Interface Modules for Extended Reality in Music

Damian Dziwis
KreativInstitut.OWL, HfM Detmold,
(TU Berlin / TH Köln)
Bielefelder Strasse 66a
32756 Detmold, Germany
damian.dziwis@hfm-detmold.de

Aristotelis Hadjakos
Center of Music and Film Informatics
HfM Detmold / TH OWL
Hornsche Str. 44
32756 Detmold, Germany
aristotelis.hadjakos@hfm-detmold.de

ABSTRACT

The growing possibilities and availability of immersive technologies in the field of Extended Reality (XR) are leading to new potentials in the context of Musical XR, like the emergence of new concepts for virtual reality musical instruments (VRMIs), reception, and composition and performance practices. Recent XR devices for Virtual and Augmented Reality allow not only to present virtual content, but also to blend real and virtual environments to create hybrid Mixed Realities. Therefore, in addition to using standard interfaces such as controllers or hand tracking from XR systems for interacting with VRMIs, developing custom musical interfaces can provide enhanced physical possibilities and experiences for interacting with VRMIs and other virtual sound-generating systems. This paper describes the development and possibilities of an open-source modular interface system for the realization of VRMIs with physical musical interfaces, interactive installations and the augmentation of virtual environments with physical environmental information. The tool was created by integrating the SPINE sensor-based musical interface toolkit with the IVES modular 3D engine for the Max programming environment. Merging these systems, with their no-code/no-soldering approach, results in a customizable tool for rapidly creating VRMIs or augmented virtual environments with corresponding physical interfaces.

Author Keywords

Extended Reality, Virtual Reality Musical Instruments, 3D Engine, Physical Computing, Sensors

CCS Concepts

•**Applied computing** → **Sound and music computing;** *Media arts;* •**Human-centered computing** → *Haptic devices;* **User interface toolkits; Mixed / augmented reality; Virtual reality;**

NIME'24, 4–6 September, Utrecht, The Netherlands.

1. INTRODUCTION

In recent years, there has been a significant increase in the development and availability of immersive devices in the context of Extended Reality (XR). Extended Reality has become an umbrella term for different levels of immersive experiences on the reality-virtuality continuum [26]. This continuum describes the gradations between the boundaries of real environments, consisting only of real objects in physical reality, and fully virtual environments, as presented in virtual reality (VR). Hybrid forms, i.e. the mixing of real and virtual objects and environments, can be found in the subset of Mixed Reality (MR) [25]. The currently most prominent form in this subset is Augmented Reality (AR). This form is the primary display of the real environment with the addition of virtual objects [26]. Complementary to this is the form of augmented virtuality. In this case, a virtual environment is primarily displayed, which is augmented with objects from the real environment [24, 25].

The current focus of development in the area of consumer XR devices is on head-mounted display (HMD)-based devices. Current devices (e.g. Meta Quest 3, VIVE XR Elite) are increasingly being equipped with features such as passthrough video [11] in order to be able to display MR forms such as AR in addition to pure VR. However, specialized HMD systems for VR or AR can also be used to display MR, e.g. AV by integrating real objects into VR using additional sensors [29], or VR on AR HMDs by completely overlaying the real environment with virtual content. Thus, the entire reality-virtuality continuum can be covered with currently available XR devices.

With the increasing availability of such immersive devices, there is a growing interest in exploring the possibilities of XR in the context of music. The field of Musical XR offers numerous possibilities in the areas of composition, development, education, entertainment, perception, performance, and sound engineering [37]. Although the tool presented here could be used in all these areas of Musical XR, the scope of this paper will be the development of Virtual Musical Instruments (VMIs), or related applications such as interactive installations, in XR with physical interfaces. Here it is especially the creative freedom and possibilities that virtuality offers in terms of visual instrument and environment design, sound synthesis and the use of spatial composition techniques [3] that allow artistic concepts and ideas to be realized that would be impossible or very difficult and expensive to realize in physical reality.

Since the early days of virtual reality, there has been interest in using VR for virtual reality musical instruments (VRMIs). In 1993, Jaron Lanier presented "The Sound of One Hand", a performance on a VRMI with early VR technology using an HMD and a dataglove [17, 16]. With the continuous development of VR hardware, many other developments of VRMIs with different features and different

concepts followed [34]. Central topics in the development of VRMIs include the underlying sound synthesis [15], the way they are interacted with [18, 19], and, apart from audio-only VRMIs [1], the visual design of the instruments. The tool presented this paper offers a solution for the visual design and audio spatialization of VRMIs and the associated interfaces for interactions with them. In addition to the use of standard controllers or hand tracking from XR devices, there are also approaches to gesture-based VRMIs [22] using custom sensor-based interfaces and physical controllers for VRMIs [35, 19].

Implementing hybrid Mixed/Augmented Reality Musical Instruments (MRMIs,ARMIs), offers benefits such as reduced cyber sickness and seeing your own body [39]. By displaying the real environment, MRMIs also have the advantage of being able to integrate with real spaces in virtual installations [23], augment acoustic instruments to hyperinstruments [32, 20], or integrate custom controllers and interfaces that must be perceived visually.

The development of VR/MR/AR/AVMIs, in the following referred to as XR Musical Instruments (XRMIs), requires a range of skills, especially when the artistic or interaction concept requires input and control capabilities that can only be realized through the development of custom physical interfaces. Beside the implementation of sound synthesis, required skills can include knowledge of 3D engines and modeling for visual XRMI design, and in the case of physical interfaces, knowledge of micro-controller programming, physical computing, and sensor technology. To facilitate the development of XRMIs, software and tools were developed. In the area of physical-based instruments and interfaces [18, 4], sculpting [28], creating graphical interfaces for interaction with standard XR controllers [12, 14] or even the development of VRMIs within VR environments [30].

To aid the development of graphical XRMI interfaces with custom physical controllers, we introduce an open-source modular interface tool for the development of XRMIs in a 3D engine with spatial audio and corresponding interfaces with various sensors. The tool was developed by integrating the sensor-based physical computing toolkit SPINE [13] as interface modules for IVES, a modular toolkit for audio-visual rendering in Max [9]. These toolkits both follow a no-code/no-soldering principle, which minimizes the above mentioned skills required for the development of XRMIs and related custom interfaces. It allows rapid development using the given modules, while the openness and customizability allows expert users to modify and extend them. In the following, the underlying toolkits are presented, the development and concepts of the interface modules are explained, and the use cases and possibilities in the context of XRMIs are described.

2. TECHNICAL FRAMEWORK

The objective of the tool described in this paper is to facilitate the development of graphical virtual interfaces with corresponding custom physical interfaces for XRMIs and related applications like virtual interactive installations. It enables the rapid creation of virtual 3D objects, environments and spatial audio controlled by custom sensor-based interfaces for screen-based and XR applications. The tool's requirements were adapted to the target group, particularly the computer music domain, making existing skills usable through the integration of established software and hardware.

One established software in computer music for programming sound synthesis, algorithmic compositions and visuals

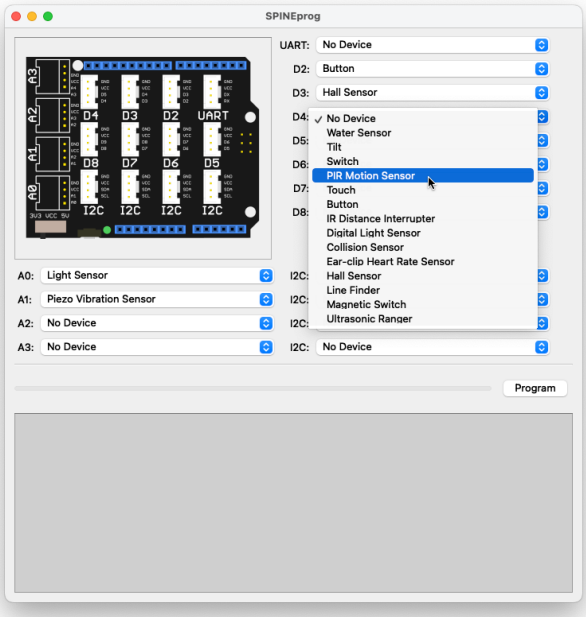

Figure 1: The SPINEprog is used to generate the firmware. It also handles communication with the microcontroller and the user's application.

```
#include "Spine.h"
#include <Wire.h>

namespace ConA0 {
    uint8_t connector = 1;
    uint8_t PIN_A = 18;
    uint8_t PIN_B = 18;
    #include "analog-generic.h"
}

namespace ConA1 {
    uint8_t connector = 2;
    uint8_t PIN_A = 19;
    uint8_t PIN_B = 19;
    #include "analog-generic.h"
}

namespace ConI2C1{
    uint8_t connector = 12;
    uint8_t PIN_A = 2;
    uint8_t PIN_B = 3;
    #include "imu-9dof.h"
}

void setup() {
    Spine.begin();
    Spine.configSensor(ConA0::connector, "Light", "f");
    ConA0::setup();
    Spine.configSensor(ConA1::connector, "Vibration", "f");
    ConA1::setup();
    Spine.configSensor(ConI2C1::connector, "IMU", "fffffffff");
    ConI2C1::setup();
}

void loop() {
    ConA0::loop();
    ConA1::loop();
    ConI2C1::loop();
    waitFor10ms();
}
```

Figure 2: The main loop of the SPINE firmware. All sensor-specific code is handled in the included header files. In the setup function, all sensors are initialized with their names and data types.

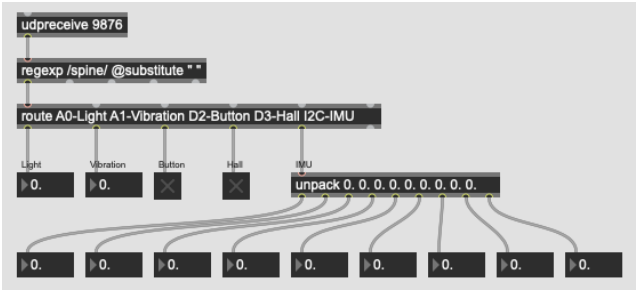

Figure 3: Patch to receive OSC/UDP packets generated by SPINEprog.

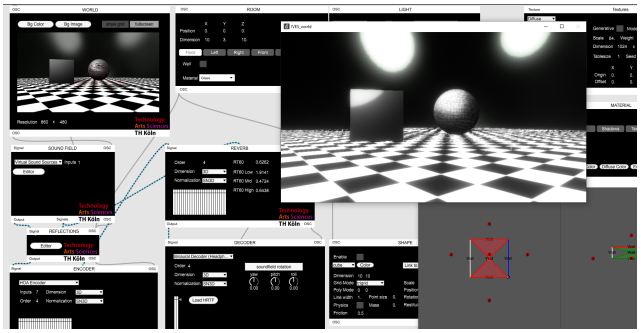

Figure 4: Virtual application created in IVES using different modules.

is Max [7]. In the field of physical computing and interface development, the Arduino platform [2] is a well-established environment for rapid development of microcontroller-based interfaces. These platforms not only enable rapid implementation of desired solutions, but also provide the flexibility to keep options open without limitations.

The modular interface tool developed also follows this principle. It allows users to rapidly develop XRMIs and physical interfaces by connecting modules without programming or soldering. Since not all possible requirements and use cases can be covered by ready-made modules, more experienced users are able to modify, extend and replace these modules as needed by adapting programming code and hardware.
Summarizing, our requirements are:

- Rapid development without coding/soldering: the tool is designed to enable rapid development of XRMIs and physical interfaces without requiring extensive knowledge of software and hardware development or 3D design.
- Adaptable code and hardware: to overcome the limitations of pre-built modules, experienced users should be able to customize them.
- Integration into the computer music ecosystem: the tool should integrate with established software to build on existing user skills.

To fulfill the need for comprehensive development of virtual 3D content and physical interfaces, we utilized two existing toolkits, IVES and SPINE, and developed interface modules to connect them. These toolkits are ideal for meeting the aforementioned requirements: they are open-source modular systems that already follow the defined requirements, enabling the development of visual 3D content and spatial audio (IVES), as well as physical sensor-based interfaces (SPINE), without requiring programming or soldering. Experienced users can customize these toolkits to meet their specific needs. The integration of these toolkits in the established software Max provides extensive possibilities, particularly the integration of a desired sound synthesis. Both toolkits are described in more detail below.

2.1 SPINE

SPINE is a physical computing platform based on open-source software (SPINEprog) and established hardware. It uses the Arduino Leonardo [2] and the Grove sensor system, a modular plug-and-play system of electronic sensors and components designed for rapid prototyping and experimentation in electronics [33]. The SPINEprog (Fig. 1) serves three purposes:

- Producing tailored firmware code and uploading the generated firmware onto the Arduino.
- Handling communication with the Arduino and forwarding the sensor data as Open Sound Control (OSC) / User Datagram Protocol (UDP) messages to the user's application.
- Generating OSC/UDP communication code that users can integrate into their own applications.

To create custom firmware, users first select which sensor is connected to which port. The Grove Shield has four analog inputs (A0-A3), seven digital inputs (D2-D8), four inputs for I^2C communication, and one input for Universal Asynchronous Receiver Transmitter (UART) communication. Grove sensors are easily connected to the shield using standard connectors. These connectors have 4-pin headers so users can connect components without having to worry about wiring or soldering. Users then select the appropriate sensor names from the SPINEprog user interface on the designated connectors. SPINEprog generates C++ code (Fig. 2). Users can export and edit this code. However, this is usually not necessary as SPINEprog uses the Arduino IDE in the background to compile and upload the code without user intervention.

SPINEprog and Arduino communicate via the USB Human Interface Device (HID) protocol. The USB HID standard simplifies the connection of input devices to computers by providing a common interface that is understood by operating systems without the need for additional drivers. This means that the operating system immediately recognizes the device and accepts input from it. SPINEprog registers itself with the corresponding device ID so that it receives all communication from the device. There are two types of messages: (1) a status message, which tells the SPINEprog what the sensor is called, which connector it is connected to, and what type of data it is transmitting, and (2) an actual data message, which provides data values for all sensors. The SPINEprog receives these messages and packs the data into several OSC/UDP packets that are forwarded to the user's application program. The OSC path of these messages is preceded by the prefix "/spine/", followed by a string consisting of the connector name and the sensor name, e.g. "/spine/A0-Light 642." indicates that a light sensor connected to the "A0" connector reports a raw value of 642. Communication is unidirectional from the SPINE to Max and IVES.

SPINEprog helps users by generating code that is used to receive and parse said OSC/UDP data. SPINEprog can generate the corresponding code for SuperCollider [36] and corresponding patches for Max and Pure Data [31]. Fig. 3 shows a generated patch for Max. This allows the user to

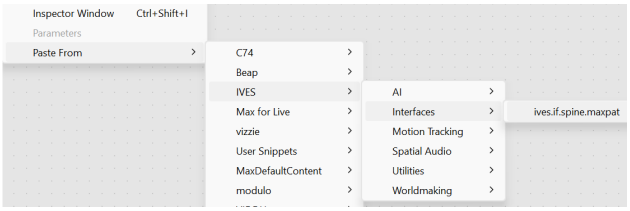

Figure 5: Max context menu showing the different categories of IVES modules.

immediately start working on their artistic projects without having to worry about low-level or communication code.

Latency and sampling rate: By default the SPINE samples its sensors at a sample rate of 100 Hz. This default rate presents a balanced approach allowing for smooth interaction while reducing the risk of overwhelming a slow processing system (for example a Max patch running complex calculations on the sensor updates). The sample rate can be reduced in the SPINE firmware code (see Fig. 2) or conversely increased, if a higher update rate is required for smoother interaction. We have not measured the latency incurred to transmit the data from the microcontroller to the computer via the USB HID protocol. However, previous studies on USB HID devices indicate that typical transmission latencies are approximately 12 ms for a device polled at 125 Hz and 2 ms for a device polled at 1000 hz [6]. The SPINEprog forwards the data via OSC/UDP to the receiving process on the local host. When OSC/UDP packets are transmitted over a network, this can introduce additional latencies, as examined in [21].

2.2 IVES

The Interactive Virtual Environment System (IVES) is a modular open-source toolkit for the development of immersive audiovisual 3D screen or XR applications. It was developed for use in Max, unifying established programming libraries such as Jitter/OpenGL, Spat [5] and the VR package [38]. IVES provides graphical user interface (GUI)-based modules that can be connected/patched together for rapid development of virtual 3D objects, environments and sound fields. In contrast to conventional 3D game engines (e.g. Unity, Unreal Engine), IVES places special emphasis on spatial audio and composition.

By connecting individual modules, complex audiovisual virtual environments and objects can be created (Fig. 4) to realize audiovisual spatial compositions and performances. The desired IVES module can be inserted by right-clicking in the Max editor (Fig. 5). Modules in the Worldmaking category can be used to add visual elements such as 3D models and shapes, associated materials and textures, light sources and the like. The WORLD module allows monoscopic rendering for screens while the WORLDXR module is used for rendering on XR devices. It enables stereoscopic rendering on Oculus, custom developed and SteamVR compatible HMDs and access to their tracking. AR and AV formats can also be created by adding video streams or additional sensors to the virtual scene. The modules in the Motion Tracking category allow the integration of external head trackers (e.g. from headphones) or Bose AR glasses for the realization of audio-only AR.

The modules in the Spatial Audio category allow the creation of virtual sound fields, the design of acoustic rooms [10], and the creation of rendering chains for encoding and decoding virtual sound fields for loudspeaker-based systems using Ambisonics [40] and on headphone-based systems using binaural rendering [27]. The Utilities modules can be used to animate and transform virtual objects or generate trajectories for complex motion patterns. The AI modules "Agent" and "Boids" allow the use of autonomous agents and the steering of the behavior of individual virtual objects and sound sources or complete "swarms" of them [8].

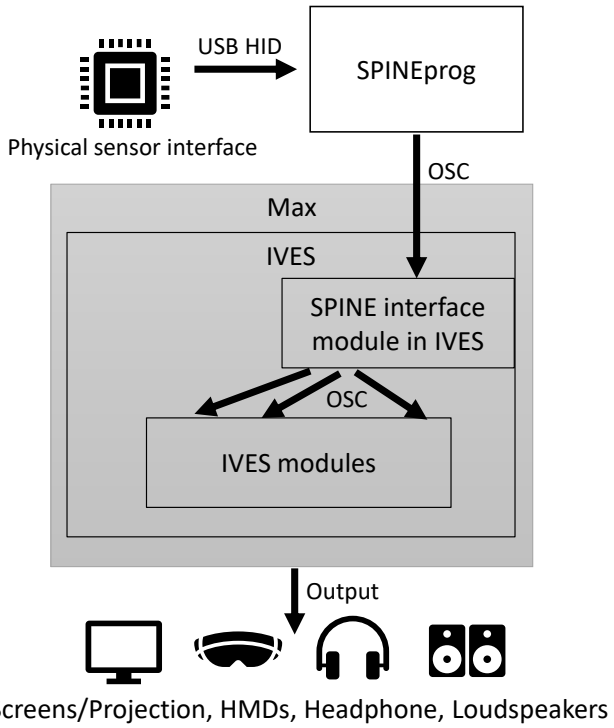

Figure 6: Overview of the SPINE/IVES system, showing the communication of components.

The modules are implemented natively in Max, with partial support by JavaScript, allowing them to be modified at runtime without recompilation. OSC messages are used for communication between modules and their parameterization. This allows the distribution modules via networked clients or the connection to other OSC sources. However, introducing additional network connections, e.g. through distributed rendering in IVES on multiple devices, leads to additional latencies which must be carefully considered for the intended use cases.

3. INTERFACE MODULES

In order to connect the IVES and SPINE toolkits described above (Sec. 2), an interface module (with several sub-modules) was developed to integrate sensor-based interfaces developed with SPINE with the IVES engine (Fig. 6). Since both toolkits use OSC messages for communication, OSC was retained as the communication protocol. The OSC data sent by the physical interface configured with SPINEprog is received by the developed IVES module "SPINE" presented in this paper. The SPINE module uses JavaScript to analyse the OSC data stream coming from the physical interface and to register a sub-module for each sensor (Fig. 7). The sub-modules themselves are implemented as native Max patches, ensuring that time-critical OSC messages carrying sensor values are being processed with the highest possible performance in a high-priority thread.

Users can switch between the sub-modules using a drop-down menu. The sub-modules provide GUIs for the respective sensor types, which can be used to adjust the behavior

and output of the sensor input. For each registered sensor, the SPINE module provides an outlet that can be connected to IVES modules for virtual objects, such as the MODEL module for 3D models or the SHAPE module for primitive 3D geometries. By interacting with the sensors of the physical interface, the virtual 3D objects can be parameterized and transformed. At the time of writing, the following sub-modules have been implemented for the corresponding sensors:

- "Button": used for the digital, manual sensors button, touch sensor and switch. The state of the sensors (0 = off, 1 = on) is used to control the color state of a connected virtual 3D object in IVES. The base color (off) and the active color (on) for virtual objects can be defined by the user. The design paradigm of color changes for pressed virtual buttons is taken from conventional user interface (UI) design commonly found in software GUIs. This mapping can be changed by experienced Max users directly in the program code of the module. As with all IVES modules, this also applies to the following sub-modules.
- "Slider": for analog, manual slider sensors. The value of the slider in the range of 0 - 1023 is used to translate a virtual 3D object or sound source. Users first select the target between object and sound source (including its ID) and specify a start and end point of the translation for each of the 3 axes X, Y and Z. The outlet of this module can then be connected to an IVES 3D object module or the SOUNDFIELD module to move the desired object between the two specified points relative to the position of the slider.
- "Rotary": for analog, manual rotary sensors outputting values between 0 and 1023 when rotated. The sensor is used to perform a rotation on an IVES 3D object. The user selects the desired axis (X, Y or Z) in the GUI and connects the output of the sub-module to a IVES 3D object to be rotated by using the sensor.
- "Tilt": for digital, manual tilt sensors. Tilt sensors allow the detection of movement and change of orientation. By selecting the axis through the user, connected virtual objects can be rotated by the specified start and end values. Unlike the rotary sensor, the tilt sensor does not transmit a continuous value, but only 1 if it is at rest or 0 if movement occurs. Thus, the rotation is triggered discretely using the previously defined values.
- "Light": for analog environmental sensors to detect illumination sources such as light and UV sensors. By defining the expected value range, users can recalibrate the sensor output data. By connecting the outlet of the module to an IVES LIGHT module, the attenuation of the light source and thus its intensity can be controlled. This sub-module in combination with light or UV sensors is suitable for the realization of Augmented Virtuality implementations, where the lighting conditions of the real environment can be transferred to the virtual environment.
- "Temperature": for analog environmental temperature sensors. In addition to the light sub-module, it provides another way to realize AV implementations by transferring the temperature of a real environment or object to virtuality. As with the Light sub-module, users can set the expected value range for recalibrating the sensor data. The sub-module can be used to apply a continuous color transition of a connected virtual object. The start and end color of the transition can be defined by the user in the GUI.

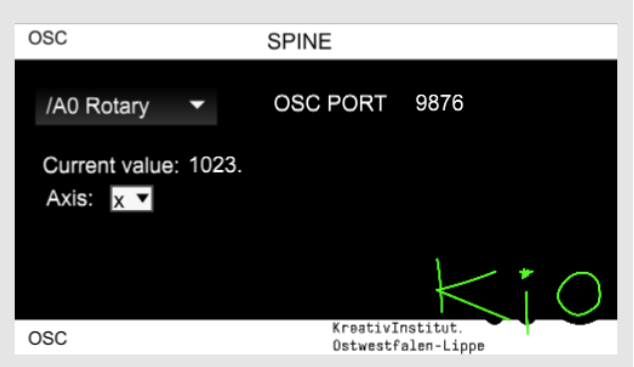

Figure 7: The SPINE interface module, with the rotary sensor sub-module selected.

With these sub-modules, the SPINE module in IVES can be used to process the input of physical interfaces with a variety of haptic manual as well as non-tactile environmental sensors. This enables a variety of transformations for virtual 3D objects and sound sources, as well as the realization of AV implementations using environmental sensors. The last outlet of the SPINE module outputs the raw OSC data of the physical interface, which can be used as indented by the SPINE toolkit for controlling and parameterizing sound generation algorithms within Max. New and existing SPINE projects can be realized in a virtual environment by adding visual presentation of XRMIs and spatializing the generated sound of the algorithms.

4. RESULTING APPLICATIONS

The integration of the SPINE physical interface toolkit with the IVES 3D engine enables a wide range of potential applications. With the SPINE toolkit, users can construct a wide variety of haptic and non-tactile physical interfaces using multiple sensors. The IVES engine allows for the rapid design of visual representations of XRMIs and virtual environments, as well as the spatialization of sound. Their integration into Max opens up far-reaching possibilities for sound synthesis and algorithmic composition in a domain-specific language that is widely used in the community and already familiar to many users.

The combination of these tools enables many possibilities for virtual works, allowing artistic ideas and concepts to be realized that would be difficult or impossible in physical reality. To remain within the scope of this paper, the possibilities in the context of XRMIs and virtual interactive installations are presented below. However, the possible applications of the system can be used beyond these areas as well as the examples described.

4.1 XRMIs

The system described can be used to implement VMIs for various forms of virtuality (Sec. 1). The possible scenarios and use cases are manifold and can be realized in any number of gradations and hybrid combinations. The XRMIs developed in IVES with SPINE-based physical interfaces can be used, for example, for concert performances. With IVES, the visual representation for the performing artist can be displayed on an HMD with binaural rendering over headphones, while at the same time a projection with

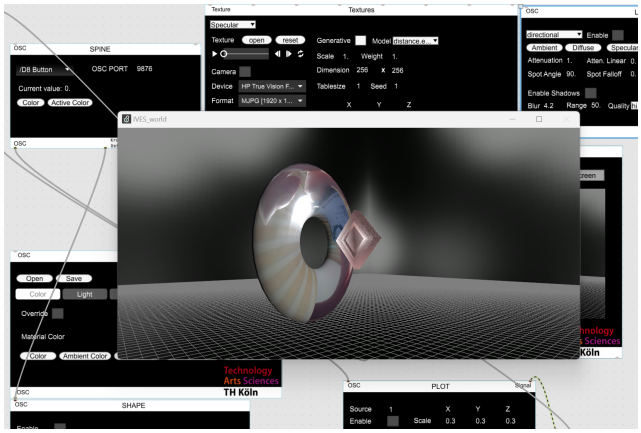

Figure 8: Example XRMI developed with IVES.

loudspeaker-based audio is used for the audience. Audio-only XRMIs using the spatial audio rendering in IVES can also be realized for performances or installations.

Thanks to various input options, such as audio/video streams or external tracking and motion capture systems, IVES allows the realization of MR applications or other hybrid formats. Integrating camera streams enables passthrough AR on HMDs, e.g. for augmented hyper-instruments. Integrating environmental sensors (e.g. light and temperature) as well as real-time audio/video streams allow to link virtual objects with the real environment for the realization of AVMIs. The implementation of the sound synthesis, which is probably the most critical element of an instrument in the context of music, is left to the user and the various possibilities that Max offers for this task. A wide variety of sound synthesis and generative composition algorithms can be implemented, as well as sample and sequencer instruments. All standard and compatible Max libraries and extensions can be used for this. The resulting audio can be spatialized in IVES as any number of virtual sound sources.

The XRMI interface shown in Fig. 8 serves as an example for the development of XRMIs in IVES and its connection to physical interfaces using SPINE. An interface floating in space was implemented, consisting of a torus and an audio-reactive quadrangular object moving through the hole of the torus. The torus, realized with the IVES SHAPE module, contains a real-time video as texture. This integration of the real environment via video and the SPINE interface to the virtual environment creates an exemplary AVMI application. The audio-reactive object, implemented with the IVES PLOT module, reacts to the result of the instrument's audio, which is additionally spatialized as a virtual sound source, whose position is assigned to the 3D object. In Max, arbitrary implementations of instrument sounds can be realized with synthesis algorithms or sample playback linked to the sensor values provided by the SPINE module. Sound synthesis is not included in the IVES engine and not limited by it, as any audio can be routed to spatialized sound sources. Using the IVES WORLDXR module and the spatial audio DECODER module in binaural mode, the resulting AVMI can be experienced on VR HMDs or presented to an audience via projection using the conventional IVES WORLD module and the DECODER module in Ambisonics mode.

The physical SPINE interface (Fig. 9) consists of a button, a slider and a rotary sensor. The button controls a color change of the audio-reactive 3D object. By linking the button to parameters of the sound synthesis, a change in the

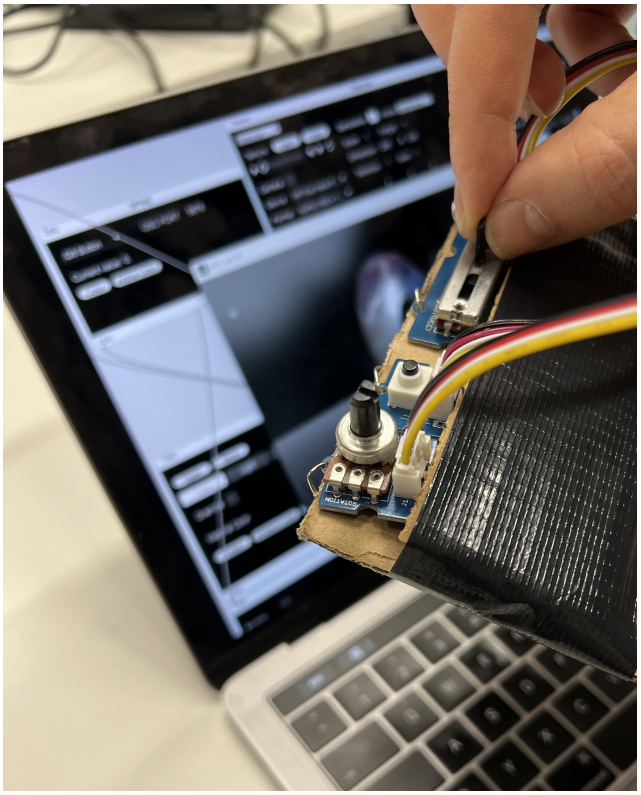

Figure 9: Prototype SPINE interface for the example XRMI.

sound also triggers a simultaneous change in the shape of the audio-reactive object. This way, the color and shape of the object can be changed along with sound.
The slider is used to move the audio-reactive object, which is also assigned to the position of the spatialized sound source. By moving the slider, the 3D object and the sound source can be moved back and forth on a plane in space through the hole in the torus.

The torus can be rotated around the Z-axis using the rotary sensor. By pointing the camera providing the input for the texture at the performer's hands and the interface, a connection is created between the real objects and their virtual representation, which can be visually altered by the rotation. At the same time, the rotation can also be used for sound synthesis, e.g. to parameterize frequencies of oscillators/filters, or attributes such as the gain of a microphone input or the output of a sound source.

The simple mapping of these two exemplary virtual objects to an interface with 3 haptic sensors already shows a variety of possibilities for combining interaction, sound synthesis, and visual and auditory representation of XRMIs with IVES and SPINE. Even the simple virtual elements selected, show the potential for instrument interfaces that would be unthinkable or very difficult to realize in physical reality: Torus-shaped displays for displaying videos are not commercially available, and moving an audio-reactively transforming sound object through the display's hole is likely very difficult to realize. Beyond this example, a variety of possibilities for XRMIs and artistic concepts can be imagined and realized through complex combinations of various sensors and unlimited virtual objects, as well as the blending of real and virtual elements.

4.2 Interactive Virtual Installations

Another use case with strong parallels to musical instruments is interactive installations. In various forms, such as spatial installations and sonic or audiovisual sculptures, interactive installations raise the same issues of interaction, sound synthesis, and audiovisual representation as musical instruments. Unlike stationary instruments, which are usually very locally bounded, installations can be very spacious in their presentation and interaction, and can include the local environment and objects within it. In the context of XR, interactive installations can be realized in IVES for HMD-based systems, with fully virtual environments in VR or hybrid MR forms like AR and AV. AR and AV allow the merging of real and virtual environments and create opportunities for hybrid concepts. Physical SPINE interfaces can enable haptic interaction with virtual objects, the development of AR installations by integrating real rooms with sensors distributed throughout the space, or the augmentation of virtuality in AV installations through SPINE environmental sensors such as light and temperature. Virtual installations can not only be realized on HMD-based systems, but also in physical augmented reality. Using the IVES modules described in Sec. 4.1, physical spaces can be augmented with virtual objects using projection mapping over objects and room elements, and spatial sounds can be reproduced using loudspeaker arrays.

5. CONCLUSIONS & FUTURE WORK

By connecting IVES and SPINE through the developed interface module, we created a solution allowing artists to rapidly develop virtual representations of XRMIs and corresponding physical interfaces. Pre-built no-code/no-soldering modules allow the rapid creation of virtual objects, environments and sound fields, as well as custom sensor-based interfaces for interaction. The customizability of the modules allows experienced users to modify and extend them to meet advanced requirements. The integration with Max not only simplifies these customizations, but also provides extensive possibilities for the user to implement sound synthesis and generative algorithms. The combination of IVES and SPINE introduces features and capabilities not found in previous state-of-the-art solutions (Sec. 1) and making them freely available to artists as open-source software.

Due to the characteristics and possibilities of IVES in the context of XR and the variety of different sensors provided by the SPINE toolkit, a large number of possible implementations for XRMIs and interactive installations (Sec. 4) can be imagined and realized, especially in hybrid forms, combining real and virtual elements. At the time of writing, IVES interface modules for various sensor types have already been implemented (Sec. 3), and these are to be extended in future work by further sensor types from the SPINE toolkit.

The SPINE interface module for the IVES engine presented in this paper is now part of the IVES toolkit, which is available at the following link: `https://github.com/AudioGroupCologne/IVES`

6. ACKNOWLEDGMENTS

The work was realized as a part of KreativInstitut.OWL. It is a project in research cooperation of the OWL University of Applied Sciences and Arts, Detmold University of Music and University Paderborn, funded by the Ministry of Economic Affairs, Industry, Climate Action and Energy of the State of North Rhine-Westphalia, Germany.

We thank Samuel Johnstone for proof-reading this paper.

7. ETHICAL STANDARDS

This work complies with NIME's "Ethical Standards", and describes open-source software without profit intentions, restrictions or conflicts due to the associated funding. It is a continuation of mentioned toolkits and thus not only a further contribution to the open-source community, but also of benefit to the arts and science community.

8. REFERENCES

[1] R. Altosaar, A. Tindale, and J. Doyle. Physically colliding with music: Full-body interactions with an audio-only virtual reality interface. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '19, page 553–557, New York, NY, USA, 2019. Association for Computing Machinery.

[2] Arduino. Arduino Platform. `https://www.arduino.cc`. Accessed: 2024-02-06.

[3] M. A. Baalman. Spatial composition techniques and sound spatialisation technologies. *Organised Sound*, 15(3):209–218, 2010.

[4] A. Boem and H. Iwata. Encounter-type haptic interfaces for virtual reality musical instruments. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1–2, 2018.

[5] T. Carpentier. A new implementation of Spat in Max. In *Proceedings of the 15th Sound and Music Computing Conference (SMC)*, pages 184–191, 2018.

[6] G. Casiez, T. Pietrzak, D. Marchal, S. Poulmane, M. Falce, and N. Roussel. Characterizing latency in touch and button-equipped interactive systems. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 29–39, 2017.

[7] Cycling74. Max. `https://cycling74.com/`. Accessed: 2024-02-06.

[8] D. Dziwis. Revisiting Reynolds - Autonomous Agents for Spatial Audiovisual Composition and Performances. *Proceedings of the AIMC 2023*, Aug. 2023.

[9] D. Dziwis, J. M. Arend, T. Lübeck, and C. Pörschmann. IVES – Interactive Virtual Environment System: A Modular Toolkit for 3D Audiovisual Composition in Max. *Proceedings of the 18th Sound and Music Computing Conference (SMC 2021)*, pages 330–337, 2021.

[10] D. Dziwis, T. Lübeck, and C. Pörschmann. Modular Room Simulation for the IVES 3D Engine. In *Proceedings of the 10th Convention of the European Acoustics Association Forum Acusticum 2023*. Forum Acusticum, 2023.

[11] E. Edwards, J. Rolland, and K. Keller. Video see-through design for merging of real and virtual environments. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 223–233, 1993.

[12] M. D.-C. Florent Berthaut and M. Hachet. Interacting with 3d reactive widgets for musical performance. *Journal of New Music Research*, 40(3):253–263, 2011.

[13] A. Hadjakos and S. Waloschek. SPINE: A TUI Toolkit and Physical Computing Hybrid. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 625–628,

London, United Kingdom, June 2014. Goldsmiths, University of London.

[14] D. Johnson, D. Damian, and G. Tzanetakis. Osc-xr: A toolkit for extended reality immersive music interfaces. In *16th Sound and Music Computing Conference (SMC2019)*, 2019.

[15] M. Karjalainen and T. Maki-Patola. Physics-based modeling of musical instruments for interactive virtual reality. In *IEEE 6th Workshop on Multimedia Signal Processing, 2004.*, pages 223–226, 2004.

[16] J. Lanier. The Sound of One Hand - Moogfest 2016. `https://www.youtube.com/watch?v=ItaPqJaUypY`. Accessed: 2024-02-06.

[17] J. Lanier. Virtual Reality and music. `https://www.jaronlanier.com/vr.html`. Accessed: 2024-02-06.

[18] J. Leonard, C. Cadoz, N. Castagne, J.-L. Florens, and A. Luciani. A virtual reality platform for musical creation: Genesis-rt. In M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad, editors, *Sound, Music, and Motion*, pages 346–371, Cham, 2014. Springer International Publishing.

[19] J. Leonard, N. Castagné, C. Cadoz, and J.-L. Florens. Interactive physical design and haptic playing of virtual musical instruments. In *International Computer Music Conference*, pages 108–115, 2013.

[20] T. Machover and J. T. Chung. Hyperinstruments: Musically intelligent and interactive performance and creativity systems. In *International Conference on Mathematics and Computing*, 1989.

[21] S. Madgwick and T. Mitchell. x-osc: A versatile wireless i/o device for creative/music applications. In *SMC Sound and Music Computing Conference*, 2013.

[22] 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*, NIME '05, page 11–16, SGP, 2005. National University of Singapore.

[23] C. P. Martin, Z. Liu, Y. Wang, W. He, and H. Gardner. Sonic Sculpture: Activating Engagement with Head- Mounted Augmented Reality. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 39–42. Zenodo, May 2021.

[24] P. Metzger. Adding reality to the virtual. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 7–13, 1993.

[25] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D:1321–1329, 1994.

[26] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: a class of displays on the reality-virtuality continuum. In H. Das, editor, *Telemanipulator and Telepresence Technologies*, volume 2351, pages 282 – 292. International Society for Optics and Photonics, SPIE, 1995.

[27] H. Møller. Fundamentals of binaural technology. *Applied Acoustics*, 36(3-4):171–218, 1992.

[28] A. G. Mulder, S. S. Fels, and K. Mase. Design of virtual 3d instruments for musical interaction. In *Proceedings of the Graphics Interface 1999 Conference, June 2-4, 1999, Kingston, Ontario, Canada*, pages 76–83, June 1999.

[29] G. Palma, S. Perry, and P. Cignoni. Augmented virtuality using touch-sensitive 3d-printed objects. *Remote Sensing*, 13(11), 2021.

[30] PatchXR. Patch World. `https://patchxr.com/`. Accessed: 2024-02-06.

[31] M. Puckette. Pure Data : another integrated computer music environment. In *second intercollege computer music concerts*, pages 37–41, 1997.

[32] G. Santini. Augmented piano in augmented reality. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 411–415. Zenodo, May 2021.

[33] Seedstudio. Grove. `https://wiki.seeedstudio.com/Grove_System/`. Accessed: 2024-02-06.

[34] 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.

[35] S. Serafin, S. Gelineck, N. Böttcher, and L. Martinussen. Virtual reality instruments capable of changing physical dimensions in real-time. In *Enactive 2005*, pages 1–7, 2005.

[36] SuperCollider. SuperCollider Website. `https://supercollider.github.io//`. Accessed: 2024-02-06.

[37] L. Turchet, R. Hamilton, and A. Çamci. Music in extended realities. *IEEE Access*, 9:15810–15832, 2021.

[38] G. Wakefield. VR library for Max. `https://github.com/worldmaking/vr`. Accessed: 2024-02-06.

[39] K. C. Zellerbach and C. Roberts. A Framework for the Design and Analysis of Mixed Reality Musical Instruments. In *NIME 2022*, jun 16 2022. https://nime.pubpub.org/pub/mrmi-framework.

[40] F. Zotter and M. Frank. *Ambisonics: A Practical 3D Audio Theory for Recording*. Springer, 2019.