UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

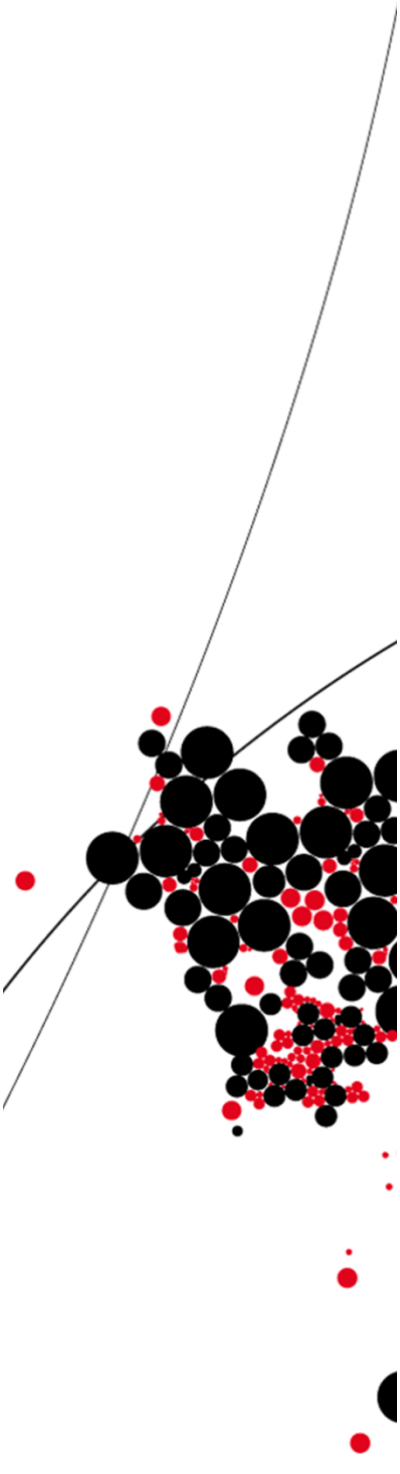
Sound Swarm

Experience sound in a new way

**The realisation of a composition tool for
a virtual version of Sound Swarm**

Rens Kruining

**Creative Technology BSc Thesis
July 2017**



Supervisors:
dr. ir. E. C. Dertien MSc

Client:
Christine Maas

**Creative Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands**

Abstract

Sound Swarm is the concept of an art installation in which audio speakers move around in a room. Each of the speakers play a different sound and by moving create a new way of experiencing audio and music. This project entails the realisation of software to compose spatial music using virtual moving speakers. To do so, the existing technology is studied and several user requirements are gathered during brainstorms with the client. The realised composition tool combines the virtual environment in the game engine Unity with the digital audio workstation Reaper. The routes of the audio sources are configured in Reaper in three dimensions, which Unity then uses to move the audio sources through the virtual room. This is done with the aid of a MIDI bridge, where Unity receives MIDI control change messages from Reaper.

While this composition tool is lacking a few detailed functions due to the concept phase the project is still in, the tool proved already enough for basic testing. The features that can be added later, are mostly the aesthetics of the virtual environment and speakers. Because the location and speakers are not determined yet, the detail of the virtual objects was omitted. It will be beneficial to the immersion of the virtual installation for the client to add more details to the virtual environment.

Acknowledgement

I would like to thank Edwin Dertien for his help, supervision and insights during this graduation project. Many times, Edwin has provided guidance for encountered problems and situations. Without his help, this project would have ended up quite differently. I would also like to thank Christine Maas for this opportunity to be part of her research process, her support and enthusiasm in the project. This project has given me the chance to work on a system that will be used in further research and has led to a new experience.
As other students have helped me on many occasions by helping with certain issues or supporting me in working on this report, I would like to show my gratitude to all of them as well.

Table of Contents

List of Figures

1. Introduction

Christine Maas is a Dutch artist who wants to develop an art installation to experience audio in a new way. To help her with a concept ide for this new art installation, she contacted the University of Twente. The installation will have listeners experience sound and music in a novel manner, by dividing the audio sources over multiple speakers in a room. The installation will be referred to, in this report, as *Sound Swarm* ("Geluidszwerm", the original Dutch name).

The name originates from the concept idea, where the installation consists of multiple speakers in a room, that can move in any direction. These moving speakers would then, during a performance, create a swarm of sound. This "Swarm" can be compared to a group of birds, moving around as an autonomous cloudlike creature. The birds in this comparison would be replaced by the speakers with each its own individuality in the form of a unique audio source.

The aim of the installation is to create a new experience as the sound is no longer a 'static' whole, but a range of effects on individual parts created by the movement of the speakers. The audience will be placed inside the room, so that this swarm moves around them. This will create an experience as if the sound, produced by the speakers, envelops the audience. As speakers get closer to or further away from the audience, different parts of the sound will draw the attention.

As this study continues on the earlier work done by Wouter Westerdijk [1], who has researched the design of *Sound Swarm*, this report will focus on the composition tools of the installation. To steer the speakers, the composer needs to be able to give directions to the software moving the speakers. Continuing the work, a virtual reality will be used for the art installation, to compose performances and test the experiences created with these compositions. This should enable the artist to produce the best and/or most unique experience of being inside a sound and being able to listen to individual sources in their spatial location.

Upon finalizing this report, several conclusions must be made. The software used to visualize the virtual art installation as well as the software to compose the audio parts and their spatial location should be optimized to create an understandable and intuitive interface. These two major components must also cooperate in a correct way to create the same experience as a composition would create with the physical installation. These requirements should provide an answer to the main question of this project:

How to design a tool to compose spatial music using virtual moving speakers?

2. Analysis

This chapter starts with the background research concerning virtual spatial sound and editors and composition tools for this. As the goal of this project is to design a composition tool, 2.3. states the first design of this tool, as suggested by the supervisor of this project and considered feasible. The rest of this chapter analyses the components of this structure.

2.1. Background Research

There are many aspects to spatial audio in VR, especially with the use of headphones. 2.2.1. focuses on the most important aspects to make the VR experience as real as possible, making it as comparable to the physical setup as possible in terms of audio. This research is substantially based upon a research review conducted by myself in [2] concerning *Sound Swarm*.

2.1.1. Spatial audio in VR

To have a realistic experience of spatial sound in virtual reality, it is essential to be aware of the elements of audio perception outside of VR. Next, 2.2.1.2. will discuss the fundamentals of translating virtual audio over headphones.

2.1.1.1. Three-Dimensional (3D) audio perception

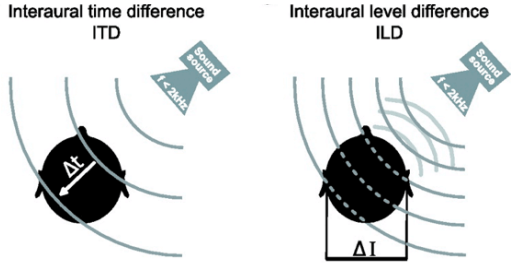


Fig. 1. Binaural Cues. Source:[12]

The art installation works on the basis of the human hearing and listeners being able to perceive distinctions between different positions in relation to the space and the listener. There are many aspects that cause this effect of 3D audio perception, enabling humans to distinguish sound-locations in both a horizontal (in front, at a side or behind) and a vertical plane (below or above). The distinction between left and right is a result of the property of sound that it needs time to travel as indicated by Willert et al. [3]. Since humans have two ears, a signal coming from an angle will arrive later at the furthest ear than the other ear. Even though this difference in time is quite small, the human mind is able to locate the source quite precisely. This binaural cue is called Inter-aural Time Difference. Another difference that the mind is able to pick up is Inter-aural Level Difference. This is the effect of the sound having a (slightly) lower volume and frequency at the ear facing away from the source. As the soundwaves are blocked by the head, a loss of high frequencies occurs. Since both differences become very small when the source is almost right in front or right behind the listener, it becomes very hard to precisely locate it.

Another reason why we are capable of 3D audio perception, is the shape of our ears. Due to the shape, the listener can perceive the vertical location as well as the difference between sound sources in front and behind them. Stated by Willert et al. in [3], “incoming sound waves at the pinna(shell of the ear) are first filtered in the outer ear that has filter characteristics with a bandpass-like transfer function.” The sounds are filtered in specific ways (as depicted in fig. 2), which the mind has come to translate to specific vertical localisation. Besides these monaural and binaural cues, the sound also changes because of surrounding reverberation. The sound bounces on the environment near the listener, resulting in delayed and slightly changed sounds arriving at the ears at different angles than the sound directly from the source. This makes it possible for humans to even locate audio sources behind objects.

2.1.1.2. Virtual 3D audio and HRTF

To make the correct translation from a physical room, with the listener among moving loudspeakers, to a virtual set-up, the step towards headphones must be made. As mentioned by Matsui et al. [4], “Sound systems using headphones are easier to set up and less expensive than systems using loudspeakers. But the sound images of an ordinary stereo sound signal are localized internally when one listens through headphones”. With

headphones, the effects of the shape of the ear are bypassed and the effects need to be computed by the software. This causes no problems in terms of horizontal localization, as the software utilizes the proper interaural level difference and interaural time difference to improve the coherence of the sound images, as notioned in the workd of Brown and Duda [5]. However, the vertical perception seems to differ a lot between listeners, which makes the effects very difficult to control. Another problem that rises with headphones is that the sounds in front often appear to be too close, and reversals between front and back are not uncommon. These problems are caused by variations in the shape of the ears of listeners. As each ear is shaped differently, each person localizes sound differently.

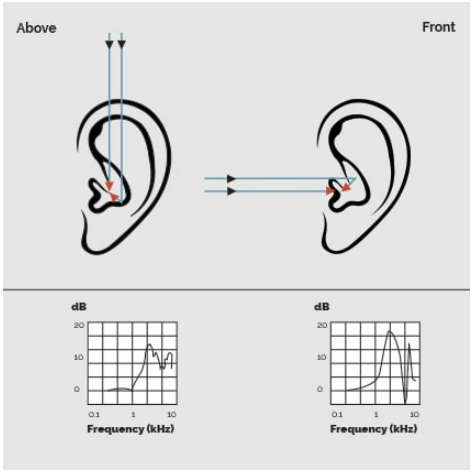


Fig. 2. Ear filtering. Source:[10]

The shapes of the ears have been analyzed in many ways to counter these problems. According to the work by Brown and Duda [5] and Nguyen et al. [6], include putting microphones in the ears of the listener and playing sounds from a large amount of speakers in an anechoic room. The speakers are arranged in a spherical array around the listener. For the measurements, each speakers plays a sound which is then caught by the microphones. The differences in recordings and the played sound is then analyzed to personalize the effects used with the headphones. This process takes a lot of time as the sound has to be played once for each speaker. During the time of the measurements, the listener cannot move and everything else in the room has to be quiet (no background noises as people talking or electronics humming). Another way of doing these measurements, is by replacing the speakers with microphones and the microphones by tiny speakers. Because of all these past measurements, there are now large databases with Head-Related Transfer Functions (HRTF) which have been used to create generic HRTF that are used by most recent software.

Another aspect of audio over headphones is that the speakers move with the head of the listener, where as the sound sources should not. Head tracking makes it possible to compensate for that, moving the virtual audio sources correspondingly, so that they appear to be in the same position relative to the listener's position. According to Shissler et al. [7], it is important to minimize the latency in head-tracking and audio/visual processing. This greatly increases the immersive properties of the Virtual Reality experience and reduces the difference between VR and a physical set-up with similar audio sources. Most common VR software has mastered this and enables this graduation project to focus on the experience itself.

It is also important to have the aural image match the visual image as "for professional sound designers, a mere 4° offset in the horizontal plane between the visual and aural image is perceptible, whereas it takes a 15° offset before the average layperson will notice.", as stated by Kyriakakis [8]. In the project, this means the audio imaging that is conveyed over the headphones should correspond with the visual image without any delay.

2.1.2. 3D audio software

Since virtual reality is a fast-growing field, as well as 3D audio, many different tools can be found to simulate a virtual audio environment that is comparable to a physical setup. As described by Devana [9] in April 2015 there were five leading 3D audio plugins. Each of them with different unique features and pricing, but all easily implemented into Unity. It lists the plugins and compares them by distinguishing their strengths and differences in terms of capability, computing power requirements and implementation. The plugins included have changed quite a bit since the publication of the article, resulting in some of them becoming unavailable for this project or

changing in pricing. Of the plugins listed, only the Oculus Audio SDK[1] and RealSpace3D[2] are still available, of which only Oculus provides the plugin for free.
A recommended free Unity plugin is Steam Audio[3], which integrates seamlessly and effortlessly into the VR setup in Unity. It was released in February 2017 after the company Valve had acquired the audio plugin company Impulsonic and It addresses all aspects of 3D audio and enables to user to experience the art installation in virtual reality.

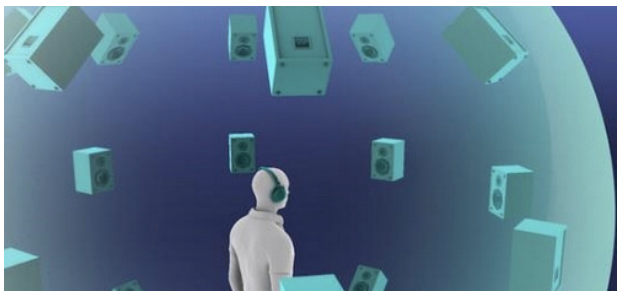


Fig. 3. 3D audio in VR. Source: [8]

2.1.3. 3D sound headphones

To prevent possible flaws of dedicated software to simulate three-dimensional audio over stereo headphones, Ossic has developed headphones that have multiple drivers and head-tracking hardware built in. The Ossic X[4] calibrates to the user's head and ear features, increasing overall sound quality and ensuring the most accurate sound placement. The headphones have sensors within the headphones to set the correct HRTF and create the virtual audio space specific to the user. The downside of using this headphone for this project is that they are a lot more expensive than a stereo headphone.

2.2. Similar projects and relevant technologies

As virtual three-dimensional audio imaging has been done before, these technologies will be evaluated for the inspiration that can be drawn from them. There are also several projects that use a game engine for an art installation. As this is closely related to *Sound Swarm*, these projects will be explored as well.

2.2.1. Projects

Several projects use virtual spaces in combination with spatial sound synthesis for unique experiences.

MagNular

Described online by Dolphin [10],"*MagNular* is a sound-toy for one or many players. A variety of particle objects are selected and dropped into a virtual room by the player(s). Each of the 15 types of particles available has simulated physics behaviours and represents a different sound type. Virtual magnets are used by the players to attract or repel the particles, allowing them to be freely moved around the room, resulting in collision events that trigger and transform sound." With *MagNular*, players change the behaviour of particles around their game object, causing sounds to be created in a 3D virtual space. The user "composes" sounds by controlling and influencing simulated physical behaviours. Andy Dolphin aimed to provide an interactive virtual sound installation. For this project, Unity3D is used in combination with an external sound engine developed within Max/MSP/Jitter[5] as stated by Dolphin [11].

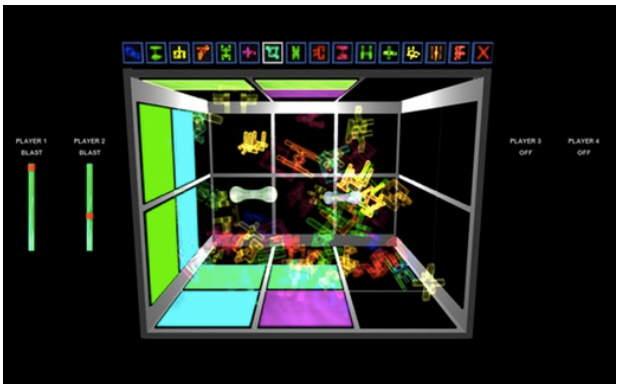


Fig. 4. MagNular (2009) by Andy Dolphin. Source: dysdar.org.uk

[1] Oculus Audio SDK: https://developer.oculus.com/audio/
[2] RealSpace3D: http://realspace3daudio.com/
[3] Valve's Steam Audio: https://valvesoftware.github.io/steam-audio/
[4] Ossic's 360 audio headphones Ossic X: https://www.ossic.com/technology/
[5] Cycling '74 Max/MSP/Jitter or Max: https://cycling74.com/products/max/

Cyclical Flow

Another “sound-toy” by Andy Dolphin [12], which makes use of a multi-channel audio systems, is Cyclical Flow. The animated user of Cyclical Flow interface controls both spatial parameters as well as synthesis parameters. It generates sound which the user sets to follow a path in a virtual space. The sounds are then played in the physical performance space using 8 (2D) or 24(3D) channels. Andy Dolphin makes the dynamic movement of sound through space a central theme in this interactive installation, where the changes in the virtual space directly influence the sounds played by the physical speakers set up in the room. For this project, Andy Dolphin developed a game engine himself and uses Max/MSP/Jitter for the construction of the synthesis engine for the spatial sounds.

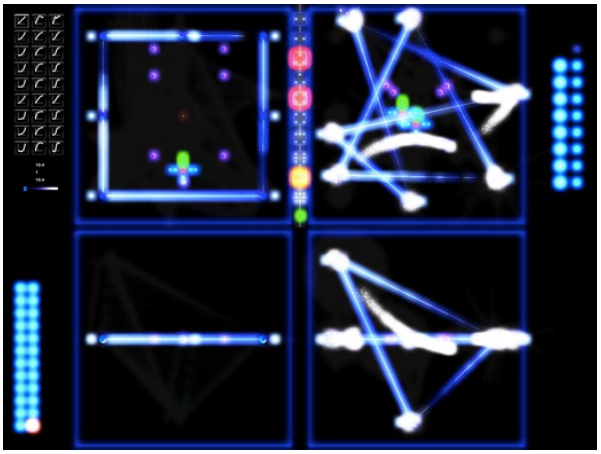


Fig. 5. Cyclical Flow (2014) by Andy Dolphin. Source: dysdar.org.uk

Other “sound toys” by Andy Dolphin

Andy Dolphin has executed multiple projects and several focus on audio-visual or audio effects. Dolphin calls five of these projects his “sound toys” and they are all interactive installations. Some aim for a combination of visual effects and sound synthesis, where others utilize the experience of spatial audio.

2.2.2. Technologies

There are several different relevant software that spatialize audio. In this subsection, some 3D tooling technologies are discussed.

Rondo360 - https://dysonics.com/rondo360/

This software by Dysonics allows the user to place any number of audio sources in a 3D virtual space using a simple interface. *Rondo360* then translates these positions to the audio system, ranging from headphones to complete surround sound systems using many different speakers in a room. Its spatial positions are static to the virtual environment, moving only in respect to the user through head-tracking. They do not move through the environment, which keeps the interface simple on one hand, but limits the usability, on the other hand, for projects like *Sound Swarm*.

Dolby Atmos for Virtual Reality - https://www.dolby.com/us/en/professional/content-creation/vr.html

Much like *Rondo360*, Dolby has focused on the virtual placement of audio sources and this software allows the user to position the audio in virtual environments. *Dolby Atmos* concentrates on the playback on headphones, as often used with VR, and offers great precision with a low computing power usage.

Spatial Audio Designer - http://www.newaudiotechnology.com/en/products/spatial-audio-designer/

This audio tool is used for creating content and monitoring in surround and 3D. It enables users to mix for several speaker configurations and headphones using virtual sources. The software then maps the audio to the configured physical speakers and can create both 2D and 3D spatial audio. It supports importing DAW automation data and uses this for panning and volume automation.

3DEV

3DEV is a GNU software developed for the creation, transformation and temporal coordination of multiple directional sound source trajectories in a three-dimensional space, according to [13]. It displays the path of an audio source in a four window-screen, with a top view, front view and side view, as well as a 3D view. Another screen shows the path in two editable envelopes indicating the azimuth and elevation angles, together with the audio signal’s waveform. This way, the orientation of the sound source can be accurately synchronized with the audio signal.

Zirkonium

Zirkonium III is asset of Mac OSC software tools to aid the composition of spatial music, as explained in [14]. The software allows to user to set out a trajectory for audio sources, which are mapped on the configured speaker set. Since all components are connected, the trajectory editor can display the audio waveform as well as the set path (as shown in Fig. 6). For *Sound Swarm,* it inspires the concept of introducing a path editor instead of using the automation tracks in a digital audio workstation. It provides the possibility to display of the paths that the speakers will follow as well as eliminating the issues that could arise due to the bridge between the two software and the computing power needed for running the two programs. The concept requires quite a bit more coding and tinkering, but would enable the composer to use only a game engine to realize the virtual installation.

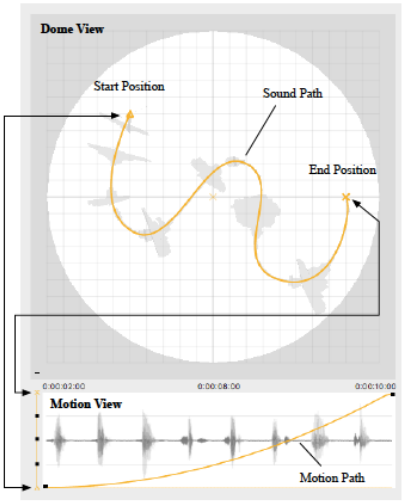


Fig. 6. Zirkonium III sound and motion path. Source:[14]

2.3. Structure

Most game engines can work with audio files on their own, but would still need a module that records and plays the movement of the speakers. Since audio software can already do recording and playback of many different formats, one of the possibilities is using that capability and making the game engine work with it.
The following diagram (fig. 7) displays the structure of the composition tool and components connected to it. From left to right shown: the input devices for the speaker paths, the Digital Audio Workstation(DAW), the communication-bridge between the DAW and the game engine, the game engine, and the head mounted display and headphones.

Fig. 7. Structure overview

2.3.1. Setting the speaker paths

In this set-up, the composer uses physical input devices (as depicted in fig. 8), such as MIDI consoles or touchscreen devices, connected to a pc. These devices will send the information, used for the paths of the speakers, to the digital audio workstation. The aim is for these devices to be as responsive and intuitive as possible, while allowing the user to control multiple variables at the same time. This is not possible in most audio editing software, or with the use of a digital cursor, but is essential to easily alter or set the paths of speakers. This will initially involve three values, namely the height of the speaker, the “x-position” and the “y-position”. The latter two can be seen in a top-down view as the distance from the west-wall and the distance from the north-wall respectively. This is chosen as the initial concept as these values are the most common techniques for setting a position and movement in three dimensions. This control may need to change later according to tests.

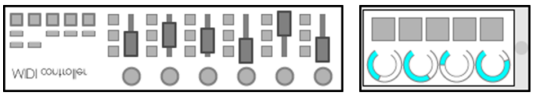


Fig. 8. Physical input devices

2.3.2. Digital Audio Workstation (DAW)

The DAW (fig. 9) is the software in this concept that the composer uses to set the audio for the individual speakers as well as the placement and movement of them. This software can record the audio with the use of instrument or microphones connected to the pc or use pre-made audio files. The movement can be recorded with the use of the aforementioned input devices. In the digital audio workstation, the composer can also alter the audio tracks themselves.

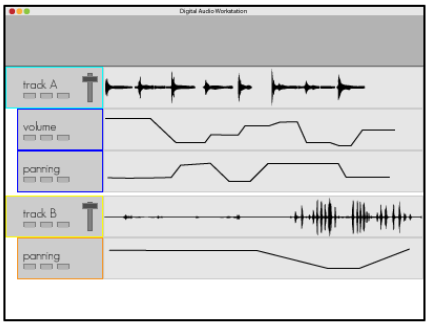


Fig. 9. Digital Audio Workstation

2.3.3. Communication bridge

To have the virtual reality react to the paths set in the digital audio workstation, they must be communicated to the game engine. This bridge will initially be set up so that it transfers the data in real-time, enabling the composer to make alterations during the playback and see the immediate effect.

2.3.4. Game engine

The game engine (fig. 10) is used to create a virtual reality and dynamically change it during the playback of the sounds configured in the DAW. In the game engine, the physical requirements and restrictions can be set up to design for the physical installation. Here the room of the physical installation can also be replicated to have the virtual reality experience as realistic and close to the one of the physical installation as possible.

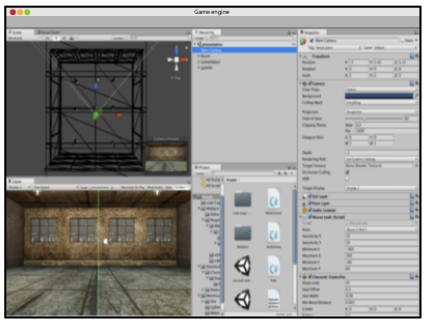


Fig. 10. Game engine

2.3.5. The display and audio

The virtual reality should be experienced similarly to the physical installation, which can easily and cheaply be done with a head mounted display and headphones (fig. 11). The display can involve the more expensive virtual reality HMDs like the HTC Vive[6], or using a smartphone in a head mount like the google cardboard[7]. These enable to user to look around the installation and, in combination with headphones and an audio plugin, experience the virtual installation in a similar way to experiencing the physical installation. This involves being able to look around and have the sounds stay in their corresponding virtual positions, as opposed to moving with the orientation of the listener.

The composition tool that will be designed will be built upon several principles. The design is determined by the available software for virtual reality (VR), audio software and the bridge between the two. These components will be discussed in 2.2. As there are many aspects to spatial audio in VR, a background research in 2.3. will examine the aspects that are essential to the structure of the tool.

As problems may arise during the design of the compositions tools for *Sound Swarm*, it is fundamental to discover earlier similar projects and comparable technologies that might have run into issues. The developers of these systems might, in some cases, have come up with solutions for the problems. Some other problems might be solved by recent technology as this field is fairly new. In 2.4. the report will consequently be covering the state-of-the-art, including the comparable projects and relevant technologies.

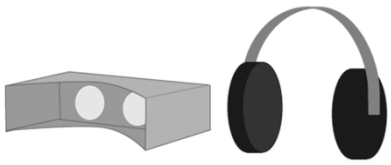


Fig. 11. Head mounted display and headphones

[6] HMD Vive developed by HTC: https://www.vive.com/eu/
[7] head mount for smartphones developed by Google: https://vr.google.com/cardboard/

2.4. Virtual reality and audio software

Since the art installation *Sound Swarm* uses audio, it would be injudicious to leave audio software unexamined. Audio software can handle multiple audio tracks, enable to user to alter these sources and even generate sounds. *Sound Swarm* uses individual sources with each different movement. To code a program in virtual reality software to enable the tasks at hand, would take a great amount of time and can be avoided by just having a bridge between VR software and audio software.

2.4.1. Virtual reality software

There are many different development tools available for VR applications. Some of them are more advanced than others, some are more focused on businesses. Fortunately, most of them have a large community behind the software and a vast number of tutorials and help on the internet. Here follow four of the best rated and free tools according to Kraft [15]:

Unreal Engine[8]

"The Unreal Engine is very well known in the games industry. This package is incredibly versatile, allowing for creation of games from 2d hand drawn looking platformers up to cinematic almost movie like experiences. They've charged into virtual reality head-on and support the latest technologies natively. There is a built in marketplace where you can find and purchase assets to include in your projects and a very large community sharing tutorials and inspiration." [15]

Unity[9]

"Over the last several years, Unity has grown from a plucky little start-up to go toe to toe with the likes of the Unreal Engine. The upcoming release of the first major commercially available VR headsets has only helped level the playing field as Unity has been aggressively courting this community. You can download Unity and begin building VR environments immediately with no prior experience." [15]

Cryengine[10]

"The Cryengine has long been known for its rich visual abilities, the flagship games from this engine often being used as benchmarks to determine a computer's strength."[15]

Lumberyard[11]

A game engine brought by Amazon, making use of their cloud services and direct twitch integration. This is a fairly new engine that still has a building community. Even though the games in development with Lumberyard are quite big and ambitious, there are only a handful of them.

Since I already have some experience with Unity, that will be the first one I will try to develop with. This is the most time efficient, preventing me to learn a new software before encountering problems. If I were to encounter fundamental problems with bridging, I will consider the other tools.

2.4.2. Audio software or Digital Audio Workstation (DAW)

Many different audio software have been around for several years and are being used for many different purposes. The initial tests will involve free software or free versions, which are listed below.

Presonus Studio One 3 Prime[12]

A DAW first released in 2009, with the help of several former developers from another older audio editor. Its free version has unlimited audio and MIDI tracks and an elegant single-window work environment with powerful drag-and-drop functionality could provide an easy interface for the composer.

---

[8] Epic Games' Unreal Engine: https://www.unrealengine.com/
[9] Unity Technologies' Unity3D: https://unity3d.com/
[10] Crytek's CryEngine: https://www.cryengine.com/
[11] Amazon's Lumberyard: https://aws.amazon.com/lumberyard/
[12] Presonus' Studio One: http://www.presonus.com/products/Studio-One

Cockos Reaper[13]
A digital audio workstation with a vast amount of capabilities and a free fully featured version makes it a viable software choice. It allows for a lot of customisation, which is useful for this project.

Avid Pro Tools | First[14]
A limited free version, but used by many beginners and can thus be considered an option for the composition tool.

Apple GarageBand[15]
A free Apple DAW, which restricts it to Mac OS, but can be tested in the early phases as its easy setup and friendly user interface enable for fast testing and early results.

Ardour (GNU General Public License)[16]
Ardour is an open source free software with many features and capabilities, making it a good alternative to limited or paid digital audio workstations.

Steinberg Cubase[17]
A DAW that was originally released in 1989, and has been updated with major changes since, still remains a relevant software. It now has a collection of the many different features developed during the years and is capable of tasks ranging from beginning projects to professional editorial work. It's free trial versions allow for the early tests to determine the relevance of the software to this project.

Apple Logic Pro X[18]
Another Mac OS only DAW, developed by Apple, is considered as one of the top DAWs of the world. It's dedication to the OS for years has led to a vast base of compatible interfaces. However, the lack of a free (trial) version and high costs make it an ill-favoured piece of software for the early tests. It can be considered for a later stage of development if the need arises.

The initial tests on the audio software, described in 2.6.2. will test the compatibility of the listed software and conclude with a choice for further testing and development.

2.4.3. Bridge between audio software and game engine

One of the formats digital audio workstations can work with, is MIDI (Musical Instrument Digital Interface). "MIDI carries event messages that specify notation, pitch and velocity, control signals for parameters such as volume, vibrato, audio panning, cues, and clock signals that set and synchronize tempo between multiple devices.", as stated on Wikipedia [16]. These signals can in our case also be used for the movement of the audio sources. For Unity to work with MIDI, a plugin[19] will be used that can read and process a real-time input. To send the MIDI-information from the audio software to the game engine, a virtual MIDI-port will be used, enabling both to run on the same computer.
Another protocol used in with electronic musical instruments and software, is Open Sound Control (OSC)[20]. This is a protocol that is optimized for modern networking technology. The protocol should be flexible and easy to implement while providing everything needed for real-time control of sound and other media processing. This protocol, however, needs a slightly more extensive implementation compared to MIDI, as it sends messages that are undefined. These must be set by hand on both ends. OSC as a communication bridge requires a plugin[21], just like it is the case with MIDI, for Unity to work with it as an input.

---

[13] Cockos' Reaper: http://www.reaper.fm/index.php
[14] Avid's Pro Tools: http://www.avid.com/pro-tools-first
[15] Apple's GarageBand: https://www.apple.com/mac/garageband/
[16] open-source DAW Ardour: https://ardour.org/
[17] Steinberg's Cubase: https://www.steinberg.net/en/products/cubase/start.html
[18] Apple's Logic Pro X: https://www.apple.com/logic-pro/
[19] Github user Keijiro's plugin MidiJack for Unity: https://github.com/keijiro/MidiJack
[20] Open Sound Control: http://opensoundcontrol.org/introduction-osc
[21] Github user Jorgegarcia's plugin UnityOSC for Unity: https://github.com/jorgegarcia/UnityOSC

The digital audio workstations listed in 2.2.2. have the following capabilities concerning these protocols, based upon internet research. These capabilities will be tested in 2.6.2.

Digital Audio Workstation	MIDI input	MIDI output	OSC input	OSC output
Studio One 3	✓	✓	×	×
Reaper	✓	✓	✓	✓
Pro Tools	✓	✓	✓	×
GarageBand	✓	✓	×	×
Ardour	✓	✓	✓	✓
Cubase	✓	✓	✓	×
Logic Pro	✓	✓	✓	×

Often, to use OSC, a bridge is used to translate OSC data to MIDI date, where the software then uses MIDI. This means the use of MIDI is more promising as it will be less prone to errors and delays.

2.2.3.1. Virtual MIDI port

To enable the audio software to send information (track automation) to the game engine, a virtual MIDI port will be used. On Mac OSX, this can be done in the operating system itself according to this tutorial[22]. With a Windows operated computer, third party software such as MIDI-OX[23] or virtualMIDI[24] needs to be installed. With such a virtual MIDI port, audio software can be set up to send the information to other software one the same computer. The game engine can then be set up to receive the information from this virtual port.

2.2.3.2. MIDI and OSC comparison

MIDI and OSC are two protocols used to for date concerning audio data and data changes. MIDI sends predefined 7-bit of 14-bit messages whereas OSC sends “user-defined” messages. This means the user must define the messages send, and received, to correspond to the right actions. Since MIDI generally need fewer bytes to make common MIDI messages than it does to make comparable OSC messages, it has a better throughput than OSC. It makes it capable of sending more messages than OSC can within the same time, as mentioned by The MIDI Association [17].
As for accuracy (the number of messages per second) can be set to the same, as MIDI allows for different accuracy settings. These accuracies can be as high as 30 frames per second and one has the choice of 24, 25, 19.97 and 30 frames per second as the frame rate. OSC includes a high-precision timestamp with picosecond-resolution whereas the MIDI beat-clock is a low-resolution clock having a precision on the order of several milliseconds at best, as stated on opensoundcontrol.org[18]. This means OSC messages can be scheduled, recorded and reproduced with minimal jitter.
MIDI has become a standard in audio software and has been around for as long as it has, it is supported by a lot more programs than OSC is. There is a standard file format for MIDI data, too, which has resulted in millions of MIDI files available on the internet and many programs and devices that will play them, as described by The MIDI Association [17].

2.5. User requirements

During an early meeting with Christine, several requirements became apparent. Two different situations are discussed, which both need to be designed for. On one hand, composing itself needs to be possible for a technical novice. Christine has no experience with audio software and game engines. This means all the processes need to be of a low difficulty and well documented. The other situation is that Christine needs to present the virtual experience to attain funds and approval for continuation and realisation of the installation.

---

[22] IAC: Get virtual MIDI ports on a MAC: https://hearandknow.wordpress.com/2010/05/09/iac-get-virtual-midi-ports-on-a-mac/
[23] Jamie O’Connell & Jerry Jorgenrud’s MIDI-OX: http://www.midiox.com/
[24] Tobias Erichsen’s virtualMIDI: https://www.tobias-erichsen.de/software/virtualmidi.html

2.5.1. Intuitive user interaction

To make the use of the composition tool as easy and intuitive as possible, the amount of actions needs to compose need to be as low as possible. There are specific tasks that will not be possible to do in advance, which means the tasks needs to be explained as much as possible and reduced to a minimum.
One of the tasks is setting the number of audio sources and have them correspond between the two programs. When a speaker is added in the game engine, it needs an audio file that is also used the corresponding track in the audio software. To decrease the difficulty for this task, adding speakers and tracks should exclude adding or changing code, setting up MIDI input or output, and preferably linking them.
Another task is recording and playback, and editing the composition, which should involve as few different programs at the same time as possible. Preferably Christine only needs to use the game engine for making the composition after setting up the tracks and speakers.

An additional way of increasing intuitiveness is to document all the steps that need to be taken as well as possible. This also involves a possible troubleshooting guide, for when unintended situations occur.

2.5.2. Virtual experience

As Christine needs to report and attain approval for the realisation of the installation, it is preferable to show a virtual version of a composition. This will be realised with the use of headphones and a mobile phone. With the use of a smartphone's accelerometer, Christine and other users will be able to look around in a virtual reality and listen to the audio in relation to their own orientation.

3. Initial tests

This chapter describes the first tests, experimenting with the suitability of MIDI as the communication bridge and determining the digital audio workstation for the initial setup.

3.1. Unity with a MIDI input

In this early test, Unity is set up with a single ball. The aim of experiment is to use a MIDI input in Unity, using the plugin provided by GitHub user Keijiro[25], to move the ball in the virtual space. Unity was already installed before commencement of this test and a nanoKONTROL2[26] USB controller had been acquired. The test was done on an Apple MacBook Air 13-inch (early 2015) with a 2,2GHz Intel Core i7 processor and 8GB 1600MHz DDR3 RAM memory. The Unity version used is 5.5.2f1 and the plugin downloaded is last changed on 16 January 2016 by the developer. The test took approximately an hour and 35 lines of code (as shown in Appendix A) with a moving ball as a result, controlled with the MIDI controller. The ball was controlled in all three dimensions without an apparent delay or stuttering. As an extension to the test, a second ball was added, which was controlled separately simultaneous with the first ball, without problems and with a minimal alteration to the code. The setup can be seen in the screenshot in Appendix B.

The test was completed within a reasonable amount of time, without apparent future issues or restrictions, and with only a small amount of coding needed. This leads to believe that using MIDI as an input for the game engine Unity seems to be a viable option as initial communication bridge.

3.2. DAW with MIDI input and virtual MIDI output

The next tests aim to see the capabilities of four different digital audio workstations regarding MIDI input and output. All tests are executed on an Apple MacBook Air 13-inch (early 2015) with a 2,2GHz Intel Core i7 processor and 8GB 1600MHz DDR3 RAM memory. Each test involves the nanoKONTROL2 USB controller as input and a digital midi port as output. To monitor the output, MIDI Monitor[27] 1.3.2 is used. The virtual midi port is already set up and selected in MIDI monitor.
The test starts with the software installed and the aim is to record a track of two different sliders from the controller and play this back on the virtual midi port. The test aspects involve only the MIDI implementation into the software. This includes both input and virtual output. The tests are used to determine the software used in further early tests.

3.2.1. Presonus Studio One 3.3.4 build 41933

Just like with Reaper, when making a track, the input has to be chosen as well as the output. Setting up the MIDI input and output both required 4 steps for each task as a separate menu must be accessed. Recording seems to go with the same ease as with Reaper. However, only one of the MIDI controls could be distinguished, whereas multiple sliders were used. Another issue appeared as the received data was not of the same format as expected (and received when using Reaper). Studio One only sent one control-name where at least two different ones should have been received with values between 0 and 127. The values received ranged from -8192 and 4357. This is within the range of data values used for “pitch bend”, but was nog distinguished as that specific message. More work is needed to discover the correct setup. The test was concluded after approximately 25 minutes.

Test	Success
Setting up a track	✓
Setting up MIDI input	✓
Recording MIDI	✓
Accessing MIDI tracks	✓

[25] Github user Keijiro’s plugin MidiJack for Unity: https://github.com/keijiro/MidiJack
[26] KORG nanoKONTROL 2: http://www.korg.com/us/products/computergear/nanokontrol2/
[27] Snoize’s MIDI Monitor https://www.macupdate.com/app/mac/9950/midi-monitor

Setting up MIDI output	✓
Receiving MIDI output upon playback	✓ (but indistinguishable values)

3.2.2. Cockos Reaper 5.35.0

As Reaper is aimed for more customisation, setting up the track requires slightly more work than with GarageBand. The select MIDI input must be chosen after creating the track, which requires another three clicks. For recording, the user needs to arm the track(s) they want to record. To access the tracks, much like with GarageBand, the audio track can be double clicked, where all the MIDI tracks are listed in a drop-down menu, and even indicated if they have been changed. Setting the MIDI output involves clicking a button on the track (which is not immediately apparent) and then selecting the output from a drop-down list. Even the specific output channel can be chosen here. Upon clicking the playback button, the MIDI Monitor receives the entire input. The test was concluded after approx. 20 minutes.

Test	Success
Setting up a track	✓
Setting up MIDI input	✓
Recording MIDI	✓
Accessing MIDI tracks	✓
Setting up MIDI output	✓
Receiving MIDI output upon playback	✓

3.2.3. Avid Pro Tools 12.3.1

Sadly, Pro Tools crashed repeatedly while starting the application, terminating the experiment after 20 minutes of trying.

Test	Success
Setting up a track	×
Setting up MIDI input	×
Recording MIDI	×
Accessing MIDI tracks	×
Setting up MIDI output	×
Receiving MIDI output upon playback	×

3.2.4. Apple GarageBand 10.1.6

As expected, setting up a track in GarageBand is a task of clicking twice. It immediately has the capabilities to record MIDI, and has the midi-tracks separately in predefined controllers. To view the MIDI tracks, one must click twice. However, cycling through the MIDI tracks seems to be slightly harder, as it is only possible by using a cycle button. The tracks are not selectable in a drop-down menu. There also seems to be no capabilities of displaying multiple or all MIDI tracks at the same time.
However, it seems the wrong conclusions were drawn in the earlier research in 2.2.3. GarageBand supports no MIDI output natively and even with the use of third-party programs, it only involves exporting and using MIDI files, whereas this project requires a real-time output.
The test with GarageBand is therefore discontinued and has come to an end after approximately 15 minutes.

Test	Success
Setting up a track	✓
Setting up MIDI input	✓
Recording MIDI	✓
Accessing MIDI tracks	✓
Setting up MIDI output	×
Receiving MIDI output upon playback	×

3.2.5. Ardour 5.8.0

With Ardour, it remained impossible for me to set any automation tracks to be controlled by the MIDI controller apart from the fader/volume control. Since this means the composer can only use the MIDI controller for setting one automation track per audio track, the composer would still have to work a lot with the cursor. After 20

minutes, the test has been terminated as it would require more work and exploration into the software to manage the correct control.

Test	Success
Setting up a track	✓
Setting up MIDI input	×
Recording MIDI	×
Accessing MIDI tracks	×
Setting up MIDI output	×
Receiving MIDI output upon playback	×

3.2.6. Steinberg Cubase 9.0.20

All during this test, it difficult to perceive different automation tracks as it recorded the MIDI controls. The tracks could not be accessed in the 25 minutes the test lasted. As Cubase recorded and send the output to the virtual MIDI port, it also seemed to occupy a lot of the computer's memory and froze several times before managing to stop playback. It is not recommended to use this software with the current settings.

Test	Success
Setting up a track	✓
Setting up MIDI input	✓
Recording MIDI	✓
Accessing MIDI tracks	×
Setting up MIDI output	×
Receiving MIDI output upon playback	×

After a very successful test with Cockos Reaper, further experiments will involve Reaper as the specific digital audio workstation. The DAW succeeded the immediate requirements and is promising for the initial design of the composition tool.

3.3. Reaper communicating with Unity using a MIDI connection

During this experiment, the communication between the game engine and the DAW is tested. To forgo creating an entire recording function in the game engine, it is essential that the connection is successful between the recording power of the audio software and the virtual reality in the game engine. This experiment is executed on an Apple MacBook Air 13-inch (early 2015) with a 2,2GHz Intel Core i7 processor and 8GB 1600MHz DDR3 RAM memory. Each test involves the nanoKONTROL2 USB controller as input and a virtual MIDI port as output set up using Mac OSX's native MIDI Studio.
Unity's version is 5.5.2f1 and Cockos Reaper v5.35 is used as the DAW.
The setup is connected as displayed in the figure 12.

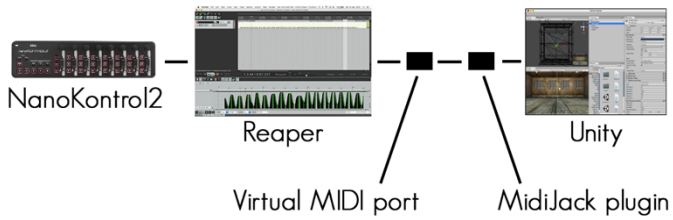


Fig. 12. Setup of initial test

3.3.1. Reaper to Unity

The aim is to navigate a ball object in the virtual space of Unity using MIDI automation tracks in Reaper. It will be done using the first MIDI channel to open possibilities for multiple navigated balls in the later experiments. Initially Unity is set up with the same scenario as used in 2.6.1.

The first step of the test concerns launching Unity and Reaper, as well as plugging in the MIDI controller and setting this up as the input for the track in Reaper. The track is then recorded as a test to make sure everything on the Unity side works correctly. The MIDI output in Reaper is then set to use the virtual MIDI port.
In Unity, the correct scene is opened and the MidiJack plugin monitor window is displayed. Upon playback in Reaper and placing window focus on the monitor, the MIDI control chances appear to be communicated correctly. When playing the Unity scene and pressing playback in Reaper moments later, it is noticed that Unity requires window focus to run. This is fixed after a short and easy change, allowing the user to make changes in Reaper while Unity is playing.
Receiving the MIDI control changes from Reaper, Unity moves the ball in the space according to the set sliders. The ball seems to move as smooth as the sliders had physically been moved. Even two control changes seemingly simultaneous seems to have no impact on the speed and accuracy of Unity.
Upon success of the playback of pre-recorded tracks, Reaper is set to recorded while Unity is running. During this recording, Unity moved the ball while Reaper recorded the control changes, marking another success towards control and usability.

3.3.2. Unity to Reaper

This test examines the possible communication from Unity to Reaper, and whether Unity can be the master software with Reaper as slave, or that Reaper controls Unity's actions concerning playing the audio.
For this test, Unity is set up to receive MIDI Channel 1 Control Change 41 to play and pause, Ch. 1 CC 45 to record and Ch. 1 CC 42 to stop. These are the play, record and stop buttons on the USB controller are therefore chosen for this test.
However, in the used (and current) version of the MidiJack plugin, it is not possible to send MIDI control changes from Unity to the virtual MIDI port. Because of this, Unity can't send commands to Reaper and can therefore not be the Master program.

4. Future concerns

As two programs and a bridge are being used for this setup, some issues can come up. In this subsection, the foreseen problems of the current concept are listed. Based on these concerns, in this early stage it can be decided to change the setup and take a different route towards the end-product.

4.1. Updates

As the development of the programs will not cease in the near future, it is possible the software will receive updates. If later versions of the software are released, it can be important that the user downloads a specific version. Otherwise the *Sound Swarm* software might not function as intended. At the end of this document, in Appendix F, a system overview is included with all software versions on which the project is confirmed to operate correctly.

4.2. Software synchronisation

Since Unity can't use the different audio tracks from Reaper as different audio sources, it has to play the sound files itself. The two programs need to be in synchronisation as the paths, controlled by the automation tracks in Reaper, are tied to the progress of the sounds. To have the two synchronised, one (master) needs to tell the other (slave) when to start playing. The synchronisation imposes some consequences on the capabilities of the whole concept and on the communication needed. Since it doesn't immediately appear possible to send timestamps over MIDI, the programs will always start at the beginning of the audio and the paths. Editing a path in the middle or at the end would be a tedious work. It also means that editing a path over and over with both programs running requires playing the entire composition up to the desired fragment.
It can also become difficult to determine when a source has to start, as that might take a separate MIDI message that needs to be recorded in the DAW. When a specific track start later than the others, it can either contain a silent audio track, or start later. If the audio file contains a silent part, both Unity and Reaper can start playing the files at the same time. However, if the audio file start playing later in Reaper, Unity needs to know when to start it as well. This can be done by a single MIDI message at the beginning of each audio track, but this needs to be coded for, and may perhaps require an extra action from the composer.

4.3. Communication bridge

Being dependent on the MIDI bridge can cause some issues and raises some concerns. As MIDI is a format with pre-defined messages, the functionality of the concept is limited. It can also cause for a delay between the two programs, bringing them out of synchronisation. If the setup has multiple speakers which each have their own two or three automation tracks with all continuous control change messages being send to through the MIDI bridge, it may be possible that the virtual port or the receiving Unity can't process them all, causing either a loss of accuracy and resolution or a delay between the two software.

4.4. Working with a DAW

This concept makes use of a DAW to record, save and play back automation tracks. However, Unity can only display the current position of the speakers and use what it receives from Reaper in real-time. The user needs to play back the recorded movement to see what it looks like, and can only get a momentary view. If Unity where to use paths for the audio sources, they could visualize the movement of the speakers during the entire composition more clearly. The combination of the DAW and Unity also requires the aforementioned synchronization and the communication bridge between the two. Not only do the two software in their entirety need to play the audio at the same time, but do the tracks also must correspond to the right audio sources in Unity.

4.5. Additional implementation plans

As development of the current concept comes with several concerns, it is important to look at the other possible setup of making the program in Unity. As everything can be coded into Unity, several advantages come to mind. No synchronisation and bridge is necessary between software, which means there is no liability or delay source outside of Unity. Making it in Unity also enables the visual aspect of paths and a user interface that would perhaps be clearer. However, it would take a lot of coding and a lot of time. Problems might arise during the coding and functionality might not be possible as intended.

As the Reaper-Unity combination seems very feasible, this project will follow the current course. The next steps are handling the synchronisation between the two, recording the movement efficiently and user-friendly, and testing the delay that might exist due to the MIDI bridge.
Further experimenting will be done in different controller types and other user and system requirements.

4.6. Additional requirements

Upon further contact with Christine, more requirements became clear. Christine would like to use the Virtual Reality to test and create a visual experience to showcase the installation before it has been build.

4.6.1. Volume control

For this, she would also like the ability to change the volume of sources. This can be achieved using another CC that will be received by Unity to control its volume per source. A master volume will also be implemented, using the same method, if this is possible in Unity.

4.6.2. Large amount of speakers

In discussing the possible setups that can be generated, it became clear that it might be possible a large number of sources (50+) will be in the setup. To accommodate for this amount, the capabilities of the use of MIDI will be examined. Another method is the inclusion of a flocking algorithm. In this case, one of the speakers would be a leading source that will be controlled by Christine, where a specified amount of other sources follow it like a flock of birds.
The first step towards this behaviour, is the implementation of physics in the movement of the audio sources. They will accelerate to get to a controlled endpoint.

4.6.3. Google VR

To easily and functionally showcase the virtual setup, Christine would like to use a smartphone in a virtual reality head mounted display. To enable this, the next steps are to consider the Google VR SDK. This software development kit for Unity seems promising in terms of showcasing the experience.

5. Realisation

As the previous chapter concluded, development will continue using Reaper as the audio software, Unity as the game engine and virtual MIDI port for communication between the two.

5.1. Synchronisation

Unity uses automation tracks played back by Reaper to move the audio sources, but plays the audio files itself. This means Unity has to start the playback at the same time as Reaper to be in synchronisation. As 2.6.3.3. already mentions, Unity can't send MIDI to other software, and Reaper must act as the master software. Because of this, Unity needs to know when to start playback on the audio sources, so that it is synchronised with Reaper and has the movement take place at the intended moments. This synchronisation therefore exists of three parts: Unity playing the audio files that are also in Reaper, Unity using the corresponding MIDI control changes, and Unity listening to a starting cue before playback of the audio sources. After the development of this, the delay caused by the MIDI bridge and coding in Unity will be measured. This can then be considered in the setup.
The code used by the audio sources in Unity is included in Appendix C.

5.1.1. Audio files

Unity and Reaper are both set up to use the same audio files in a set folder. This is done manually, where later it might be possible to code this into Unity, so it will be added automatically. Then, to limit the playback of the audio to Unity, the track volume lowered to 0% in Reaper.

5.1.2. MIDI control changes

Unity is set up to receive data from the first three sliders on the USB controller, being received as knob 00, 01 and 02. Each source now uses a different MIDI channel. In the inspector panel, it is possible to change this for each source individually as shown in Appendix D. Reaper is then set up with a new track that records the USB controller where the first three sliders are used. Each track sends the MIDI data to the virtual MIDI port using a different channel.

5.1.3. Starting cue

To synchronise the playback of the audio in Unity with the playback of the MIDI control changes, a starting cue message is send from Reaper to Unity using MIDI. For this, two times the control change 29 is send in rapid succession, the first having a value of 127 and the second of 0. This enables the users to restart the playback in Unity by placing the playback cursor back at the start of the tracks. Upon receiving this data, Unity stops the playback and instantly starts the audio source. The interface of is included in Appendix E.
As Reaper can be used to consolidate several separate items to one full length track, in the case Christine wants one speaker to play multiple items with different starting times, she can combine these beforehand and create tracks that all start at the beginning of the entire piece.

5.2. Recording of movement paths in Reaper

To set the movement of the paths, the user can record the USB controller input in Reaper. To visually display the paths, Unity can show the movement in real time. To enable this behaviour, Reaper has to send its input to Unity at the same time as it is recording it. To prevent Unity from using two signals at the same time, Unity needs to ignore the input directly received from the USB controller. With this change in the code of the MidiJack plugin, the Unity will not receive both input streams, which would otherwise possibly have caused a delay or latency in the visualisation.
This is done with the following addition in the code of MidiJack.MidiDriver.cs:

```
// CC message
if (statusCode == 0xb) {
        if (Marshal.PtrToStringAnsi (MidiJackGetEndpointName (message.source)).Contains
("Virtual")) {
                // Normalize the value.
                var level = 1.0f / 127 * message.data2;
                // Update the channel if it already exists, or add a new channel.
                _channelArray [channelNumber]._knobMap [message.data1] = level;
                // Do again for All-ch.
                _channelArray [(int)MidiChannel.All]._knobMap [message.data1] = level;
                if (knobDelegate != null)
```

```
                knobDelegate ((MidiChannel)channelNumber, message.data1, level);          }
}
```
and adding:
```
[DllImport("MidiJackPlugin")]
static extern System.IntPtr MidiJackGetEndpointName(uint id);
```

This change relies on the fact that the virtual MIDI port has a name with “Virtual” in it. Were this to be different, this specific alteration would need a different string in the “Contains” function.

5.3. Delay in playback

As the movement is part of the playback in Reaper whereas the audio is played back in Unity, the synchronisation is very important. If Unity starts the playback too late, the movement, it if were made without the use of Unity and based upon the audio files, would not correspond with the audio. This possible delay should be compensated for in an early state. To discover the delay, QuickTime player is used on the used MacBook Air to record the screen. The Reaper and Unity programs are both placed upon a single monitor (as this is what QuickTime supports) and the recording is begun. This recording has a frame rate of 60 frames per second, meaning the delay can be determined with a 17ms accuracy.
During the recording, Unity will be started and Reaper’s playback will be begun with 16 linked audio sources for the test. This is a test scenario similar to a very simple final setup. For the test, the difference in time will be determined by examining the video frames and comparing the moment where the control changes in Reaper indicate a change and the moment where object position values in Unity start changing accordingly.
One track in Reaper was changed to have perceivable changes within a few seconds of starting the track. These changes went from the values 0 to 127 in one step once in the first test-part, and multiple times in rapid succession in the second part. In fig. 13 the test-parts of the track can be seen. Al other tracks were filled with to simulate an intensive moment where all tracks are sending data and Unity is receiving and moving all audio sources.

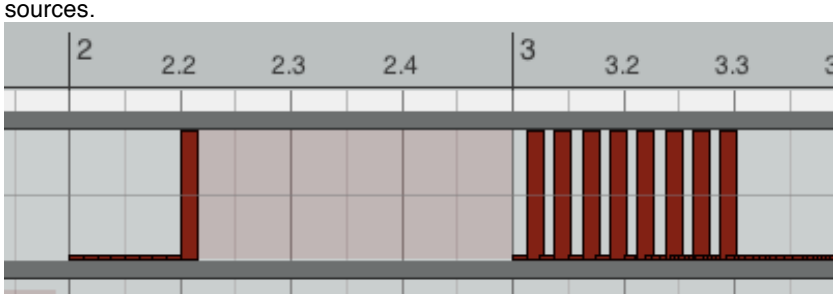


Fig. 13. Control Changes of test track in Reaper

5.3.1. test results

In a video editing program (Adobe Premiere Pro CS6), the delay between the two software seems smaller than the steps of 10ms. It is, however, noticed that the changes in rapid succession put some strain on Unity or the screen recording, as not all peaks in the second test-part were displayed movements. In further tests, with more audio sources, it is important to test this again as it may impose more strain on Unity and cause a delay.

5.4. Volume control

To accommodate for the control of the volume, a small addition is made to the code and new MIDI CC tracks are added in Reaper. The script used by the sources now receives Control Change messages and uses this to set the volume of the audio sources. This is done by adding the following lines of code to the script:
```
        public int CC_Volume = 03;
        track.volume = MidiMaster.GetKnob (channel, CC_Volume, track.volume);
```
To enable the use of a master volume, another three lines in Unity and a single track in Reaper are added.
```
        public int masterVolumeCC = 30;
        public MidiChannel masterChannel = MidiChannel.Ch1;
        AudioListener.volume = MidiMaster.GetKnob (MasterChannel, masterVolumeCC, AudioList
ener.volume);
```
Both look for specific MIDI CC messages on specific channels, both individually defined by the user.

5.5. Physics in movement

To have the audio sources move in a manner similar to a physical setup, the code for the movement was altered to interpret the MIDI CC messages as target coordinates instead of direct positioning values. This means the virtual audio sources will no longer jump from location to location, but instead move gradually towards this point. The following code was added to achieve this behaviour.

```
/* physics movement */
this.acceleration = this.endPoint - this.transform.position;
this.acceleration = this.acceleration.normalized * maxAcceleration;
this.velocity += acceleration;
if (this.velocity.magnitude > maxVelocity) {
	this.velocity = this.velocity.normalized * maxVelocity;
}
this.transform.position += velocity;
```

5.6. Capabilities for large amount of speakers

5.6.1. MIDI capabilities

As it might be possible that Christine is going to use a very large amount of audio sources, it is important to know how many sources can be individually controlled with the current setup. Reaper indicates 120 numbered CC tracks in its editor, which can certainly be used to control the sources. As all 16 channels can be used, there are 1920 possible tracks that can be used. Each audio source uses 4 CC tracks for the x, y and z-position in the virtual space, and the volume of the source. The current setup uses two tracks that can't be used for the sources, namely the track for master volume and the track for the start signal from Reaper to Unity. This results in 479 sources that can currently certainly be individually controlled.
To extend this, new virtual ports can be used, but in this case the code in Unity has to be altered to accommodate the selection of MIDI ports.

5.6.2. Code optimisation

To optimise the user interface, several changes have been made to the setup and to the scripts. The script used for the main volume (changing the audio listener volume settings) is moved to a single game object, instead of having all audio sources change it. The variables for the start signal are also moved to this game object, enabling the user to only change this value once. This was previously to be done at each audio source object individually, with the same outcome.
Other global properties that are now implemented, are the dimensions of the room. It is now only implemented to be a box-shaped volume in which the sources can move, but later it might be implemented to have different shapes, such as a pyramid, triangle or even cylinder.

5.7. Google VR

5.7.1. Google VR SDK

The initial idea to enable the experience on a mobile device using the Google VR SDK[28]. The package enables the creation of a mobile app that has the installation's virtual experience. Due to the nature of this kit, it is not possible to port the current setup to a mobile app, as it now requires the combination of Unity and Reaper. To get this project on a mobile device with VR capabilities enabled, one of the options is to record the paths of the audio sources in Unity, which can then be built into the mobile app. It is not possible to record a 3D or 360 (degrees) video directly, as visually saving the frames takes time, which delays the playback in Unity. During this recording, Reaper will continue to playback at a steady rate, causing the two software to lose synchronisation. I think this problem does not arise with Unity recording object positions, especially when a suitable resolution of these path nodes is chosen.

[28] Google VR Software Development Kit: https://developers.google.com/vr/unity/download

5.7.2. Path recording in Unity

To have the paths in the mobile app, it needs to be possible to have the source of the movement in Unity. On a mobile phone, it isn't possible to have the app interact with Reaper the same way it does on a laptop. To enable the movement with the same paths as set in Reaper, the setup is changed to have the option of recording the movement. The initial plan is to record the positions of audio sources in lists with a resolution set by the user. A set amount of times each second, Unity will save the position target positions of each audio source. This can then, after recording it, be used with the same result as the data sent by Reaper has.
The code is changed to accommodate for this function and the implementation works as follows. As the user checks a box for recording in the settings before starting, Unity will record the target positions as soon as the playback starts in Reaper. It saves for a set period with a chosen frequency. During this time, the script will save the position and the volume of the audio source to a text file. Afterwards it stops the playback in Unity.
Whenever the user unchecks the box for the recording, Unity will try to use the text files. If it fails to find them, it will wait for input from Reaper. When data is recorded, Unity will use this to control the audio sources. Depending on the frequency of the saved data, the movement can be as precise as it was defined in Reaper.
In a test with a LG Nexus 5 android phone, the method has the desired effect. The phone can be rotated around the user to look around in the virtual environment, experiencing the moving sound sources in both audio and visual aspects.

5.8. Flocking

To improve the behaviour of the audio sources in the terms of the original concept, a flocking algorithm is added. With this adaptation, Christine only needs to control a single audio source to have multiple speakers move like a swarm of birds or insects. To do this, the movement of each audio source to its target location is not changed. However, the determination of the target location is changed. For each audio source, the user can choose another audio source to follow. If no leading source is chosen, the movement will be determined by the data send from Reaper or recorded. If the speaker is following another one, it will keep a specified distance from the target and from other sources following it as well.
To keep the program fast and without much delay, the code is changed in a simple manner. Christine can set a speaker that needs to be followed, for each audio source. To prevent audio sources from colliding, some code has been added so that the speakers keep a set distance from each other.

5.9. Virtual environments

To have a better sense of the effect of *Sound Swarm* in one of the possible rooms or spaces, one of them is replicated in a simplistic manner in Unity. An impression of this can be found in Fig 14. As it might later be required, specific locations can be precisely replicated in Unity for an exact virtual experience. However, since several aspects, among which the location, are not certain yet, it is only done in low detail.

5.10. Audio plugin

To provide the most realistic experience, using the capabilities of the Google VR SDK, sound effects such as reverb, the reflectivity of the environment and material elements have been added. Using this provided tool, the virtual environment now takes into account walls, ceiling and floor upon which the sound can reflect as well as the material of the surfaces. This tool also simulates the reverb of the audio as it would have in a real environment.

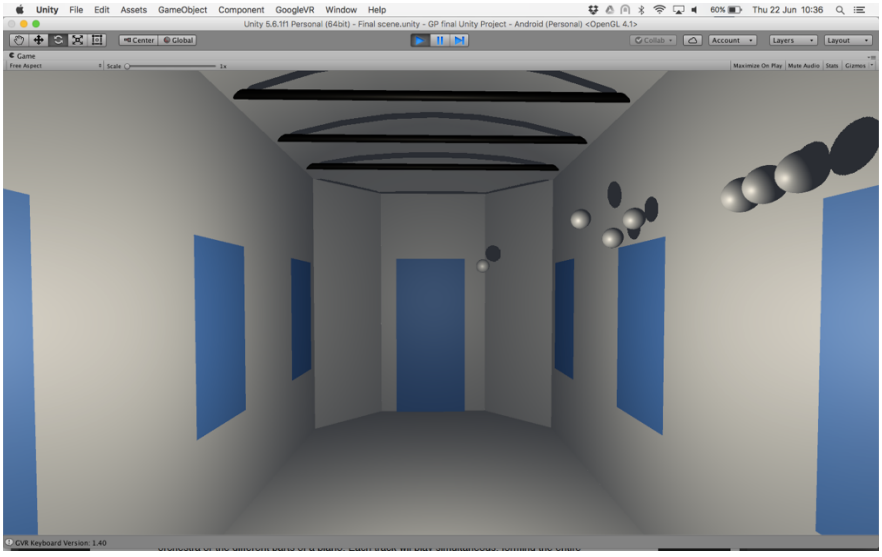


Fig. 14. Virtual environment in Unity

6. Conclusion

This chapter evaluates the components and design choices of the composition tool, as well as the answer on the research question *"How to design a tool to compose spatial music using virtual moving speakers?"*. It also discusses future features of the setup and the future use of the tool.

6.1. Physical setup considerations

To make the tool prepared for the testing of a physical setup, several design choices have been made. Since the physical construction is yet to be determined, the considerations are only in the most general manner. The methods used are included in the physics in the movement, where the audio sources move as they would in a physical setup and where the audio sources cannot pass through each other.
Because of the implementation of these measures, the differences between the virtual test environment and the physical art installation have been reduced. This makes the step from testing to realisation smaller and the progress towards the art installation's concept greater.

6.2. Design choices

During the realisation of the composition tool, the brainstorms with the supervisor and Christine led to several design choices, as well personal insight. These choices mainly focused on resulting in an intuitive composition tool. The methods used are consistency of values and names, minimal amount steps, and relevant information near interactive parts and user input. During the evaluation session with Christine, these measures proved useful and appreciated. The steps to be taken were for each function of a low amount and only a brief manual was needed for the regular use of the tool.

6.3. Composition tool

During the evaluation with Christine, the tool proved to be designed correctly for the purpose. All features included are with a purpose and function, and only minor features are missing. The missing functionality is mostly additions on the current components, as described in 4.4. The prospected goal of testing in a virtual environment is achieved with the software. The software is capable of moving multiple audio sources in a virtual environment with realistic motion, based upon the routing Christine can set up. Christine also has the option to view the virtual installation on a mobile phone, creating an even more immersive experience, great for testing with users.

6.4. Future features

The tool still has several features missing, as they can only be implemented when more details are revealed. One of these is volumetric shape in which the virtual speaker move around. Currently it is limited to a box shape. The dimensions are of the user's choice, however it is desirable for later tests to have different shapes available. These shapes can include a triangular prism, a pyramid, a cylinder, among other shapes.
Another feature to greatly enhance the tests, but the visual aspect rather than the mechanics of movement. To increase the realness of the virtual environment, the detail of the room can be increased. For this, it is needed to have details of specified real locations where the installation is going to be situated. Christine can also acquire the help of an advanced 3D model designer for this, to create a high quality virtual environment.
Just like the environment, the virtual audio sources can be modelled for a greater immersion. Currently they are only represented as spheres, but when the concept(s) of the physical setup is(/are) determined, virtual models can enhance the experience of the virtual *Sound Swarm*. This would improve the results of tests with prospective audience members.

6.5. Future use

The use of the composition tool has one main aspect: trying out the concept movement of the speakers and the concept audio in the virtual environment. The tool can be used to try the movement in combination with audio tracks and rate and perfect the experience with it. It is also possible to use the results of Wouter Westerdijk's research in [1] for more immersive and better rated setups.

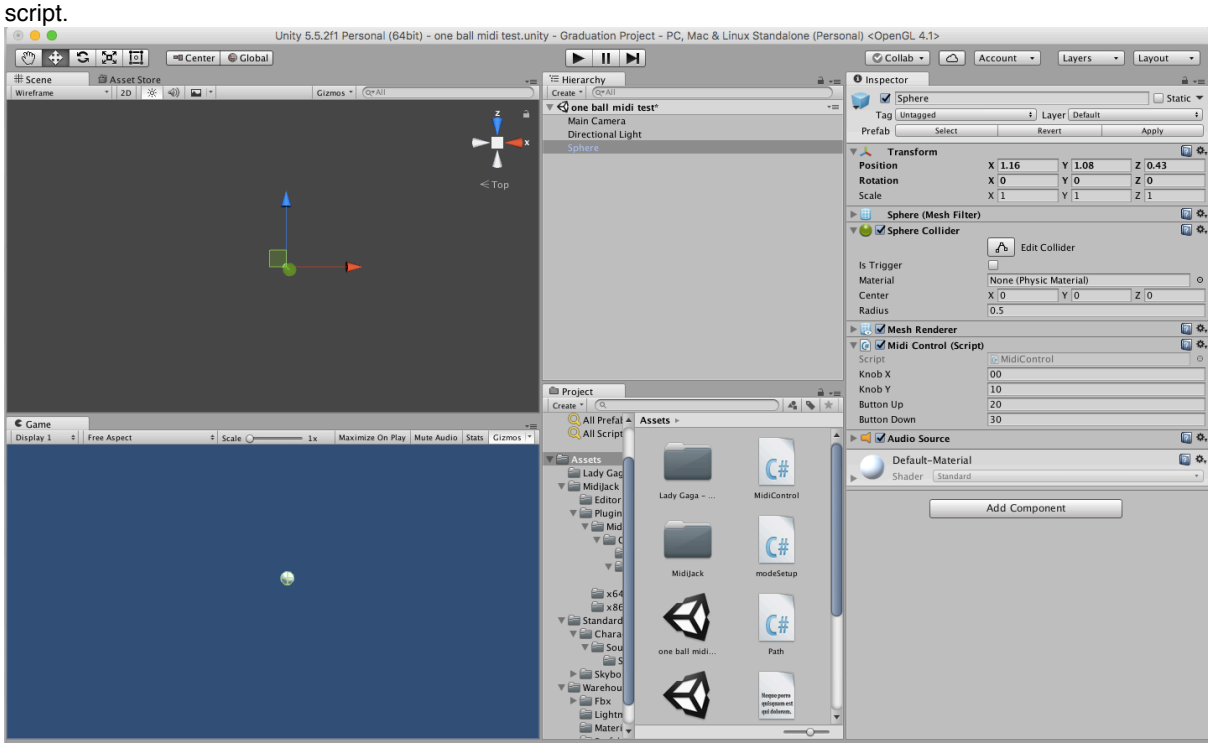
Appendix A

Code for controlling the Unity object with the use of the MIDI controller panel. Game engine scripting language: Unity3D-JavaScript/C#.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MidiControl : MonoBehaviour {

	public string knobX = "00";
	public string knobY = "10";
	public string buttonUp = "20";
	public string buttonDown = "30";

	// Update is called once per frame
	void Update () {
		float newZ = this.transform.position.z;
		if (MidiJack.MidiMaster.GetKnob(MidiJack.MidiChannel.All,
HexToInt(buttonUp)) > 0) {
			newZ += 0.5f;
		}
		if (MidiJack.MidiMaster.GetKnob(MidiJack.MidiChannel.All,
HexToInt(buttonDown)) > 0) {
			newZ -= 0.5f;
		}
		this.transform.position = new Vector3(
			Remap (MidiJack.MidiMaster.GetKnob (MidiJack.MidiChannel.All,
HexToInt(knobX), 0), 0, 1, -10, 10),
			Remap (MidiJack.MidiMaster.GetKnob (MidiJack.MidiChannel.All,
HexToInt(knobY), 0), 0, 1, -10, 10),
			newZ);
	}

	private float Remap ( float value, float from1, float to1, float from2, float to2)
{
		return (value - from1) / (to1 - from1) * (to2 - from2) + from2;
	}

	private int HexToInt ( string hexValue) {
		return int.Parse (hexValue, System.Globalization.NumberStyles.HexNumber);
	}
}
```

Appendix B

Screenshot of the single ball in unity which is controlled with a MIDI controller panel using the “MIDI Control” script.



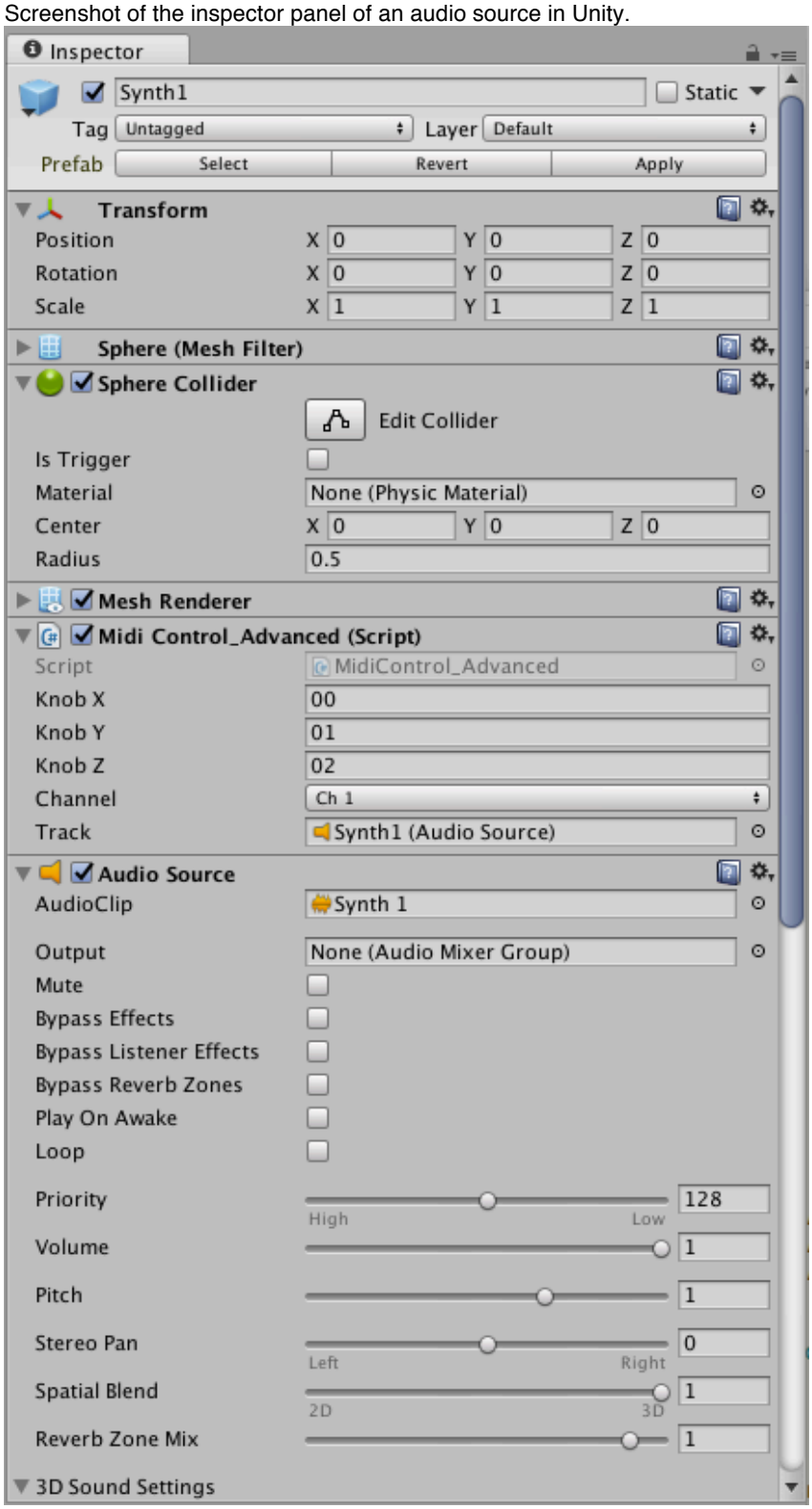
Appendix C

Code for controlling the Unity object with the use of the MIDI controller panel. Game engine scripting language: Unity3D-JavaScript/C#.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using MidiJack;

public class MidiControl_Advanced : MonoBehaviour {

	public string knobX = "00";
	public string knobY = "01";
	public string knobZ = "02";
	public MidiChannel channel = MidiChannel.Ch1;
	private bool playing = false;
	public AudioSource track;
	private string startCC = "29";

	void Start () {
		track.playOnAwake = false;
	}

	// Update is called once per frame
	void Update () {
		if (MidiMaster.GetKnob(MidiChannel.All, HexToInt(startCC)) > 0) {
			track.Stop ();
			playing = true;
			track.Play ();
		}
		this.transform.position = new Vector3(
			Remap (MidiMaster.GetKnob (channel, HexToInt(knobX), 0), 0, 1, -10,
10),
			Remap (MidiMaster.GetKnob (channel, HexToInt(knobY), 0), 0, 1, -10,
10),
			Remap (MidiMaster.GetKnob (channel, HexToInt(knobZ), 0), 0, 1, -10,
10));
	}

	private float Remap ( float value, float from1, float to1, float from2, float to2)
{
		return (value - from1) / (to1 - from1) * (to2 - from2) + from2;
	}

	private int HexToInt ( string hexValue) {
		return int.Parse (hexValue, System.Globalization.NumberStyles.HexNumber);
	}

}
```

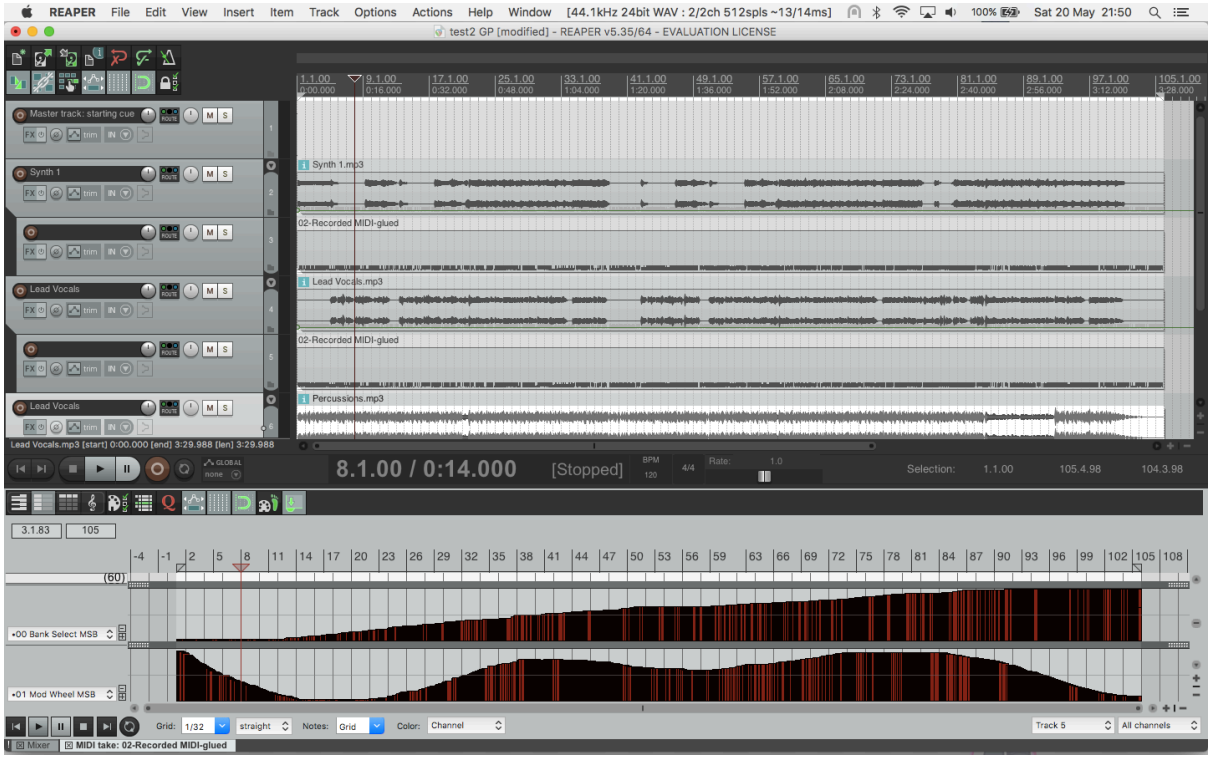
Appendix D

Screenshot of the inspector panel of an audio source in Unity.



Appendix E

Screenshot of the interface of Reaper with multiple tracks, a track with a starting cue and control changes in detail in the bottom half.



Appendix F

Setup Snapshot:

Software used:

Unity 5.6.1f1
Cockos Reaper 5.40 64bit
Android Studio 2.3.3
Apple Xcode 8.3.3

Plugins:

Keijiro's MidiJack
Google VR SDK

Hardware used:

Apple laptop: MacBook Air 13-inch (early 2015) with a 2,2GHz Intel Core i7 processor and 8GB 1600MHz DDR3 RAM memory
Android Phone: LG Nexus 5 16GB, Android 6.0.1
Stereo headphones: Razer Electra
USB MIDI Controller: KORG NanoKONTROL2

References

[1] W. S. K. Westerdijk, “Sound Swarm,” 2017.

[2] R. Kruining, “Using Virtual Reality to design for a 3D audio experience,” 2017.

[3] V. Willert, J. Eggert, J. Adamy, R. Stahl, and E. Körner, “A probabilistic model for binaural sound localization,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 36, no. 5, pp. 982–994, 2006.

[4] M. O. and I. K. and S. A. and H. Matsui, “Sound image rendering system for headphones,” *IEEE Trans. Consum. Electron.*, vol. 43, no. 3, pp. 689–693, 1997.

[5] C. Phillip Brown and R. O. Duda, “A structural model for binaural sound synthesis,” *IEEE Trans. Speech Audio Process.*, vol. 6, no. 5, pp. 476–488, 1998.

[6] K.-V. Nguyen, C. Suied, I. Viaud-Delmon, and O. Warusfel, “Spatial audition in a static virtual environment: the role of auditory-visual interaction,” *J. Virtual Real. Broascasting*, vol. 6, no. 5, 2009.

[7] C. Schissler, A. Nicholls, and R. Mehra, “Efficient HRTF-based Spatial Audio for Area and Volumetric Sources,” *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 4, pp. 1356–1366, 2016.

[8] C. Kyriakakis, “Fundamental and Technological Limitations of Immersive Audio Systems,” *Proc. IEEE*, vol. 86, no. 5, pp. 941–951, 1998.

[9] A. Devana, “3D Audio: Weighing The Options,” 2015. [Online]. Available: http://designingsound.org/2015/04/3d-audio-weighing-the-options/.

[10] A. Dolphin, “MagNular.” [Online]. Available: http://www.dysdar.org.uk/game_engine_projects/magnular.html.

[11] A. Dolphin, “MagNular : Symbolic Control of an External Sound Engine Using an Animated Interface,” *Proc. Int. Conf. New Interfaces Music. Expr.*, pp. 159–160, 2009.

[12] A. Dolphin, “Cyclical Flow.”

[13] O. P. Di Liscia and E. Calcagno, “3DEV: A tool for the control of multiple directional sound source trajectories in a 3D space,” 2010.

[14] C. Miyama, G. Dipper, and L. Brümmer, “Zirkonium MK III - A Toolkit For Spatial Composition,” *J. Japanese Soc. Sonic Arts*, vol. 1, no. 1, pp. 17–21, 2013.

[15] C. (Makezine) Kraft, “Getting Started With VR: The Best Software Tools Are Free,” 2016. [Online]. Available: http://makezine.com/2016/03/24/makers-introduction-vr-best-software-tools-free/.

[16] “MIDI - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/MIDI.

[17] “White Paper: Comparison of MIDI and OSC,” *The MIDI Association*. [Online]. Available: https://www.midi.org/articles/white-paper-comparison-of-midi-and-osc.

[18] “What is the difference between OSC and MIDI?,” *opensoundcontrol.org*. [Online]. Available: opensoundcontrol.org/what-difference-between-osc-and-midi.