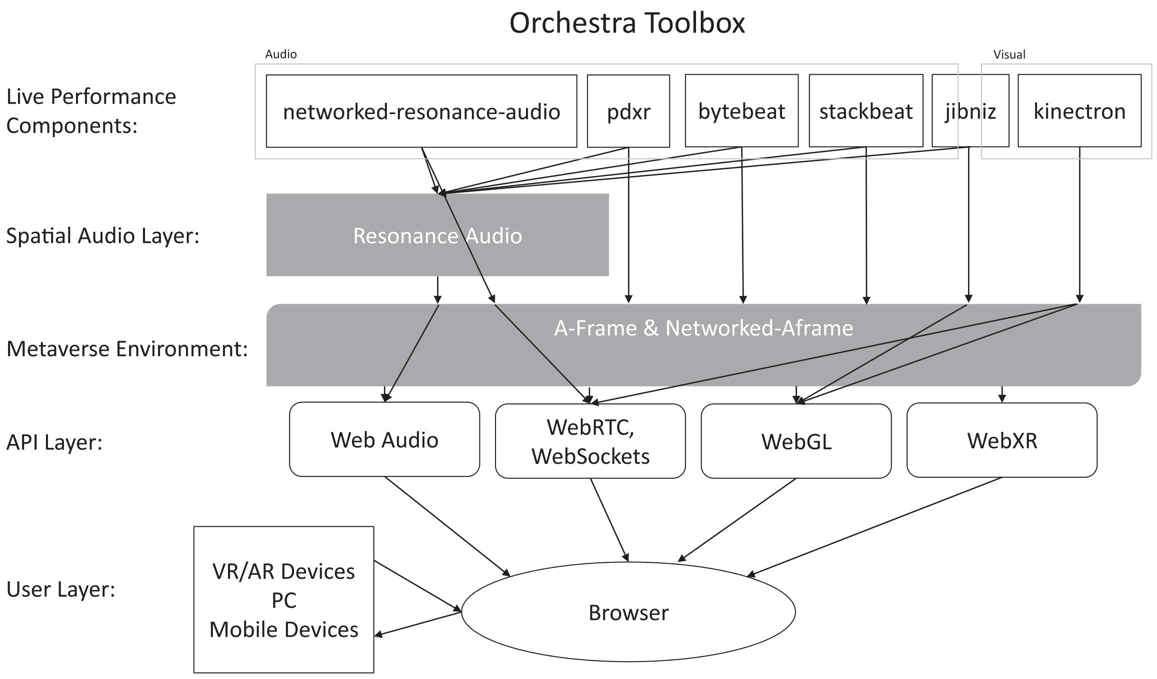

Fig. 1. Architecture of the Orchestra toolbox with all components, dependencies, involved frameworks, and programming interfaces.

Web Audio API. It considers the room geometry and material properties of the reflecting surfaces of a specified room and thus supplements the binaural rendering with a spatial room simulation. A ray-tracing-based model is implemented for early reflections and filters for late reverberation on ambisonic content [45].

The Resonance Audio spatializer is a suitable choice for web-based solutions [46], particularly for the Orchestra toolbox, because there is already an open-source port for the A-Frame framework. The component [47] included in Orchestra is a fork of this existing A-Frame component [48], extended with the possibility of moving sound sources and implemented for use with the Orchestra live performance components. The extended Resonance Audio component thus forms the spatial audio rendering basis for Orchestra and must therefore be integrated into the metaverse environments created.

3 COMPONENTS FOR MUSIC PERFORMANCES

The Orchestra toolbox is a collection of A-Frame/NAF-compatible components for live music performances. It is developed in JavaScript and combines various web-based technologies (see Fig. 1). As described in the previous section, Orchestra is intended for use in A-Frame/NAF-based metaverse environments, where the included Resonance Audio component provides the spatial audio layer for the audio components. The audio components use the Web Audio API to render audio in the browser.

In addition to the audio components, the Orchestra toolbox also provides visual, WebGL-based components, such as the Kinectron component for rendering streamed volumetric video [49], or the visual rendering of the IBNIZ audiovisual live coding language [50]. As the visual parts are beyond the scope of this paper, they will not be discussed further. In the following, we describe each included audio component's implementation and functionality.

3.1 Networked-Resonance-Audio

The networked-resonance-audio component enables real-time streaming of audio signals over WebRTC as sound sources for the Resonance Audio spatializer. It is closely based on the networked-audio-source component from the NAF library [51] and has been extended to integrate with the Resonance Audio layer. In the original networked-audio-source component, WebRTC streamed audio is used as a MediaStream [52] for a Three.js PositionalAudio object [53]. This already enables binaural auralization, but to integrate room simulation as well, the networked-resonance-audio component included in Orchestra implements the stream as the source for an attached Resonance Audio Source (reseonance-audio-src) [54]. In this way, audio streams are reverberated to match the characteristics of the specified acoustic room (see SEC. 2.2).

NAFs WebRTC implementation [55] uses RTCPeerConnection [56] for audio MediaStreams with the Opus codec [57] by default and aims for low-latency broadcasts using the Real-Time Transport Protocol (RTP) [58]. Depending on a number of factors, latency can vary greatly here [59]. When transferred over User Datagram Protocol (UDP) [60], RTP is comparable with plain UDP with the addition of a header, and the latency is additionally influenced by factors such as the underlying network speed/latency, encoding and decoding, and the audio input/output (I/O) latency.

Linked to an avatar and its position, users can broadcast speech or acoustic interaction such as applause via microphone sharing in the browser. Likewise, immersive audiovisual volumetric live performances can be staged,

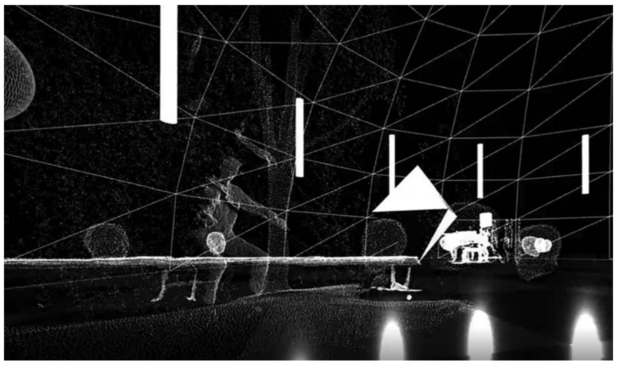
Fig. 2. A screen capture of the metaverse performance “The Entanglement” [83]. A telematic improvisation piece using streamed volumetric audio and video involving components included in the Orchestra toolbox.

especially when linked to the volumetric video rendering of the Kinectron component (see Fig. 2).

3.2 PdXR

The PdXR (Pure Data extended reality) components [61] allow the execution and shared interaction of Pure Data (Pd) patches in metaverse environments. Pd is an open-source visual programming language that offers a wide range of possibilities for audio and multimedia programming [62], such as virtual musical instruments or interactive generative algorithms. In addition to MIDI or signal processing objects for programming algorithms, Pd provides numerous Graphical User Interaction (GUI) objects for interactive algorithms. A project that allows running and interacting with Pd patches in the browser is PdWebParty [63]. Here Pd, transpiled into WebAssembly [64] using emscripten [65], is used as the Pd runtime, while GUI objects are rendered as interactive HTML elements in the browser. This project has been adapted to develop PdXR for metaverse environments.

Based on PdWebParty, the Pd WebAssembly runtime, which uses the SLD2 Audio API [66] with Web Audio as the audio rendering backend, has been adapted for use with the Resonance Audio spatializer. The emscripten-generated implementation of the signal processing as a ScriptProcessorNode [67] with the resulting signal as a MediaStream has been modified to be used as the source for a resonance-audio-src. The audio from running Pd patches is thus rendered as a monoaural sound source that is binauralized and spatialized in the metaverse environment. The JavaScript-based parser and API to handle the GUI objects and their interaction are adapted to render them as interactive 3D A-Frame entities (see Fig. 3). Changed parameters of the objects, or applied interactions, are shared with all users inside a metaverse environment using NAF. This enables low-latency, bandwidth-saving NMPs, as there is no need to stream audio.

As Pd is an open programming language, the algorithms that can be implemented are manifold and almost countless. Because PdXR is an implementation of the native Pd environment, the possibilities are similarly diverse. Limitations are mainly related to GUI objects, data management, and audio I/O. GUI objects implemented at the time of writing are bang, toggle, vertical/horizontal slider, number2, floatatom, and symbolatom. Although the implementation of other GUI objects is in progress, some cannot be usefully ported to virtual environments (e.g., patch design elements). The integration of Web MIDI API [68] is also in the works. Data management (e.g., to support external files such as samples) or I/O features of Pd such as microphone input or multichannel audio cannot be usefully integrated in the foreseeable future. The multiuser ability for interaction with all Pd GUI objects makes it possible to perform together on virtual instruments or to create interactive virtual installations.

3.3 Live Coding Components

The Orchestra toolbox contains components for live coding within metaverse environments [69]. Available live coding components include the Bytebeat, StackBeat, and IBNIZ languages. Not only is the implementation of these components similar, but so are the properties of the languages. The similarity lies in the minimalist language design, and the reduced instruction set. For this reason, these

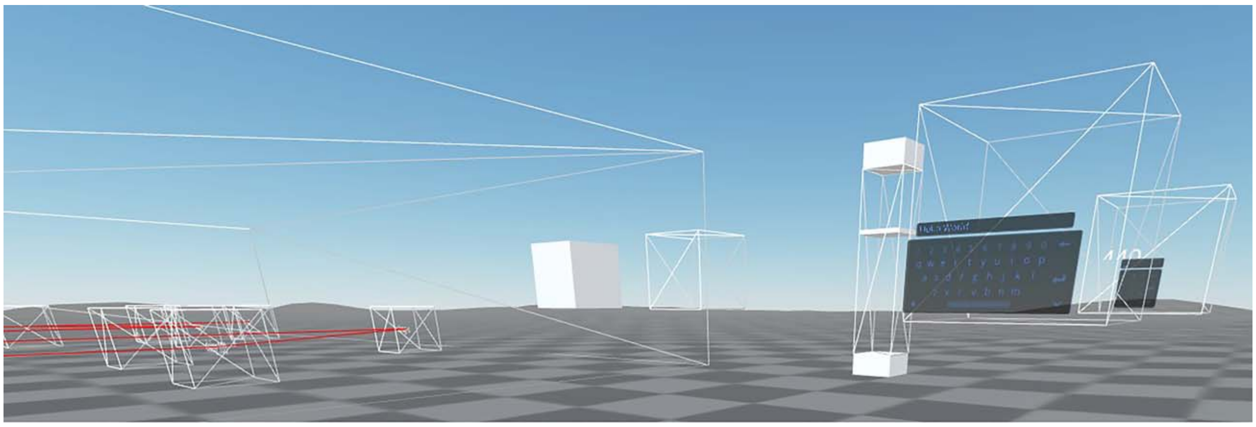
Fig. 3. A screen capture of an example metaverse environment with the PdXR component. The image shows the involved Pd patch with the interactive GUI objects ‘bang,’ ‘toggle,’ ‘vertical slider,’ ‘floatatom,’ and ‘symbolatom.’

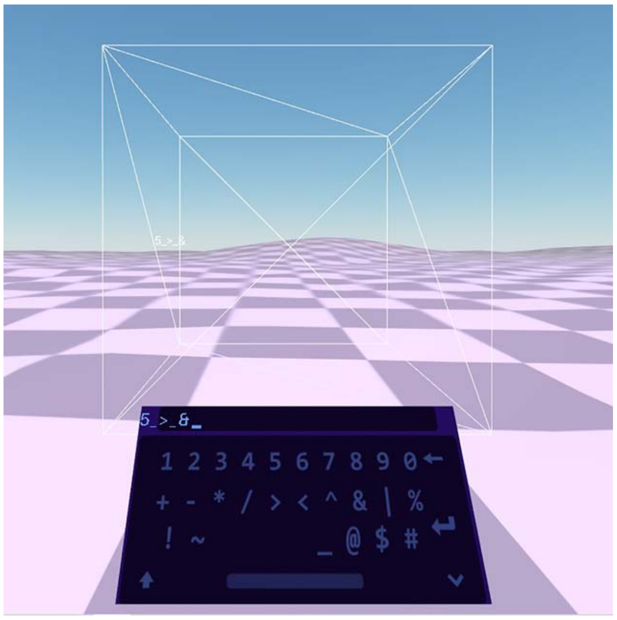

Fig. 4. A screen capture showing the StackBeat live coding component. It includes an open cube containing the current source code and a virtual keyboard as a programming terminal.

are considered esoteric programming languages (Esolangs). Esolangs are a type of programming languages that are not intended for general-purpose use but for the implementation of conceptual, creative, and sometimes artistic language designs [70].

Because of the reduced instruction set, consisting of single characters, the included live coding languages are particularly efficient for programming in virtual environments. Thus, even when using virtual keyboards with arbitrary Human Interface Devices (HIDs), ranging from VR/AR controllers to hand tracking and conventional computer mice, programming can be done quickly and comfortably. All three live coding components feature an A-Frame-based programming interface (see Fig. 4) with an interactive virtual keyboard that can be used with the aforementioned HIDs. The programming interface was developed using a modified version [71] of the Aframe-Super-Keyboard [72]. By using NAF, the live coding source code is shared between all users. Again, this enables low-latency and low-bandwidth performances without audio streaming. Running multiple instances of live coding components allows for realizing NMPs or virtual installations.

As StackBeat and IBNIZ are based on the Bytebeat concept, the languages also have a lot in common aesthetically. The musical output of all three can be described as 8-bit/Chiptune-like. Because the languages have a high degree of complexity and expressiveness despite their minimalism, this aesthetic limitation is probably the most significant. In the following, we describe the implementation of the individual live coding components in more detail.

3.3.1 *aframe-bytebeat*

Bytebeat is a concept for composing algorithmic music [73]. It uses arithmetic and logical operations as one-line formulas. Bytebeat can be implemented in many programming languages, in its most minimal form consisting of arithmetic and logical expressions on a variable *t*, incremented in an ideally infinite loop. The 8-bit data output can be rendered as a raw PCM audio stream [74]. The arithmetic and logic expressions on the infinite variable *t* can be considered as a small domain-specific programming language [75]:

```
t&t>>8
```

Code 1. Bytebeat code for a Sierpinski triangle composition [73].

There are several implementations of web-based Bytebeat live coding environments, even for NMPs [76]. For Bytebeat live coding within metaverse environments, we have developed a JavaScript-based on-the-fly interpreter component integrated with A-Frame/NAF. The parser was designed to handle an adapted instruction set that includes the conventional Bytebeat syntax with a few variations. To reduce instructions to single characters, multicharacter instructions were replaced by single characters (">>" becomes "r" or "sin" simply becomes "s").

Using the Web Audio API, audio rendering is implemented as an AudioWorklet processor [77] for prioritized threading and seamless signal processing. The resulting real-time audio stream is spatialized with the described Resonance Audio layer as a positional sound source with the location of the programming interface.

3.3.2 *aframe-stackbeat*

StackBeat is a stack-based implementation of the Bytebeat concept [78]. By using a stack-machine architecture, the number of instructions and characters required can be further reduced. The algorithm for a fractal Sierpinski triangle composition can be reduced from the Bytebeat variant with 6 characters (see Code 1 in Sec. 3.3.1) to a StackBeat algorithm with only 5 characters: 8_ > _&. This makes StackBeat even more efficient for live coding in metaverse environments with virtual keyboards. The original one-line JavaScript implementation found on esolangs generates an offline rendered wave file containing the composed music at a specified length, making it unsuitable for live coding [78]. To enable StackBeat live coding in the metaverse, the code parser of the original implementation was partially adopted and reimplemented into an on-the-fly interpreter [79]. The result is again an A-Frame component that uses the Web Audio API, largely equivalent to the Bytebeat component. It has a similar implementation of the AudioWorklet processor and spatial audio rendering.

3.3.3 *aframe-jibniz*

IBNIZ, "Ideally Bare Numeric Impression giZmo," is a virtual machine (VM) for low-level programming of audiovisual algorithms [80]. It is closely related to the concept of Bytebeat but extends its musical possibilities with visual capabilities. Because of the on-the-fly interpreter of the VM, IBNIZ is also suitable for live coding [81]. It includes a small set of stack operations that can manipulate both a video and audio stack, separately or simultaneously

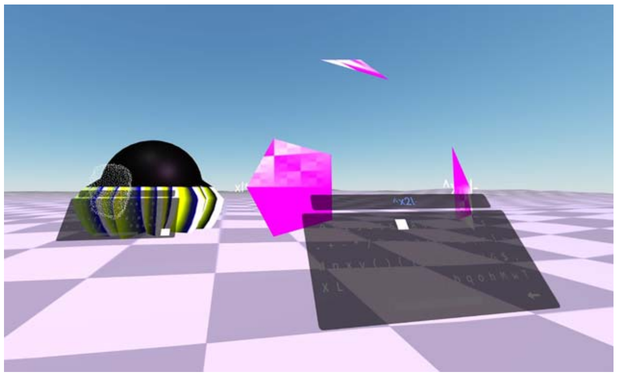

Fig. 5. A screen capture showing two users collaborating on two audiovisual jibniz live coding components.

[82]. This allows for efficient live coding of audiovisual algorithms.

One project that brings IBNIZ to the browser is jibniz, a JavaScript port of the IBNIZ VM [50]. For metaverse environments, we developed an implementation that adapts and wraps jibniz as an A-Frame component and integrates it with the NAF library. As with the previous live coding components, they share the same style of programming terminal, the source code is shared between all users, and the Web Audio implementation is adapted to the Resonance Audio spatializer. The WebGL-based visual rendering acts as a displacement texture for any 3D geometries. This allows the jibniz live coding component to create animated, constantly changing audiovisual objects using short, one-line algorithms (see Fig. 5).

4 USE CASES

Components from the Orchestra toolbox have already been used in several live performances. The realizations took place in different metaverse environments and for different artistic concepts and performance practices. The first public performance using described components took place as part of the telematic artistic-research residency of the Institute for Computer Music and Sound Technology (ICST) at the Zurich University of the Arts (Switzerland). The telematic metaverse piece "The Entanglement" (2022) [83] was realized as a metaverse performance with volumetric audiovisual rendering, using live-streamed networked audio and 3D depth camera video of two violinists. It is an improvisation based on a quantum computer algorithm [84]. The telematic performance was realized within the network of the Zurich University of the Arts, and both audio and depth video from two Kinect 3D cameras were streamed online to the audience in a P2P network topology. The audience consisted of approximately 10 or more participants in the two runs of the performance. In addition to the online performance in the metaverse, it was also presented as a local AR concert installation in the Immersive Arts Space of the ICST.

The live coding piece "Mnemonic Garden" (2022) premiered at the "die digitale" festival in Düsseldorf (Germany) [85]. "Mnemonic Garden" was realized using the Bytebeat live coding component in a virtual environment that was specifically designed for the concept of the piece (see Fig. 6). As this was also a hybrid concert performance for a virtual audience, as well as one in physical reality, some elements of a conventional live coding performance were adopted. The virtual world designed was an extension of the stage idea but included conventional live coding elements such as a large projection screen to display the source code. The virtual audience consisted of about 5 people, with latencies being negligible despite the P2P network topology because only changes to the source code need to be transmitted as text data.

Live performances in metaverse environments make it possible to experience concerts online in a way comparable with physical ones through immersive technology and social participation. However, in the use cases mentioned, it was primarily the capabilities of virtualization that made unique stage designs and the implementation of specific artistic concepts possible in the first place. The potential of live performances in metaverse environments lies not only in bringing artists and audiences together online but above all, in realizing novel art experiences and performance practices. This is where the advantages of the Orchestra toolbox become particularly apparent. The integration and combination of technologies for audiovisual volumetric live streaming, virtual instruments, and live coding can be used for different combinations of performance practices, for single performers or ensembles, or interactive installations.

Because the audio and visual representations of the components are interpreted as spatial sound sources and virtual objects in a 3D virtual environment, they can also be combined with common spatial composition techniques, such as creating trajectories and adding reverberation and echoes through acoustic simulation [86]. Besides the use cases already realized, virtual environments also allow the realization of new practices, such as the integration of gamification elements [87], like worldbuilding concepts known from sandbox multiplayer games such as Minecraft [88], and much more to be explored.

5 CONCLUSION AND FUTURE WORK

With Orchestra, we have developed an open-source toolbox that combines several technologies for different types of live performances in A-Frame/NAF-based metaverse environments. It allows the realization of immersive online solo performances, NMPs, interactive virtual installations, and the creation of new art practices for metaverse environments. The open-source code allows deep customization and extension to desired implementations. So far, the toolbox includes components for volumetric audio and video streaming, a Pure Data implementation for virtual instruments and interactive algorithms, and three programming languages for live coding. As shown in the use cases, public performances have already been realized with these components.

These also revealed some limitations with the current implementations. Even if, as described in SEC. 4 with the

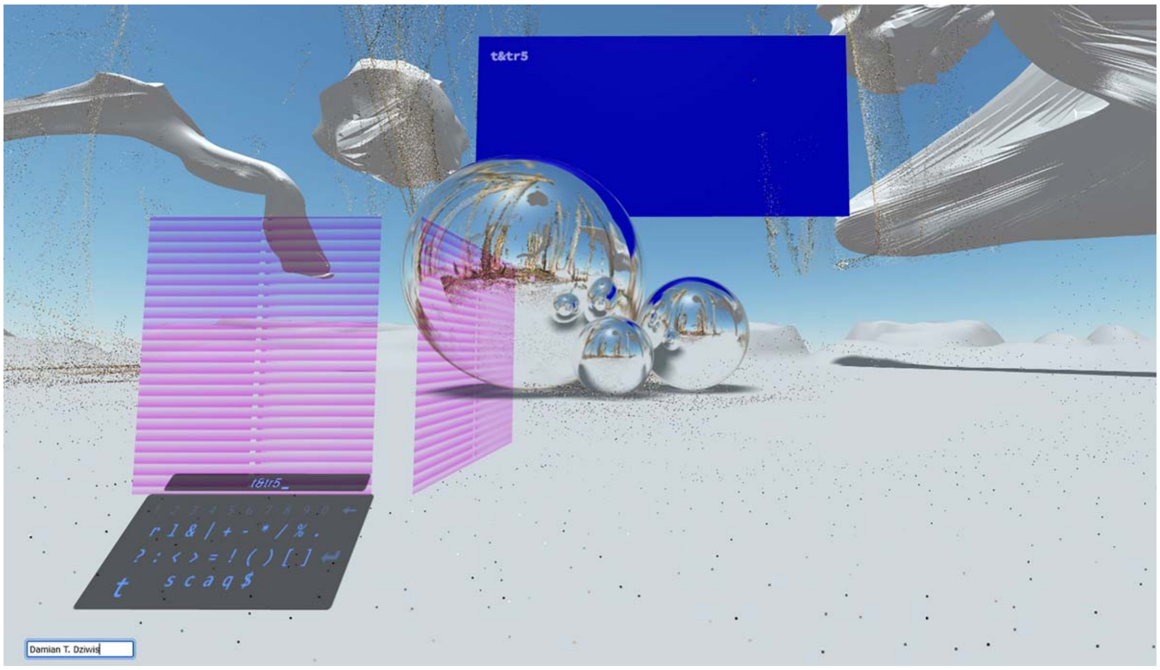

Fig. 6. A screen capture from the “Mnemonic Garden” live coding performance [85]. Realized with the Bytebeat live coding component, included in a custom-designed virtual environment, and the programming terminal enhanced with an audio visualization.

"The Entanglement" performance, NMPs with audio and video streaming via WebRTC in P2P topologies are possible with high-bandwidth networks and latency uncritical compositions, limits in terms of latency and stability of the connections can quickly be reached. As latency can vary strongly and no synchronization process is built in, care needed to be taken during the conception and composition of the piece to ensure that latency would not have a negative impact on performance and reception. As a result, the current implementation is more suited to realizing novel and tailor-made music pieces and performances than transforming existing material. Even under good conditions, latencies in standard WebRTC audio streams can be perceived as too high ($>$60 ms) for latency-critical NMPs [89]; this is also true for volumetric video streaming [90].

As a solution, audio streaming for networked performances in the metaverse can be combined with software optimized for NMPs [12] to achieve tolerable latencies on the performer side. On the audience side, because higher latency is negligible, volumetric audio and video streams can be transmitted using HTTP Live Streaming (HLS) from video streaming platforms. This type of streaming architecture would also be beneficial for solo performances with large audiences. HLS streaming and optimizing the WebRTC audio streams for lower latency [89] are being considered for future development. With the current components, latency-critical NMPs and solo performances are also achievable when using virtual instruments with PdXR or the live coding components, as no audio streaming is required. The three live coding languages presented, although one of them is audiovisual, are similar in their aesthetic results.

In order to expand the possibilities of live coding in the metaverse, the implementation of other languages is also being considered. Although the realization as browser-based applications means high accessibility for users because they are platform-independent, device-agnostic, and do not require additional software installation, browsers can also impose additional barriers. Compatibility can vary even among popular browsers (e.g., incompatibility with Apple’s Safari), particularly due to security mechanisms (such as disabling automatic sound playback) or local settings that are beyond the developer’s control. Through extensive testing and additional scripts, high compatibility with Mozilla Firefox and Chromium-based browsers (e.g., Google Chrome or Microsoft Edge) has been achieved. To mitigate bugs and add compatibility and features, the components in the Orchestra toolbox are constantly being developed. The repository of the Orchestra toolbox can be found here: https://github.com/AudioGroupCologne/Orchestra

6 REFERENCES

[1] N. Stephenson, *Snow Crash* (Bantam Books, New York, NY, 1992).

[2] S. Mystakidis, “Metaverse,” *Encyclopedia*, vol. 2, no. 1, pp. 486–497 (2022 Mar.). https://doi.org/10.3390/encyclopedia2010031.

[3] Epic Games, “Fortnite,” https://www.fortnite.com/ (accessed Feb. 6, 2023).

[4] L.-H. Lee, Z. Lin, R. Hu, et al., “When Creators Meet the Metaverse: A Survey on Computational Arts,” *arXiv preprintarXiv:2111.13486v1* (2021). https://doi.org/10.48550/arXiv.2111.13486.

[5] VRChat Inc, “VRChat,” https://hello.vrchat.com/ (accessed Feb. 6, 2023).

[6] Meta, “Horizon Worlds,” https://www.meta.com/horizon-worlds/ (accessed Feb. 6, 2023).

[7] Linden Research, Inc, “Second Life,” https://secondlife.com/ (accessed Feb. 6, 2023).

[8] Psychic VR Lab, “STYLY,” https://gallery.styly.cc/ (accessed Feb. 6, 2023).

[9] Mozilla Corporation, “Mozilla Hubs,” https://hubs.mozilla.com/ (accessed Feb. 6, 2023).

[10] P. Oliveros, S. Weaver, M. Dresser, J. Pitcher, J. Braasch, and C. Chafe, “Telematic Music: Six Perspectives,” *Leonardo Music J.*, vol. 19, pp. 95–96 (2009 Dec.). https://doi.org/10.1162/lmj.2009.19.95.

[11] J. P. Cáceres and C. Chafe, “Jacktrip: Under the Hood of an Engine for Network Audio,” *J. New Music Res.*, vol. 39, no. 3, pp. 183–187 (2010 Nov.). https://doi.org/10.1080/09298215.2010.481361.

[12] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, “An Overview on Networked Music Performance Technologies,” *IEEE Access*, vol. 4, pp. 8823–8843 (2016 Dec.). https://doi.org/10.1109/ACCESS.2016.2628440.

[13] G. Hajdu, “Quintet.net: An Environment for Composing and Performing Music on the Internet,” *Leonardo*, vol. 38, no. 1, pp. 23–30 (2005 Feb.). https://doi.org/10.1162/leon.2005.38.1.23.

[14] C. Nilson, “Live Coding Practice,” in *Proceedings of the Seventh International Conference on New Interfaces for Musical Expression*, pp. 112–117 (New York, NY) (2007 Jun.). https://doi.org/10.1145/1279740.1279760.

[15] N. Collins, A. McLean, J. Rohrhuber, and A. Ward, “Live Coding in Laptop Performance,” *Organised Sound*, vol. 8, no. 3, pp. 321–330 (2003 Dec.). https://doi.org/10.1017/S135577180300030X.

[16] D. Ogborn, J. Beverley, L. Navarro Del Angel, E. Tsabary, and A. McLean, “Estuary: Browser-Based Collaborative Projectional Live Coding of Musical Patterns,” in *Proceedings of the International Conference on Live Coding* (Morelia, Mexico) (2017 Dec.).

[17] C. Basica, “Quarantine Sessions #3,” https://www.youtube.com/watch?v=5eSeLVCh2cE (accessed Feb. 15, 2023).

[18] L. Turchet, “Musical Metaverse: Vision, Opportunities, and Challenges,” *Pers. Ubiquitous Comput.* (2023 Jan.). https://doi.org/10.1007/s00779-023-01708-1.

[19] L. Turchet, R. Hamilton, and A. Çamci, “Music in Extended Realities,” *IEEE Access*, vol. 9, pp. 15810–15832 (2021 Jan.). https://doi.org/10.1109/ACCESS.2021.3052931.

[20] B. Loveridge, “Networked Music Performance in Virtual Reality: Current Perspectives,” *J. Network Music Arts*, vol. 2, no. 1, paper 2 (2020 Aug.).

[21] M. Ciciliani, “Virtual 3D Environments as Composition and Performance Spaces*,” *J. New Music R.*, vol. 49, no. 1, pp. 104–113 (2020 Jan.). https://doi.org/10.1080/09298215.2019.1703013.

[22] M. Slater, B. Lotto, M. M. Arnold, and M. V. Sanchez-Vives, “How We Experience Immersive Virtual Environments: The Concept of Presence and Its Measurement,” *Anuario de Psicologia*, vol. 40, no. 2, pp. 193–210 (2009 Sep.).

[23] S.-M. Park and Y.-G. Kim, “A Metaverse: Taxonomy, Components, Applications, and Open Challenges,” *IEEE Access*, vol. 10, pp. 4209–4251 (2022 Jan.). https://doi.org/10.1109/ACCESS.2021.3140175.

[24] R. Elen, “Music in the Metaverse,” in *Proceedings of the AES UK 23rd Conference: Music Everywhere* (2008 Apr.), paper 16.

[25] Avatar Orchestra Metaverse, “Avatar Orchestra Metaverse,” https://avatarorchestra.blogspot.com/ (accessed Feb. 6, 2023).

[26] T. Scott, “Travis Scott and Fortnite Present: Astronomical (Full Event Video),” https://www.youtube.com/watch?v=wYeFAlVC8qU (accessed Feb. 6, 2023).

[27] VRROOM “Oxymore - Overview,” https://vimeo.com/767098450/6fcf797c5a (accessed Feb. 6, 2023).

[28] W3C, “Web Audio API,” https://www.w3.org/TR/webaudio/ (accessed Feb. 6, 2023).

[29] Khronos Group, “WebGL,” https://www.khronos.org/webgl/ (accessed Feb. 6, 2023).

[30] W3C, “WebXR,” https://immersiveweb.dev/ (accessed Feb. 6, 2023).

[31] Unity Technologies, “Unity Game Engine,” https://unity.com/ (accessed Feb. 6, 2023).

[32] Spatial Systems, Inc., “Spatial Creator Toolkit,” https://www.spatial.io/ (accessed Feb. 6, 2023).

[33] Three.js, “Three.js,” https://threejs.org/ (accessed Feb. 6, 2023).

[34] Supermedium, “A-Frame,” https://aframe.io (accessed Feb. 6, 2023).

[35] W3C, “HTML,” https://www.w3.org/html/ (accessed Feb. 15, 2023).

[36] Networked-Aframe, “Entity Systems Wiki,” http://entity-systems.wikidot.com/ (accessed Feb. 15, 2023).

[37] “Networked-Aframe,” https://github.com/networked-aframe (accessed Feb. 6, 2023).

[38] Google WebRTC Team, “WebRTC,” https://webrtc.org/ (accessed Feb. 6, 2023).

[39] WHATWG, “WebSockets,” https://websockets.spec.whatwg.org/ (accessed Feb. 15, 2023).

[40] H. Møller, “Fundamentals of Binaural Technology,” *Appl. Acoust.*, vol. 36, nos. 3–4, pp. 171–218 (1992 Mar.). https://doi.org/10.1016/0003-682X(92)90046-U.

[41] Mozilla Corporation, “Mozilla, Web Audio PannenNode,” https://developer.mozilla.org/en-US/docs/Web/API/PannerNode (accessed Feb. 15, 2023).

[42] S. Werner, F. Klein, T. Mayenfels, and K. Brandenburg, “A Summary on Acoustic Room Divergence and Its Effect on Externalization of Auditory Events,” in *Proceedings of the 8th International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6 (Lisbon, Portugal) (2016 Jun.). https://doi.org/10.1109/QoMEX.2016.7498973.

[43] Resonance Audio, “Resonance Audio,” https://resonance-audio.github.io/resonance-audio/ (accessed Feb. 6, 2023).

[44] Google, “Omnitone Repository,” https://googlechrome.github.io/omnitone (accessed Feb. 15, 2023).

[45] Resonance Audio, “Web SDK Source Code,” https://github.com/resonance-audio/resonance-audio-web-sdk/tree/master/src (accessed Feb. 6, 2023).

[46] A. McArthur, C. van Tonder, L. Gaston-Bird, and A. Knight-Hill, “A Survey of 3D Audio Through the Browser: Practitioner Perspectives,” in *Proceedings of the Immersive and 3D Audio: From Architecture to Automotive (I3DA)* (Bologna, Italy) (2021 Sep.). https://doi.org/10.1109/I3DA48870.2021.9610839.

[47] D. Dziwis, “A-Frame Resonance Audio,” https://github.com/AudioGroupCologne/aframe-resonance-audio-component (accessed Feb. 6, 2023).

[48] M. Kungla, “A-Frame Resonance Audio,” https://github.com/mkungla/aframe-resonance-audio-component/ (accessed Feb. 6, 2023).

[49] D. Dziwis, “A-Frame Kinectron,” https://kinectron.github.io/#/aframe/getting-started (accessed Feb. 21, 2023).

[50] L. Escot, “jibniz,” https://github.com/flupe/jibniz (accessed Feb. 15, 2023).

[51] Networked-Aframe, “Networked-Audio-Source,” https://github.com/networked-aframe/networked-aframe/blob/master/src/components/networked-audio-source.js (accessed Feb. 21, 2023).

[52] Mozilla Corporation, “MediaStream,” https://developer.mozilla.org/en-US/docs/Web/API/MediaStream (accessed Feb. 15, 2023).

[53] Three.js, “PositionalAudio,” https://threejs.org/docs/index.html#api/en/audio/PositionalAudio (accessed Feb. 21, 2023).

[54] M. Kungla, “Resonance Audio Source,” https://github.com/mkungla/aframe-resonance-audio-component/blob/main/src/resonance-audio-src.js (accessed Feb. 21, 2023).

[55] J. Valin and C. Bran, “WebRTC Codec and Media Processing Requirements,” *RFC 7874* (2016 May). https://doi.org/10.17487/RFC7874.

[56] Mozilla Corporation, “RTCPeerConnection,” https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection (accessed Mar. 13, 2023).

[57] J. M. Valin and K. Vos, “Updates to the Opus Audio Codec,” *RFC 8251* (2017 Oct.). https://doi.org/10.17487/RFC8251.

[58] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” *RFC 3550* (2003 Jul.). https://doi.org/10.17487/RFC3550.

[59] O. Komperda, H. Melvin, and P. Pocta, “A Black Box Analysis of Webrtc Mouth-To-Ear Delays,” *Commun.*, vol. 18, no. 1, pp. 11–16 (2016 Feb.). https://doi.org/10.26552/com.C.2016.1.11-16.

[60] J. Postel, “User Datagram Protocol,” *RFC 768* (1980 Aug.). https://doi.org/10.17487/RFC0768.

[61] D. Dziwis, “PdXR,” https://github.com/AudioGroupCologne/PdXR (accessed Feb. 15, 2023).

[62] M. Puckette, “Pure Data: Another Integrated Computer Music Environment,” in *Proceedings of the Second Intercollege Computer Music Concerts*, pp. 37–41 (Tachikawa, Japan) (1997 May).

[63] Z. Lee, “PdWebParty,” https://github.com/cuinjune/PdWebParty (accessed Feb. 21, 2023).

[64] Mozilla Corporation, “WebAssembly,” https://developer.mozilla.org/en-US/docs/WebAssembly (accessed Oct. 17, 2023).

[65] Emscripten Contributors, “emscripten,” https://emscripten.org/ (accessed Feb. 21, 2023).

[66] “Simple DirectMedia Layer,” https://www.libsdl.org/ (accessed Feb. 21, 2023).

[67] Mozilla Corporation, “Mozilla, Web Audio ScriptProcessorNode,” https://developer.mozilla.org/en-US/docs/Web/API/ScriptProcessorNode (accessed Feb. 15, 2023).

[68] W3C, “Web MIDI API,” https://www.w3.org/TR/webmidi/ (accessed Feb. 15, 2023).

[69] D. Dziwis, H. von Coler, and C. Pörschmann, “Live Coding in the Metaverse,” in *Proceedings of the Fourth International Symposium on the Internet of Sounds (2023)*. (in press).

[70] D. Temkin, “Language Without Code: Intentionally Unusable, Uncomputable, or Conceptual Programming Languages,” *J. Sci. Technol. Arts*, vol. 9, no. 3, pp. 83–91 (2017 Sep.). https://doi.org/10.7559/citarj.v9i3.432.

[71] D. Dziwis, “Aframe-Super-Keyboard,” https://github.com/AudioGroupCologne/aframe-super-keyboard (accessed Feb. 6, 2023).

[72] Supermedium, “Aframe-Super-Keyboard,” https://github.com/supermedium/aframe-super-keyboard (accessed Feb. 6, 2023).

[73] V.-M. Heikkilä, “Discovering Novel Computer Music Techniques by Exploring the Space of Short Computer Programs,” *arXiv preprint arXiv:1112.1368* (2011). https://doi.org/10.48550/arXiv.1112.1368.

[74] Kragen, “Bytebeat,” http://canonical.org/∼kragen/bytebeat/ (accessed Feb. 6, 2023).

[75] M. Voelter, S. Benz, C. Dietrich, et al., *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages* (Stuttgart, Germany, 2013).

[76] I. J. Clester, “Kilobeat: Low-Level Collaborative Livecoding,” in *Proceedings of the Web Audio Conference* (Barcelona, Spain) (2021 Jul.).

[77] Mozilla Corporation, “Web Audio AudioWorklet,” https://developer.mozilla.org/en-US/docs/Web/API/AudioWorklet (accessed Feb. 15, 2023).

[78] Esolang, “StackBeat,” https://esolangs.org/wiki/StackBeat (accessed Feb. 15, 2023).

[79] D. Dziwis, “aframe-stackbeat,” https://github.com/AudioGroupCologne/aframe-stackbeat (accessed Feb. 15, 2023).

[80] V. Heikkilä, “IBNIZ,” http://viznut.fi/ibniz/ (accessed Feb. 15, 2023).

[81] D. T. Dziwis, “m e t a − m i ph o s is,” https://www.youtube.com/watch?v=zGNpgOPByZY (accessed Feb. 6, 2023).

[82] V. Heikkilä, “IBNIZ Documentation,” https://github.com/viznut/IBNIZ/blob/master/src/ibniz.txt (accessed Feb. 15, 2023).

[83] D. T. Dziwis, “The Entanglement,” https://www.youtube.com/watch?v=yKGc8dYJSLs (accessed Feb. 6, 2023).

[84] D. Dziwis and H. von Coler, “The Entanglement – Volumetric Music Performances in a Virtual Metaverse Environment,” *J. Network Music Arts*, vol. 5, no. 1, paper 3 (2023 May.).

[85] D. T. Dziwis, “Mnemonic Garden,” https://www.youtube.com/watch?v=HJGbvXs45sk (accessed Feb. 6, 2023).

[86] M. A. Baalman, “Spatial Composition Techniques and Sound Spatialisation Technologies,” *Organ. Sound*, vol. 15, no. 3, pp. 209–218 (2010 Dec.). https://doi.org/10.1017/S1355771810000245.

[87] R. Hamilton, “Collaborative and Competitive Futures for Virtual Reality Music and Sound,” in *Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1510–1512 (Osaka, Japan) (2019 Mar.). https://doi.org/10.1109/VR.2019.8798166.

[88] Microsoft, “Minecraft Homepage,” https://www.minecraft.net/ (accessed Feb. 6, 2023).

[89] M. Sacchetto, P. Gastaldi, C. Chafe, C. Rottondi, and A. Servetti, “Web-Based Networked Music Performances via WebRTC: A Low-Latency PCM Audio Solution,” *J. Audio Eng. Soc.*, vol. 70, no. 11, pp. 926–937 (2022 Nov.). https://doi.org/10.17743/jaes.2022.0021.

[90] L. Turchet, N. Garau, and N. Conci, “Networked Musical XR: Where’s the Limit? A Preliminary Investigation on the Joint Use of Point Clouds and Low-Latency Audio Communication,” in *Proceedings of the 17th International Audio Mostly Conference*, pp. 226–230 (St. Pölten, Austria) (2022 Sep.). https://doi.org/10.1145/3561212.3561237.

THE AUTHORS

Damian Dziwis

Henrik von Coler

Christoph Pörschmann

Damian Dziwis is a researcher and composer/media artist based in Düsseldorf, Germany. He received his B.Eng. in media technology from the Hochschule Düsseldorf - University of Applied Sciences in 2015 and his M.Mus. in electronic composition from the Cologne University of Music (HfMT Köln) in 2017. Since 2017, he is working on his Ph.D. at the TH Köln - University of Applied Sciences and the Technische Universität Berlin in the field of music in virtual environments. Since 2023, he is a visiting professor at the Hochschule für Musik Detmold in the department of composition and sound design. He was also artist and researcher in residence at the ZKM (Center for Art and Media) Karlsruhe and at the Institute for Computer Music and Sound Technology (ICST) at the Zurich University of the Arts. His compositions and installations have been shown at various international festivals.

•

Henrik von Coler is a researcher, teacher, and composer/performer in the field of Electroacoustic Music. He studied electrical engineering at HAW Hamburg, followed by Audio Communication and Technology (MSc) at the TU Berlin where he also obtained his Doctoral Degree in 2021. Since 2015, he is the director of the "TU Studio" for electronic music at Technical University Berlin. His research areas include algorithms for sound analysis/synthesis, spatial audio practices, musical interfaces, and network systems for music interaction. Combining research questions with applications in electronic music and sound art, he is in charge of several systems for spatial sound reproduction.

•

Christoph Pörschmann studied Electrical Engineering at the Ruhr-Universität Bochum (Germany) and at Uppsala Universitet (Sweden). In 2001, he obtained his Doctoral Degree (Dr.-Ing.) from the Electrical Engineering and Information Technology Faculty of the Ruhr-Universität Bochum as a result of his research at the Institute of Communication Acoustics. Since 2004, he is a professor of Acoustics at TH Köln - University of Applied Sciences. His research interests are in the field of virtual acoustics, spatial hearing, and the related perceptual processes. Christoph Pörschmann is member of the German Acoustical Society and the Acoustical Society of America.