

BandSimLIVE: Design and Implementation of a VR-to-MIDI Drumming Interface

JACEY SCHELL,*  AES Member
(jxs7684@miami.edu)

Department of Music Engineering Technology, Frost School of Music, University of Miami, Coral Gables, Florida, USA

This project introduces an alternative music performance and recording technique through a virtual reality experience titled “BandSimLIVE.” While the ultimate artistic goal of my work is to develop a multi-user virtual space for music creation without the need for physical instruments through hand-gesture recognition, the initial prototype focuses on a single-user, controller-based drumming experience. I limited my objective to creating a “proof-of-concept” single-user experience centered solely around playing the drums utilizing Meta Quest controllers in order to focus on the technical aspects and needs of the project. By embedding MIDI components directly into Unity, I was able to map specific hand positions, velocities, and collision events to the corresponding MIDI note values and velocities, thus creating a custom virtual drum kit that responded to the user’s movements. This proof-of-concept represents my intentions and future plans with BandSimLIVE efficiently and accurately.

Keywords: Virtual Reality, MIDI, Musical Interfaces, Gesture-to-Sound Mapping, Real-Time Audio, Unity

1 INTRODUCTION

1.1 Project Overview and Scope

This project introduces an alternative music performance and recording technique through a virtual reality experience titled “BandSimLIVE.” While the ultimate artistic goal of my work is to develop a multi-user virtual space for music creation without the need for physical instruments through hand-gesture recognition, the short-term aim was simplified in order to meet deadline requirements. I limited my objective to creating a “proof-of-concept” single-user experience focused solely on playing the drums using Meta Quest controllers in order to focus on the technical aspects and needs of the project.

1.2 Technical Objectives

To support this technical approach, I implemented a wide variety of tools and techniques including the following:

- Custom Signal Processing - transforming location-based data into MIDI triggers along with processing virtual instrument packs for added realism

- MIDI 1.0 Features - the sending/receiving of MIDI between Unity and Ableton
- Alternative Controller - mapping MIDI note values based on the locations of Meta Quest’s hand-held controllers
- Synchronization - real-time data flow communication between Unity and Ableton
- Live Performance Setup - built for live and real-time music creation via virtual reality
- Interactive Elements - virtual drum kit that triggers MIDI notes when spatially interacted with
- Real-time Manipulation - live velocity-reactant drum kit to simulate dynamics

2 BACKGROUND AND INSPIRATION

2.1 Embodied Musical Instruments

A large inspiration for this project was the WAM Jam Party presented at the Web Audio Conference 2025 by Michel Buffa, Marco Winckler, and Amad Mir-sadjadi [1]. WAM Jam Party is a multi-user virtual reality experience that allows users to play instruments via Web Audio Modules.¹ The team behind WAM Jam Party recently developed a 3D drum kit containing collision-based MIDI triggers that are sent to a Web Audio Module drum sampler. Although

* To whom correspondence should be addressed, email: jxs7684@miami.edu.

¹ Modules similar to plug-ins created for web-based audio/MIDI processing.

WAM Jam Party is a web-based VR experience and Band-SimLIVE is developed in Unity, the concept of “interfaces designed to replicate real-world instrumental interaction,” better known as embodied musical instruments, stuck out to me [1]. Not only was this idea innovative but also educational for those wishing to learn existing physical instruments who might not have the resources or abilities to do so.

3 SYSTEM ARCHITECTURE

3.1 Signal Flow and MIDI Routing

Figure 1 shows the basic signal flow diagram for Band-SimLIVE, depicting how the Meta Quest hand position data would reach my DAW of choice, Ableton, as MIDI messages. I considered making use of a custom Max device, but, since my goal was to minimize latency for eventual multi-player use, I wanted to limit the amount of systems that my data would run through. As a result, I utilized a .NET library titled DryWetMIDI² which allowed for the transformation of hand position and velocity data into MIDI triggers directly via C# scripts, as shown in Listings 1 and 2, in Unity and a virtual MIDI port.



Fig. 1. Basic signal flow diagram from the planning stage illustrating the hand-to-MIDI process.

3.2 Hardware and Software Environment

The technical specifications for my project are as follows:

- Unity 6.2 6000.2.0f1
- Ableton Live Suite 12.2.5
- Meta Quest 3
 - Headset and Controllers
- “Corvaire Kit” virtual instrument

4 IMPLEMENTATION

4.1 Gesture-to-MIDI Mapping

By embedding MIDI components directly into Unity, I was able to map specific hand positions, velocities, and collision events to corresponding MIDI note values and velocities, thus creating a custom virtual drum kit that responded to the user’s movements. This approach also allows for greater flexibility in the future iterations of the project dealing with custom behaviors all handled within C#.

4.2 Ableton Integration

Although my Ableton session was reasonably simple with just a singular MIDI instrument track receiving the

Listing 1. Constructing and sending a MIDI Note On message from MidiOut script.

```

public void NoteOn(int noteNumber, int
    velocity = 100)
{
    if (outputDevice == null)
        return;
    var noteOnEvent = new NoteOnEvent((
        SevenBitNumber)noteNumber, (
        SevenBitNumber)velocity)
    {
        Channel = (FourBitNumber)
            midiChannel
    };
    outputDevice.SendEvent(noteOnEvent);
}
  
```

Listing 2. Mapping a normalized value to the MIDI range and sending a Control Change message in MidiOut script.

```

public void SendCC(int ccNumber, float
    normalizedValue)
{
    if (outputDevice == null)
        return;
    int midiValue = Mathf.Clamp(Mathf.
        RoundToInt(normalizedValue * 127f),
        0, 127);
    var ccEvent = new ControlChangeEvent((
        SevenBitNumber)ccNumber, (
        SevenBitNumber)midiValue)
    {
        Channel = (FourBitNumber)
            midiChannel
    };
    outputDevice.SendEvent(ccEvent);
}
  
```

data from my scripts, I hope to expand on this in tandem with the multi-user concept by developing an entire multi-track session.

4.3 Real-Time Performance Considerations

Focusing on the backbone technicalities of this project allowed me to further optimize the project for live performance by minimizing buffer sizes, reducing Unity script complexity, and prioritizing real-time processes to reduce latency.

These decisions were crucial in terms of reaching my final goal of a multi-user experience with player-to-player audio/MIDI synchronization without sacrificing playability.

4.4 Virtual Instrument Construction

As previously mentioned, the artistic goal of my work is to develop a multi-user music creation tool in Unity’s virtual reality environment. The implementation of virtual reality adds a layer of required visual and aesthetic consideration. For example, a priority in my project was having instruments as 3-Dimensional objects in the virtual space

² <https://github.com/melanchall/drywetmidi>

so that users can interact similarly to how they would in the real world. I searched the Unity Asset Store³ for a drum kit that I could place into my Unity project but, unfortunately, there were none for free and I wasn't willing to pay money for my proof-of-concept just yet. This meant that I had to build the drum kit myself in my Unity project making use of Unity's basic 3D shapes: Cube, Sphere, Capsule, Cylinder, and Plane. By implementing mostly cylinders and capsules, I built my low-poly drum kit from scratch as shown in Figure 2.

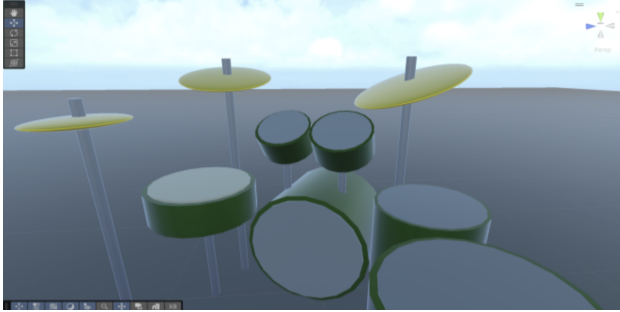


Fig. 2. Self-made low-poly drum kit out of Unity's cylinders and capsules.

4.5 Sound Selection and Spatialization

Likewise, in order to provide users with a fairly realistic drum-playing experience, I chose the Corvaire Kit virtual instrument sound pack on Ableton which resembles an acoustic drum kit. As seen in Figure 3, I also panned each drum relative to how it would sound for a user sitting at the drumset for increased realism and immersion.

5 DEVELOPMENT CHALLENGES AND LIMITATIONS

5.1 Hand Tracking and Hardware Constraints

My initial idea, which is also my final goal for BandSimLIVE, was to utilize motion-tracked gloves rather than the Meta Quest controllers. Since I don't have current access to such gloves, I decided to make an attempt at using Meta Quest's built-in hand recognition system. I faced the most challenges at this stage revolving around C# scripting, lack of access to the Meta Quest headset, and a time crunch due to the scope of the project.



Fig. 3. Panning of drum components to match player's perspective.

³ <https://assetstore.unity.com/>

5.2 Controller-Based Prototype

I unfortunately wasn't able to resolve the issues that I was facing, but I did figure out how to incorporate the Meta Quest controllers as a replacement option. This is shown in Figure 4. Although it's not what I hope for in my future final iteration of the project, I was able to continue working on my proof-of-concept using the controllers.

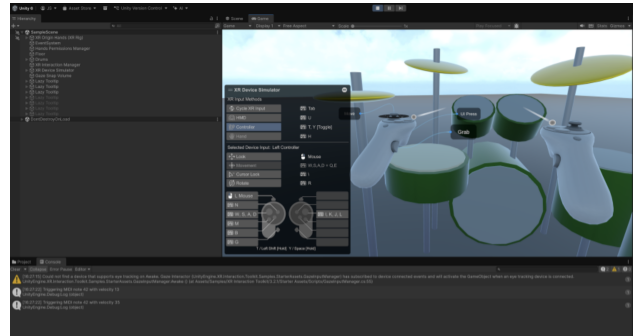


Fig. 4. Unity project of BandSimLIVE showcasing use of controllers, bindings, and triggered output in the console.

6 FUTURE WORK

6.1 Instrument Expansion and Multiplayer Interaction

I am ultimately looking forward to developing my project further. I plan to increase the instrument options including adding MIDI pedals for the kick drum and hi-hat, incorporate a network component for multi-user capability, and implement either Meta Quest's built-in hand-tracking or motion capture gloves for real-time haptic feedback, such as in Figure 5.



Fig. 5. "MiMU Gloves" designed for human-computer interaction and haptic feedback responses for future possibilities of BandSimLIVE.

7 CONCLUSION

BandSimLIVE could open up a world of possibilities for music creation and collaboration because of its freeplay ability in contrast to many of the multiplayer VR music

experiences out there already, such as Band Space VR⁴ and Beat Saber⁵, which fall under the pre-programmed “rhythm game” genre. Additionally, BandSimLIVE replicates existing instruments to lessen the learning curve between physical and virtual instruments, rather than games like Virtuoso⁶, which incorporate more abstract music-creation tools.

This proof-of-concept represents my intentions and future plans with BandSimLIVE efficiently and accurately.

8 SUPPLEMENTAL MATERIAL

BandSimLIVE’s prototype and proof-of-concept is open-source and can be downloaded at <https://github.com/jacesch/BandSimLIVE>. A four-minute walkthrough and demo can be found at <https://youtu.be/k-ZakG0LS3U>.

9 ACKNOWLEDGMENT

This work has been completed as a requirement of two courses that are part of the Music Engineering Technology Department curriculum at the Frost School of Music, University of Miami: Music in the Metaverse- Dr. Tom Collins, and Electronic Production Techniques- Prof. Susie Green.

10 REFERENCES

[1] M. Buffa, M. Winckler, and A. Mir-Sadjadi, “Embodied Virtual Instruments in Web-Based Multi-User VR: A Case Study with a 3D Drum Kit and Web Audio Modules,” in *Proc. WAC 2025-9th Web Audio Conference* (2025), <https://doi.org/10.5281/zenodo.17741327>.

THE AUTHOR



Jacey Schell

Jacey Schell is an audio engineer and DSP-focused developer pursuing a B.S. in Music Engineering Technology with a minor in Computer Engineering along with an M.S. in Music Engineering Technology. Her work spans real-

time audio processing, ML-driven systems, creative DSP, and immersive sound.

⁴ <https://www.meta.com/experiences/band-space/>

⁵ <https://beatsaber.com/>

⁶ <https://virtuoso-vr.com/>