A MODULAR ECOSYSTEM FOR AUDIO-VISUAL AUGMENTED REALITY PERFORMANCES

Henrik von Coler

School of Music
Georgia Institute of Technology, USA
hvc@gatech.edu

ABSTRACT

The modular approach of Linux audio and multimedia systems allows the combination of different software components into an ecosystem with maximum flexibility. This paper describes such a Linux-based system for audio-visual performances with Augmented Reality (AR) interfaces. OSC messages and video feeds are sent from one or multiple Meta Quest 3 devices to a Linux machine. This machine implements a spatialization system in SuperCollider, which creates Ambisonics or binaural output with dedicated spatial sound effects. In addition, a video stream of the devices' first person view is streamed for the audience, using the Android Debugging Bridge and Pure Data. With this combination, the system allows the manipulation of the video stream through the audio input. The proposed approach can be run on a simple machine at reasonable latencies while being robust enough for large-scale live performances.

1. INTRODUCTION

Extended Reality (XR) can be incorporated in live music performances and music production in many ways. Mobile Augmented Relity (AR) allows participatory performances [1] - the audience can interact with the events on stage. Virtual Reality (VR) can be used to realize immersive telematic concerts [2]. Head-mounted displays (HMD) can create virtual and augmented experiences for both the audience and the performers [3]. This technology makes it possible to use AR interfaces next to physical instruments to create a hybrid performance situation [4].

The flexibility of Linux-based multimedia systems allows the combination of various free and open source components. Often times, advanced tasks - like multichannel sound spatialization - can be run on commodity hardware, when properly configured [5]. Although experimental and professional tools and environments - like Pure Data or Max/MSP - can be used to create self-contained audio-visual experiences, additional components are necessary to include XR hardware, like head-mounted displays.

The system presented in this paper is an audio-visual AR performance environment. Performers or musicians wear a HMD - more precisely a Meta Quest 3 - on stage to explore new possibilities in Human Computer Interaction (HCI) during a musical performances. Two principles are central to the proposed approach:

I **Control:** A core function of the system is to offer means for expressive control of computer music systems through AR interfaces. These interfaces can be placed next to physical instruments, thus creating a hybrid setup for new possibilities.

II **Transparency:** Another central aspect of the proposed system is the ability to stream the first person view of the musicians to the audience. Especially in electronic music performances, this can enhance the connection between performers and audience.

The system described in this paper has initially been deployed in human-computer interaction (HCI) research [4] to evaluate the general usability of AR interfaces next to physical instruments. In its recent version, the system is used for a live electroacoustic performance with double bass and modular synth [6]. Both musicians are equipped with a HMD to control digital audio effects and stream their first person view.

Aspects of user experience and the creative possibilities have been described in the above cited publications in detail. This paper aims at giving a detailed introduction into the workflow and tools used in this modular AR performance system, highlighting best practices, pitfalls and strengths. It should allow readers to replicate, change and expand on this approach.

2. TECHNOLOGY

The components and basic signal flow of the proposed system are visualized in Figure 1. Audio and video processes are shown in separate blocks. They can either run on the same physical machine or on two individual computers. One or multiple Meta Quest 3 send OSC data to SuperCollider [7], where it is used for audio effects and spatialization and forwarded Pure Data [8] for visual processing. The Ambisonics channels from SuperCollider are sent to an external decoder via JACK [9].

Crucial steps and aspects are explained in the remainder of this section. Section 2.1 contains details on the video part of the system, while the audio part is covered in Section 2.2. All scripts and configuration files for the *2CUBES* performance, using two AR headsets, can be found in the dedicated repository[1].

2.1. Video

A key aspect of the proposed system is the ability to make the first person view of one or multiple performers visible to the audience, with the additional option to manipulate the video in real-time. The official options to show the camera feed from the Meta Quest 3 headset include[2]:

- Cast to the Meta Horizon app. This only works for Android and IOS receivers and is not suitable.
- Cast to a browser via WiFi (`oculus.com/casting`). This method relies on an internet access to establish a connection between the glasses and the the receiver. For a real-time performance system used at different venues, this is not acceptable.

Since none of the official methods meets the requirements of this project, multiple software components are combined to stream video from the Meta Quest to a projector via a Linux machine. The key dependencies involved are *Android Debug Bridge (adb)*, *scrcpy*, *Video for Linux 2 (V4L2)*, and *Gem* in *Pure Data (Pd)*. Each tool plays a crucial role in capturing, processing, and outputting the video stream:

1. **adb**: Establishes communication between the Meta Quest and the Linux PC, enabling screen capture via scrcpy.

[1] `https://github.com/L42i/2CUBES`

[2] `https://www.meta.com/help/quest/192719842695017/`

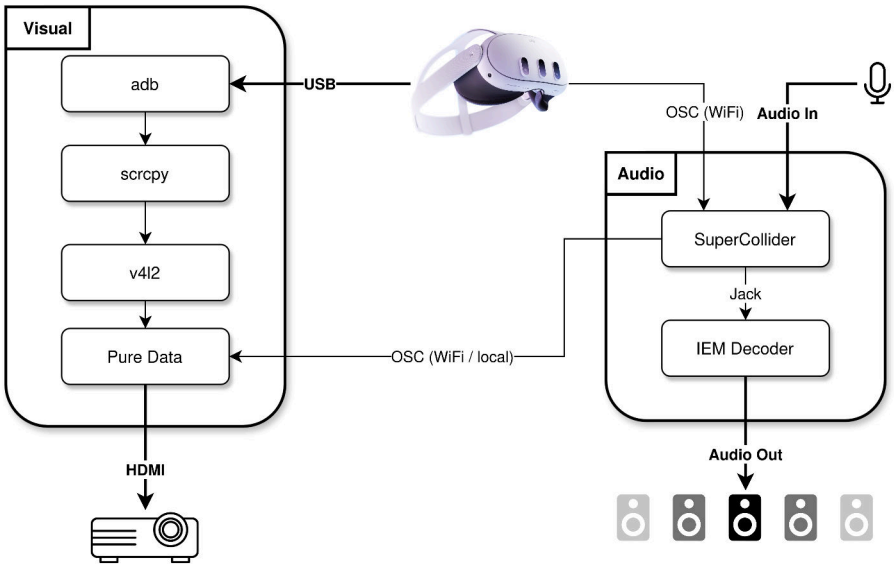

Figure 1: Components of the audio-visual system with signal flow.

2. **scrcpy**: Uses adb to mirror the Meta Quest display onto the PC, providing a real-time video feed.
3. **V4L2 and v4l2loopback**: Creates a virtual video device, allowing the scrcpy stream to be accessed as a camera input.
4. **Gem (Pd)**: Processes the video stream for artistic manipulation and projection.

2.1.1. The Android Debug Bridge (adb)

The Android Debug Bridge (adb) [10] is a versatile command-line tool, designed to allow communication between a development machine and an Android device or emulator. As part of the Android Software Development Kit (SDK), adb is essential for deploying applications, debugging, and accessing the Android system at a low level. It is used to optimize performance by adjusting system parameters and automating testing procedures through scripting.

The Android Debug Bridge operates through a client-server architecture, consisting of three components: the `adb` client, which runs on the development machine; the `adb` daemon, which runs on the target Android device; and the `adb` server, which manages communication between the two. `adb` provides critical functionalities such as shell access for executing commands on the device, file transfer capabilities, and port forwarding for debugging networked applications.

As part of the proposed system, `adb` establishes the connection between the HMD and the Linux machine. Each Android device is associated with an Android ID, a unique 14-sign alphanumeric string. Throughout this document, the placeholder `0123456789ABCD` will be used as a placeholder for an arbitrary ID. Android devices, respectively Meta Quests, connected via USB can be listed with:

```
$ adb devices
```

For a newly connected Android device, this command gives the following output:

```
List of devices attached
0123456789ABCD  no permissions (missing udev
    rules? user is in the plugdev group); see [
    http://developer.android.com/tools/device.
    html]
```

In this case, the Android device needs to be authorized, which is done on the device itself. Right after connecting to the computer via USB, the Quest will show a prompt in a popup window:

```
USB Detected
Click on this notification to allow
connected device to access files.
```

After clicking on this prompt, the user gets the choice to *Deny*, *Allow* or *Always allow from this computer*. Even when selecting the latter option, a single click confirmation is necessary every time the device is reconnected to the computer via USB. The output of `adb devices` for an authorized device will give:

```
List of devices attached
0123456789ABCD  device
```

2.1.2. The v4l2loopback Module

Video for Linux 2 (V4L2) [11] is the Linux kernel's multimedia framework for handling video capture and output devices. It provides a standardized API for accessing video hardware, supporting webcams, TV tuners, and other video input sources. V4L2 offers extensive control over video formats, frame rates, and device parameters, making it a powerful tool for real-time video processing. In this project, V4L2 is essential for capturing and processing live video feeds from the Meta Quest. It allows seamless integration of the camera streams into arbitrary applications, ensuring low-latency performance critical for live music environments. Additionally, V4L2 works in conjunction with modules like `v4l2loopback` to facilitate virtual video devices for software-based video manipulation.

The **v4l2loopback** [12] kernel module provides a virtual video device that allows users to create loop-back video streams in Linux. It is commonly used to route video output from software applications to virtual camera devices, making it accessible to other applications that require camera input. For the proposed video solution, `v4l2loopback` enables seamless integration of the AR visuals by allowing software-generated imagery to be treated as a camera feed. This is particularly useful for real-time video processing, streaming, and interactive visual performances. The module supports multiple virtual devices, configurable frame formats, and resolution settings, making it highly adaptable for different performance environments.

The v4l2loopback kernel module has to be loaded, either temporarily or permanently. With `modprobe`, loading the module takes additional arguments to specify the number of devices and their labels. This example prepares two virtual devices with individual labels:

```
$ sudo modprobe v4l2loopback devices=2 video_nr
    =10,11 card_label='QuestA','QuestB'
```

2.1.3. scrcpy

scrcpy [13] is a lightweight, open-source application that provides low-latency screen mirroring and remote control of Android devices from a desktop computer. It operates via the Android Debug Bridge (adb) and does not require root access, making it a highly accessible tool for developers and performers. scrcpy supports high-resolution video streaming, low-latency input forwarding, and features such as clipboard sharing and screen recording. Its efficiency, with minimal latency and resource usage, ensures that visual feedback remains synchronized with live audio and interactive elements.

Within the proposed AR performance system, `scrcpy` enables real-time visualization and interaction with AR applications running on Android devices. `scrcpy` can be launched once the adb devices have been connected and authorized and after creating the virtual video devices with V4L2. The tool offers several command-line arguments to adjust the stream quality. After a heuristic tuning, the following settings delivered a clear image without introducing significant latency:

```
$ scrcpy -s 0123456789ABCD --v4l2-sink=/dev/
    video10 --bit-rate 20M --max-fps=30 -N
```

2.1.4. Gem / Pure Data

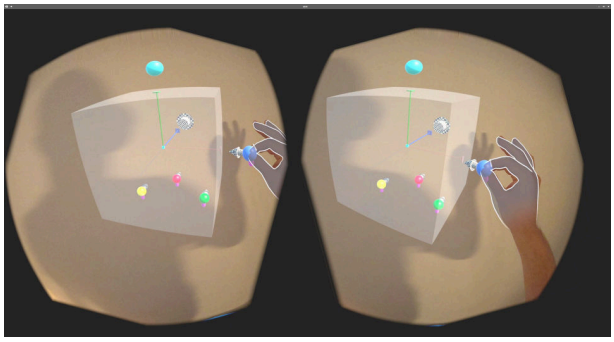

Figure 2: Stereoscopic stream from the Quest 3 in Gem.

The Graphics Environment for Multimedia (Gem) [14] is an external library for Pure Data (Pd) that enables real-time graphics rendering, particularly for video and OpenGL-based visuals. By leveraging OpenGL, it supports real-time rendering of textures, shaders, and 3D objects, enabling immersive visual experiences synchronized with musical elements. Gem extends PD's capabilities by providing objects for generating and manipulating 2D and 3D graphics, making it a powerful tool for interactive audiovisual performances.

Gem is the final rendering stage in the visualization approach presented in this paper. Figure 3 shows part of a PD patch for grabbing the video from a V4L2 loop-back device, using the object `pix_video`. An inlet of this object allows the selection of a v4l2loopback device. The stream contains a stereoscopic view. The left and the right eye's image are isolated, scaled and rotated with Gem objects to result in the image shown in Figure 2. OSC receivers in the Pd patch allow the dynamic manipulation of visual elements based on the live audio inputs in SuperCollider.

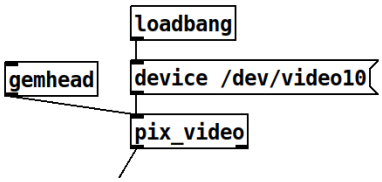

Figure 3: Grabbing video from a v42l loopback device in PD.

2.2. Audio

The modular approach of the audio processing and rendering is based on the JACK Audio Connection Kit [9]. Figure 4 shows all audio connections in QjackCtl [15], with the signal flow going from left to right. This exact routing is used for the 28 channel setup in the 3DBox[3] of the Lab for Interaction and Immersion (L42i). Audio is sent to SuperCollider (SC) from the hardware inputs for processing and Ambisonics rendering. The encoded signal is sent to a decoder and finally to the hardware outputs. In addition, SC extracts audio features from the inputs and sends them to Pure Data for a processing of the video stream.

Figure 4: JACK audio connections in Qjackctl.

2.2.1. Audio Effects and Ambisonics Encoding

Audio effects and spatialization, respectively Ambisonics encoding, are implemented in SuperCollider. For Ambisonics encoding, the project has been running with the SC-HOA [16] extensions since its beginning. Recent versions now use the *ATK for SuperCollider* [17], due to more frequently updated repositories and extended rendering capabilities. Live inputs from instruments are treated with different effects and then distributed in space. This process is in detail explained in a dedicated publication [6].

2.2.2. Ambisonics Decoding

Designing decoders for the SC-HOA extension in SuperCollider involves several steps and additional software (Faust, Octave)

[3] `https://l42i.music.gatech.edu/projects/3dbox`

and is this not easy to use, especially when decoders are needed for changing performance venues. While the ATK allows the design of Ambisonics decoders with fewer steps, completely inside SuperCollider, the use of external decoder software offers even more flexibility. The standalone version of AllRADecoder from the IEM plugin suite [18] has a flexible and intuitive way of designing decoders with a graphic interface. As a JACK client it can be fed with the encoded Ambisonics stream from SuperCollider.

2.2.3. Audio Connection Management

JACK connections are managed with `aj-snapshot` [19]. In this workflow, connections have to be set manually for a new project, respectively a new configuration, once. They are then stored in an XML file, when taking a snapshot. Once connections are stored, `aj-snapshot` can be launched in daemon mode to auto-connect what has been stored. The auto-connect option in `jackd` has to be disabled for this behavior (using the `-a a` flag).

3. SYSTEM PERFORMANCE AND HANDLING

First performances with the described system were realized on two machines: one for the audio and one for the video part. In later stages, a single machine was used for both domains: A *GEEKOM MiniPC* with a 12th Gen Intel(R) Core(TM) i7-12650H.

Audio relies on plain JACK, after deactivating PipeWire. This was found to be more reliable in terms of connection handling. A JACK server is booted with a real-time priority of 95 and a buffer size of 256 samples (n=2) at 48 kHz. The resulting latency allows for smooth real-time interaction without significant xruns. Different audio interfaces have been used, each performing well with these server settings.

The gain in flexibility with the modular software system comes at the price of an increase in management effort. Several processes have to be launched in the right sequence and might need to be changed during runtime. Two `tmux` [20] sessions are used to cope with this - one for the audio part and one for the video part. All components, including `jackd` are launched in individual tiles. Each program can be relaunched smoothly for development and debugging, with aj-snapshot taking care of connections. This approach has proven to be highly reliable and deterministic.

4. CONCLUSIONS

Overall, the proposed audio-visual system can be considered a professional tool for various applications in research and creative practice. It is important to note that this is just one possible solution. Any component may be replaced to improve certain aspects. The audio part of this system has been applied and tested repeatedly. It is reliable, efficient and delivers high quality results. Although PipeWire has been excluded at this stage of the project, in might well be a solution for a complete audio-visual routing system.

A major drawback of the proposed video solution is the USB connection for streaming, especially since the AR interface only uses WiFi to send OSC. With high quality cables of 20 feet length, this issue is negligible for many instruments. In addition, it extends the battery life. Without battery add on, the HMDs can be drained quickly when rehearsing, followed by a performance. However, for free movement ant the aesthetics of the performance, the cable should be replaced with a wireless option to stream the video.

While there are alternative ways to design a similar system, also using only a single piece of software, the modular approach is able to combine different tools, each with their individual strength. The use of command line tools like `tmux` might scare unexperienced users off in the first place. However, once mastered, this paradigm offers advanced opportunities for control and maintenance.

5. REFERENCES

[1] Anna N. Nagele, Valentin Bauer, Patrick G. T. Healey, Joshua D. Reiss, Henry Cooke, Tim Cowlishaw, Chris Baume, and Chris Pike, "Interactive audio augmented reality in participatory performance," *Frontiers in Virtual Reality*, vol. 1, 2021.

[2] Damian Dziwis and Henrik von Coler, "The entanglement: Volumetric music performances in a virtual metaverse environment," *Journal of Network Music and Arts*, vol. 5, no. 1, pp. 3, 2023.

[3] Luca Turchet, Rob Hamilton, and Anil Çamci, "Music in extended realities," *IEEE Access*, vol. 9, pp. 15810–15832, 2021.

[4] Hyunkyung Shin and Henrik von Coler, "Arcube: Hybrid spatial interaction for immersive audio," in *Proceedings of the 2024 ACM Symposium on Spatial User Interaction*. 2024, Association for Computing Machinery.

[5] Fernando Lopez-Lezcano, "Towards open 3d sound diffusion systems," in *Proceedings of the International Computer Music Conference (ICMC)*, 2014.

[6] Hyunkyung Shin and Henrik von Coler, "2CUBES: A Multi-User Augmented Reality Performance," in *Proceedings of the International Computer Music Conference (ICMC)*, 2025.

[7] James McCartney, "SuperCollider: a new real time synthesis language," in *Proceedings of the International Computer Music Conference (ICMC)*, 1996.

[8] Miller S. Puckette, "Pure Data," in *Proceedings of the International Computer Music Conference (ICMC)*, 1996.

[9] Davies, Paul, "JACK API," `http://www.jackaudio.org/`, 2020, Accessed: 2025-03-31.

[10] Android Developers, "Android debug bridge (adb)," https://developer.android.com/tools/adb, 2025, Accessed:2025-03-27.

[11] Linux Kernel Organization, *Video for Linux 2 (V4L2) API Documentation*, 2024, Accessed: 2025-03-30.

[12] Vadim Umrikhin and Contributors, "v4l2loopback - a kernel module to create v4l2 loopback devices," 2024, Accessed: 2025-03-30.

[13] Romain Vimont, "scrcpy (screen copy)," https://github.com/Genymobile/scrcpy, 2025, Accessed: 2025-03-30.

[14] IOhannes M. Zmölnig, "Gem - graphics environment for multimedia," https://github.com/umlaeute/Gem, 2025, Accessed:2025-03-30.

[15] Rui Nuno Capela, *QjackCtl*, 2025, Accessed: 2025-03-31.

[16] Florian Grond and Pierre Lecomte, "Higher order ambisonics for supercollider," in *Proceedings of the Linux Audio Conference*, 2017.

[17] Olivier Warusfel, "Atk for supercollider," `https://github.com/ambisonictoolkit/atk-sc3`, 2025, Accessed: 2025-03-31.

[18] Institute of Electronic Music and Acoustics (IEM), "IEM Plug-in Suite," https://plugins.iem.at/, 2025, Accessed: 2025-03-30.

[19] Lieven Moors and Jari Suominen, "Aj snapshot: A tool for capturing and analyzing data," https://aj-snapshot.sourceforge.io/, 2012, Accessed: 2025-03-30.

[20] Nicholas Marriott, "tmux," https://github.com/tmux/tmux, 2025, Accessed: 2025-03-30.