Master Thesis in Sound Design
-
Dialog
A Sonic Encounter with a Large Language Model in VR

Jonas Füllemann
Schreinerstrasse 7
9000 St. Gallen
kontakt@jrfsound.ch
+41 77 478 37 31

Github: JFuellem

Mentor: Daniel Hug

Zurich University of the Arts
September 2025

Abstract

This master's thesis documents the design, implementation, and evaluation of *Dialog*, a virtual reality (VR) experience that presents a tangible sonic encounter with a Large Language Model (LLM). The project is driven by two motivations: a technical desire to streamline the creation of custom audio plug-ins for game engines, and an artistic goal to translate an LLM's internal states, as well as the user interacting with it, into a sonic playground that encourages experimentation, exploration, and perhaps even a more *conscious* interaction with artificial intelligence.

The resulting sonic composition is built on a dialogue between two distinct but interwoven palettes. The LLM's internal token embeddings are translated into synthetic, bell-like sounds and a four-part harmony. In parallel, the user's physical movements and behaviours are transformed into organic sound textures stemming from cello and koto recordings. This interplay is extended by additional audio processes, such as formant filtering or waveguide synthesis, that respond to the user's behaviour or specific gestures, turning the experience into an interactive and adaptive composition.

As a technical outcome, this work introduces a pipeline for prototyping audio effects using Max MSP RNBO and FMOD, with the created tools made publicly available. The thesis presents findings from a qualitative user test that informed the design iteration and provides insight into the creation of such an interactive system. *Dialog* demonstrates a method to examine the inner workings of an LLM, laying a foundation for future, more fine-grained explorations in the interactive sonification of artificial intelligence.

Acknowledgements

I would like to thank my Mentor Dr. Phil. Daniel Hug for guiding and supporting me throughout the studies. I am also grateful to Prof. Felix Baumann, who made many things possible with great engagement and generosity. Many thanks also go to Olav Lervik, Rama Gottfried, and Philippe Kocher, who sparked my interest in Max and taught me a lot about sound synthesis. Additionally, thanks to Thomas Resch for his lessons in DSP processing, the results of which I can only describe as sound alchemy. Without all of them, I could only dream of doing this.

I am also grateful to all the professors who have influenced this journey and shape the Sound Design programme, even if they are not mentioned by name.

Thanks to Flurin Fischer from the Artists in Labs residency programme and the IVIA Lab under Mennatallah El-Assady and Rita Sevastjanova for their welcoming, interested, and supportive insights, and for giving me the opportunity to spend time in the field of AI research. Many thanks also to the participants of the user tests for their openness and valuable feedback.

Finally, I would like to warmly thank my parents for their support and trust that they have given me throughout.

Contents

1. Introduction

This thesis documents my journey, from technical innovation in plug-in creation for games to exploring this artistically in a scientific context of AI research. I describe the design, its implementation and evaluation of *Dialog*, an experience in virtual reality, where users can interact with a Large Language Model through word, sound, and movement.

The internal references are colour-coded with blue and the external links with brown. The most important links to audio demos, movies, or websites are clickable QR codes shown in the outer margins that allow one to find the resources both in print and digitally.

This introductory chapter starts with the personal and technical Motivation and Goals (1.1) and provides an Outline (1.2) of the following chapters.

1.1 Motivation and Goals

For my Bachelor's project "Journey to Audio", an audio-only Virtual-Reality (VR)-experience, I have made first attempts at extending my audio toolkit by writing new plug-ins. With the help of Cabbage Audio[1] and its csound-FMOD[2] wrapper, a new world opened up in possibilities. For example, with the built resonant filter, I was able to tell a sonic story by slowly transforming sound design into music by gradually filtering the wind into a melody.

After taking lessons in digital signal processing (DSP), I started writing simple effects from the ground up for ongoing game projects. In the game *[I] doesn't exist* by LUAL Games[3], for example, with a custom-written bitcrusher, I was able to dynamically influence the sound quality depending on where the story went. However, writing more elaborate plug-ins is a time-consuming and cumbersome task, which until then often took me out of a creative flow. Therefore, in early 2024, I started exploring the authoring of plug-ins in Max MSP RNBO[4] and wrapping its exportable source code for the FMOD middleware[5]. This not only speeds up the

[1]Cabbage Audio is an open source application by Rory Walsh based on Csound and JUCE that allows to make effects and instruments. For more information, see https://cabbageaudio.com/

[2]FMOD is a program developed by Firelight Technologies Pty Ltd that facilitates the implementation of sound into game engines. See: https://www.fmod.com/

[3]For more information about the game see https://store.steampowered.com/app/1943420/I_doesnt_exist__a_modern_text_adventure/

[4]RNBO is an additional part of the node-based programming environment Max MSP by Cycling 74, that allows to export the authored instruments or effects as VST/AU plug-ins, Java-Script and C++ code. See https://rnbo.cycling74.com/

[5]I have also developed a wrapper for hvcc (https://github.com/Wasted-Audio/hvcc), which generates exportable source code from a Pure Data patch; the entire pipeline is open source. However, the current lack of single sample manipulation, as available through gen~ and FFT processing, led me to the use of RNBO. However,

development process of new effects and instruments, but also enables me to test and preview those directly in the Max environment.

Equipped with these new capabilities, I searched for a project that allowed me to work intensively with this technique and test its robustness.

The project should benefit from rapid prototyping and require some degree of complexity where custom effects and instruments would enable a new kind of interactive experience.

With my Artists in Labs residency programme at the IVIA Lab at ETH Zurich[6] the perfect opportunity arose. For two months I was able to follow a research team that is focussed on artificial intelligence visualisations and intelligence augmentation, and develop my own project during the stay. Some researchers worked extensively with text and Large Language Models (LLMs), so the idea soon was to sonify an LLM, which is a complex, high-dimensional, and dynamic system.

(1) BERTophon App and Video

The result during the residency was (modern)BERTophon [1] , a web application that allows the user to insert multiple sentences and explore them sonically based on the embeddings of the BERT model, which is an encoder-only model[7]. As the concept of this type of model is quite difficult to understand, I developed it further using a chatbot, a generative decoder-only model that enables a conversation with the user. This exchange of words should be presented similarly to the first project, but in a more tangible, interactive, and engaging way.

As AI and LLMs are being used increasingly in more and more aspects of everyday life, I see potential dangers, but also opportunities, in their adoption. For that reason, I wanted to shift the focus from the model alone to the user and their interaction with it, encouraging a more *conscious* interaction with the technology. This was done by extending my sonification approach by analysing the user's behaviour and making their measured movements audible. The interactions with the model also produce sound, which turns them into instruments. This allows an exchange with the model on not only a textual but also a musical level.

Using virtual reality (VR) as a work medium was an easy choice, because of previous experiences in my Bachelor's project, projects that followed, and the simplicity of retrieving the user's motion data. It also allows for visual immersion in a fictional interaction space and for direct interactions with the model through the user's own voice, hands, and gestures.

hvcc is actively developed by Wasted-Audio and has an active discord community

[6]The Artists in Labs Residency is organised by the ZHdK culture analysis department and offers dedicated residencies for master students in their master series. For more information on the programme, see https://artistsinlabs.ch/ and the IVIA Lab https://ivia.ch/

[7]More details later in Large Language Models (LLMs) (2.1)

Therefore, my goals for this thesis were twofold:

- To test, improve and demonstrate the sound synthesis plug-in pipeline for game engines in a concrete project.
- Creating this project in developing an application in which users can encounter an LLM in VR. The resulting embodied interaction of the sonic as well as textual elements should invite experimentation, give insights into the data behind, and perhaps even lead to a more *conscious* interaction with a Large Language Model.

Project Outcomes

The created plug-ins will not only serve me in future projects, but have been made public including the project files on my online crowd plug-in platform [2] [8]. This provides an example of how plug-ins can be shared and collaborated on there. The wrappers have been improved and can be used by anyone through the online platform or directly on GitHub for RNBO [3] and hvcc [4] .

(2) Online Compiler

(3) RNBO Wrapper

(4) hvcc Wrapper

The design and implementations of *Dialog* describe a novel way of interacting with and exploring the internal states of an LLM by using sound, visuals and gestures in a virtual environment. The results of testing the design and iterating over it provide insight into the requirements of creating such a system. Finally, the described challenges and opportunities of the sonification approach and the techniques used provide a basis for future work and exploration.

1.2 Outline

First, a theoretical framework for machine learning, sonification and related work is found in Background and Related Work (2). In Prototype Design and Development (3) I describe the development history with details of the overall architecture, synthesis, and other techniques used. The following chapter User Testing and Design Iteration (4) contains information about the state of the experience at the time of the user testing, the user testing itself, its findings, and the resulting design iteration and final result. Discussion and Reflection (5) reflects on the goals, the sonifications, and the process and shows limitations and future work. The thesis ends with the Conclusion (6). The appendix at Additional Technical Descriptions (A) contains additional technical background for further reading and documentary purposes.

[8]This website makes it possible to compile source code from RNBO and hvcc to FMOD plug-ins. The built plug-ins are stored and managed through a database so they can be shared with a community

2. Background and Related Work

In this chapter, I give an introduction to the two main topics, *Large Language Models* and *Sonification*. First, I give an overview of LLMs in Large Language Models (LLMs) (2.1) and what data they work on; and second, possible ways that these data can be made audible with an insight into sonification in Sonification (2.2).

Finally, based on the basics of the first two sections, in AI as Artistic Material and Related Works (2.3) I want to place my work within existing projects.

2.1 Large Language Models (LLMs)

In this section, I will briefly explain the transformer architecture, which is the technology behind today's most prominent Large Language Models. This is far from being an exhaustive summary on this topic, but I want to cover the basics that are important to understand the sonification decisions that I made during development and to establish some taxonomies, which I will use throughout the paper. My explanation will mainly focus on inference, which involves using a pre-trained neural network to get outputs from different inputs.

In order for the model to work with text, the words first need to be tokenised. Tokens are derived from the model's vocabulary and are whole or segmented words. The aim is to store the largest representable text variety in the least number of different building blocks. For example, a common word like *like* is probably its own token, while *likeable* would be composed of *like* and *able*, as *able* could also stand for itself or be used in other adjectivisations. Each token type, meaning each token inside the vocabulary has a learnt embedding, which is a high-dimensional vector (a representation of a token by many numbers).

Those embeddings are then processed through stacks of identical transformer blocks (also called layers) and are refined by taking the whole sequence into account. The transformed embeddings that are output by a layer are fed into the input of the next layer. Those outputs are the numeric values that I am using for the sonification.

An important principle of transformer models is called multi-head self-attention, where each token *attends* to the other tokens in a sentence to assess its contextual meaning. *Multi-head* means that this happens in parallel, where different *heads* act as different specialists. This might mean that a head might resolve pronouns, others that primarily attend to punctuation, or simply one that focusses on adjacency. For example, when having a sentence like: "The reader is learning about sonification, because *it* is interesting." The token 'it' would probably attend the most to the tokens of *sonification* in one head, on the other to the comma right after, and in a third directly to the previous *because*.

While being essential for the function of transformers, this mechanism did not find its way into my sonification approach due to its complexity, and only a rough visualisation takes place by drawing lines from the three most attended tokens to the new one, averaged over all the existing heads.

2.1.1 Dimensionality Reduction / Feature Projection

To be able to visualise high-dimensional vectors, dimensionality reduction / feature projection is used. It is an algorithm (most common are UMAP, PCA and t-SNE), that is able to reduce a set of high dimensionality vectors to a set of lower dimensionality, while trying to maintain their structural relationship. A projection to 3 dimensions allows the embeddings to be placed in space. It can often be seen in feature maps, where similar tokens are closer together. An example in the audio domain is *The Infinite Drum Machine* [5] by Manny Tan and Kyle McDonald, which organises a set of sounds based on their similarity in a two-dimensional plane using t-SNE.

(5) The Infinite Drum Machine

2.1.2 Two Different Architectures: NLU and NLG

Transformers are typically designed for two different use cases in Natural Language Processing (NLP). Natural language understanding (NLU) or natural language generation (NLG).

Natural Language Understanding (NLU)

NLU models like BERT (Bidirectional Encoder Representations from Transformers), used in my first project, turn text into a rich numeric representation that can be used by computers to better understand the meaning of a text or semantics of words within. It is *bidirectional* meaning a token attends to the tokens before and after itself and is processing a sentence as a whole. Its training is based on a "masked language modelling" task in which the model has to predict a hidden word within a sentence. Through its architecture, it also produces a [CLS] token, which is placed at the start of the sentence and that aggregates the information of the whole sentence (Devlin et al., 2019). It is also called an encoder-only model, as it only uses the encoder part of the model during inference to encode text to this numeric representation.

Natural Language Generation (NLG)

NLG models like Qwen, used in this project, are designed to generate novel text content. This is done one by one, and the model is therefore trained on the goal of predicting the next token. They are *autoregressive* meaning that the newly generated token is a result based on all previous tokens. This means that they are only attending to earlier tokens by nature. At the end of the stack of layers and after further transformation, omitted here, the embeddings are turned into logits, which are numbers representing the likelihood of the next token in the vocabulary. They are therefore also called decoder-only models, decoding their internal state into tokens, which become human-understandable words and phrases, when connected together. An excellent interactive visualisation of this process can be found on the Transformer Explainer website [6] (Cho et al., 2024).

(6) Transformer Explainer

A modern use of this are models that do what is called chain-of-thought (CoT) *reasoning.* They are trained on data sets with questions, answers and the underlying written reasoning behind them. When such a model responds to questions during inference, it first produces a set of *thinking* tokens, where a problem is broken down into smaller steps, before giving a final answer. The generation of these additional tokens comes with an increasing cost of compute resources and also takes longer, but generally gives a more precise response (Raschka, 2025). Therefore, I implemented the possibility for the user to selectively enable or disable the reasoning process. When enabled, intermediary tokens that are typically not seen by the end user are also made audible and displayed.

The mentioned architectures above lay the foundation how today's LLMs understand and generate text. Development, however, advances from going from text generation to enabling the models to act on a broader range. Next to Mixture of Experts (MoE) models, where subnetworks within a single model process different information, the newly released GPT-5, now called an AI system, instead of an AI model, is routing prompts dynamically between different models (OpenAI, 2025)[1]. Linked to that, I also wanted to touch on the concept of *tool-use*, that has become more accessible in the last months. The keyword is the Model Context Protocol (MCP), which is an open standard protocol through which language models are given the ability to fetch information by, for instance, browsing the web or to control any application, and therefore could also be used to build an agentic system.

Although these advances exceed the scope of my project, they show an interesting future direction for creating interactive AI systems.

[1]... which is problematic in my eyes as it reduces the agency of the user to choose the right tool in favour of an obfuscated, automated decision process

2.2 Sonification

Sonification describes the process of turning data into non-speech sound. Although it is generally less known than its sibling, the visualisation, it is an ubiquitous part in everyday life. Hermann identifies the following requirements for a sound to be called a sonification: data determinism, objectivity, systematicness, and reproducibility (Hermann, 2008, p. 2). The last one means that the same input should yield the same output.

Our hearing excels at pattern recognition, even under noisy conditions (Hermann et al., 2011, p. 2), which to this day has not been surpassed by computers. For example, take a skilled orchestrator that is able to discern and transcribe a score played by 90 musicians, decoding set of sound pressure waves. In the second chapter of the sonification handbook, Walker and Nees (2011, p. 12) present four broad categories of functions of sonification:

1. alarms, alerts, and warnings
2. status, process, and monitoring messages
3. data exploration
4. art, entertainment, sports, and exercise

This project mostly makes use of the last two, using sonification as a means to explore the underlying data of an LLM and as an artistic and interactive experience. However, also the second one is sparsely used to convey states of the application.

To understand my specific approaches in DSP Plugin Implementations (3.4), I will briefly summarise three key types of sonification: Audification, Parameter Mapping, and Model-Based Sonification.

Audification

The most direct form of sonification is achieved by treating the data directly as audio material. Data points are treated as discrete amplitudes of an audio file and played back at a speed such that the resulting signal enters the audible hearing range. With this, it is possible to scan large amounts of data with our ears in a short time. Classic examples are the sonifications of seismic events that have a long history dating back to 1961 and where in later works categorisations of such events only by hearing were correct in more than 90% (Dombois, 2002). Other more recent works at NASA focus on the audification of space probes (Alexander, 2024) or solar winds (20kHz Podcast, 2025), both under the hand of Robert Alexander. In the latter case, he was able to hear the Sun's rotation based on a swelling hum that was not evident when looking at the raw data.

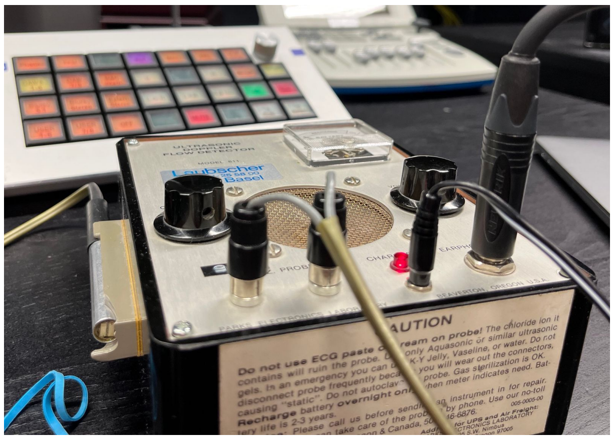


Figure 2.1: An example of audification in medicine. An ultrasonic doppler device for analysing blood flow. Source: personal collection.

It also has a history in the field of medicine, for example in the sonification of EEG waves or the ultrasonic Doppler technique, which is used by angiologists to analyse blood flow in blood vessels (See Figure 2.1 and its recording [7] of my carotid artery.

(7) Doppler Recording

Parameter Mapping Sonification (PMSon)

Parameter mapping sonification (short PMSon) is the most widely used sonification type. In their chapter on this topic, Grond and Berger (2011) state that it involves assigning information to auditory parameters to display data (p. 363). The choice of auditory parameters is vast and can range from the most obvious ones like pitch to "complexes in the time and/or frequency domains such as timbre, chords, melodic, or rhythmic patterns" (p. 365). As data through the immense possibilities of mappings can be scaled, quantised, and otherwise altered to fit the target domain, it is "useful to consider what information is lost" (p.365) when deciding on a strategy.

They also highlight different mapping topologies (Grond & Berger, 2011, pp. 363, 365):

- **one-to-one**: The assignment of a single data domain parameter to an auditory parameter.
- **one-to-many**: The assignment of ranges of a single data domain parameter to different auditory parameters.
- **many-to-one**: The assignment of multiple data domain parameters to one auditory parameter.

They omit the **many-to-many** mapping without elaborating why, presumably because this type of mapping makes it difficult if not impossible to interpret the sonic result.

In my project, I used a combination of all three mentioned topologies. For example, the **one-to-one** mapping is used for the layer selection, where each layer corresponds to a pitch. For the bells, a **one-to-many** mapping is used to turn one complex data source (the embedding vector of a token) into many auditory parameters (multiple frequencies that contribute to the bell's timbre). Finally, a **many-to-one** mapping acts on the granular cello pad, where multiple streams of user movement data (head and hand positions) are averaged to control the single parameter of playback position within the audio sample.

Model-Based Sonification (MBS)

Model-based sonification is less well known and involves creating a physical acoustic system that is built on data. It is typically silent until it is excited and in doing so different features of the data are revealed. Hermann (2011) mentions a mass-spring system, where data points are located in a 2D or 3D virtually simulated physical space consisting of elements of different mass attached to springs of different stiffness. The user would then be able to send a shock wave from a location that would propagate through space and excite the masses. The physical reactions of the masses mapped to defined sonic parameters would then lead to a sonic result.

I could also imagine the use of other physical modelling synthesis approaches, such as Modalys from IRCAM[2], assigning different data to different material properties or even 3D meshes, ultimately being able to listen to the data by exciting the virtual object at different places.

Another example would be the sonification in the particle system, developed by Alexander Mordvintsev called *Particle Lenia*. It is based on the principles of cellular automata, but using particles and energy fields. The sonification gives each particle a voice based on two parameters, its speed and potential energy, mapped to volume and frequency, respectively. The sonification, however, is not the primary focus of that work and is more a fun by-product (Mordvintsev, 2022).

Although my project implements the Karplus Strong algorithm that counts as physical modelling, I have not used this approach. However, I wanted to include it because I will come back to it in the end in Limitations and Future Work (5.4).

[2]Modalys is a framework that also integrates into Max MSP, that is focussed on phyiscal modelling. For more information visit https://forum.ircam.fr/projects/detail/modalys/.

2.3 AI as Artistic Material and Related Works

Many projects use generative neural networks to produce a final artistic output[3]. In contrast, this project focusses on the underlying data of the network itself, translating its internal states into sound.

In this section, I want to give insight into other works that go into this direction. Additionally, since I translate movement into sound, I will include works that use hands and sound and highlight a dance sonification project that stood out during my research.

Artificial Neurons for Music and Sound Design - Simon Hutchinson

Simon Hutchinson is a composer and media artist and holds the Chair of the Department of Music, Theatre, and Dance at the University of New Haven. In his experiments and research on the workings of machine learning, he recreated the work principles of an artificial neuron in pure data, translating it into the signal domain. By combining multiple neurons of this kind, he creates chaotic systems that mix singals, trigger sounds, or control analogue synthesisers (Hutchinson, 2024).

The values he takes to drive the musical parameters are essentially the same values that I use for mine. The difference is scale and mapping precision. Where he might use 20 neurons that are selectively assigned, the two models currently used in *Dialog* have around 3 and 25 million respectively, where most of the values are averaged, reduced through feature projection, or omitted.

Singling: The Song of Our Words

Singling is an open source application developed at the University of British Columbia and was presented at the 26th International Conference on Auditory Display (ICAD 2021). Its user interface allows for the assigning of MIDI events to lexicogrammatical text features to create a musical score for a given text. They focus on word families such as verbs or nouns, other metadata taken from WordNet, a large English word database. In later revisions they also use natural language processing algorithms to perform Parts of Speech (POS) tagging, which allows for determining the linguistical role of each token in a sentence or finding corresponding words. They propose the term *literacoustics* "[...] the act of reading a text by listening to its sonification" (Morales et al., 2021).

[3]Probably the best known works are coming from Refik Anadol https://refikanadol.com/

Their work comes closer to *(modern)BERTophon* than *Dialog*, as a tool for exploring text input, but with a focus on linguistic properties and flexible mapping. While my mapping is fixed and is purely based on the raw embeddings, their tool allows the user to freely assign one-to-one and one-to-many mappings to concrete properties, for instance, to increase the volume and right panning each time a noun appears or to change the instrument to a saxophone each time a verb connected to feeling occurs. They have also planned to do an online version, showing an architecture similar to mine with front- and back-end [4].

Mark II - Philipp Schmitt

Treating a completely different architecture, the Convolutional Neural Network (CNN), famously used for identifying handwriting, Philipp Schmitt made a kinetic sculpture depicting the inner workings of such a model during his residency at the NYU Center for Data Science. It is based on a researcher's drawing of the schematic for the model (Schmitt, 2020).

The sculpture is a fully working model, but focusses more on its architecture than a novel visual or sonic representation or transformation of intermediate numeric representation, which are depicted using different intensities of light.

The Hands and Hands

The Hands from Michel Waisvisz from 1984-2000 is one of the most famous example when looking at the history of digital music instruments, especially for controlling sounds with the hands. He built three versions of a hand interface with a multitude of sensors. The third version included, in addition to thirty push keys on each hand, potentiometers, and many more, an ultrasonic sender and three receivers that could be used to measure the distance from each hand. This was mapped to MIDI notes or expressions and allowed him to control synthesisers and other MIDI devices (Torre et al., 2016).

Forty years later, a composition by Yannis Kyriakides called *hands* was performed[5], also in the Netherlands. It is a composition with a video score for any number or kind of instrument. The video consists of 46 different 3D animated hand gestures. It can be performed by live instruments and people wearing a sensor that is linked to a granular synthesiser that manipulates a recorded buffer. The intended sensor is a MYO sensor that measures the electric potential of a muscle[6], but any sensor or technology can be used. The players react to the gestures shown on the video as they like and are invited to move through the space as they play (Kyriakides,

[4]See https://dlsn.lled.educ.ubc.ca/wordpress/singling/
[5]See https://www.youtube.com/watch?v=5v6HVv8-Vq0&t=410s
[6]See https://myoware.com/products/muscle-sensor/

2024).

My work stands on the shoulders of both, but instead of having physical interactions and/or measurements, it relies on computer vision.

Biodata Sonata

In my research on user sonification, one project stood out through its detailed blog-documentation (Minimi Dance Theatre, 2023): *Biodata sonata.* The project comes from the Finnish dance theatre *Minimi*, in collaboration with the SonicMove Project, the University of East Finland, the VTT Technical Research Centre of Finland, and Aalto University. A team of choreographers and researchers, alongside a sound designer called Josu Mämmi, created a dance piece in which the dancer's movement controls sound and light following a sonata form.

They used multiple inertial measurement units (IMUs) per dancer, which measured and reported the orientation, acceleration, and forces exerted on it. Three computers in tandem calculated movement mappings, controlled Ableton with Max for Live patches, and one TouchDesigner for the visuals. Granular and FM synthesis, dynamic sequencing of samples, and cross-fades of pads with different timbral characteristics were integrated into a seamless interchange.

A phrase stood out for me from the dancer Katja Mustonen: "At best, one's agency in this creation will be based on how instantaneously the sound reacts to movement making the dancer's experience more reflective and sentient." This is exactly what I am looking for in *Dialog* when formulating the goal of a *conscious* interaction through the embodiment of sound. The correspondence of sound with movements altering proprioception, ideally leading to experimentation and self-reflection. However, where they use IMUs, I use the headset's computer vision capabilities, which predict the position of the hands. This will never reach this level of precision and instantaneity and is a trade-off to the simplicity of my setup, which requires *only* one device to measure movement, synthesise sound and create visuals. However, I still think that the mentioned embodiment of sound is possible under these conditions.

3. Prototype Design and Development

This section describes the whole development process. In Core Design Concepts (3.1), the core design decisions that provided the scaffolding are described first. Then, in Preliminary Prototyping in Python and Max (3.3) and DSP Plugin Implementations (3.4) the written plug-ins and their usage are described. Further details of the implementation are given in Further Audio Implementations (3.6).

3.1 Core Design Concepts

To create a composition that takes the user's action into account, I made the decision to sonify the LLM (Machine-Sonification) and the user's movements and behaviour (User-Sonification). In a similar way that LLMs imitate human speech, which gets harder and harder to distinguish, the two sonic approaches should be clearly distinguishable, but also interwoven and blurred.

Machine-Sonification The source material should be of pure synthetic origin as a symbolic pole for artificial (intelligence). The selected data sources are the token embeddings and attentions of all layers. In order to make the experience more accessible and to give the model a more human characteristic, the text output should be given a voice and processed in a way that fits the setting.

User-Sonification The source material should be of pure organic origin as a symbolic pole for human nature. The selected data sources are the movement of the user. Additionally, the behaviour should have an impact on the tonality, such as how long the user spends listening, speaking, or interacting.

Furthermore, my interest lies in the sonification of the available raw data and not in their interpretation, for example, by using another neural model that predicts semantic or emotional content[1].

Finally, its design should invite experimentation and interaction. For this reason I aimed for a balance that the sonification is pleasant to listen to but potentially can be interpreted.

[1]An exception is the user's movement, which for this project is most definitely estimated by a neural network behind the scenes using computer vision.

3.2 Overview of the System and Components

Learnt through the residency, the usage of a back-end server is essential. Although it would be great to calculate the language model, its analysis, and text-to-speech on-device, the implementation is impractical and slow, especially on a mobile device. Although Unity provides a neural network inference API for ONNX models[2], the model would have to be exported in a custom way to fit the project's requirements to output the embeddings and attentions, which was shown to be error-prone and too complex in early experiments. Therefore, I built a Python REST API using FastAPI[3]. Hereafter, back-end refers to the API, and front-end refers to the Unity app.

The chosen architecture is WebGL to explore a promising direction of browser-based applications that are platform-agnostic even for VR and refreshingly open source. It is mentioned here because it has some additional implications for development. More information can be found in the appendix under Platform Independent Developing (A.1).

Figure 3.1 shows the final architecture as a whole. Figure 3.2 zooms in and shows a simplified schematic of the final audio system implemented. They provide a visual reference for the next sections that explain the elements in detail. The numbers correspond to the corresponding chapters. The *Game-Logic* is not elaborated further but includes the communication with the back-end, the management of all data, the creation of the visualisations and other required components that are not shown explicitly.

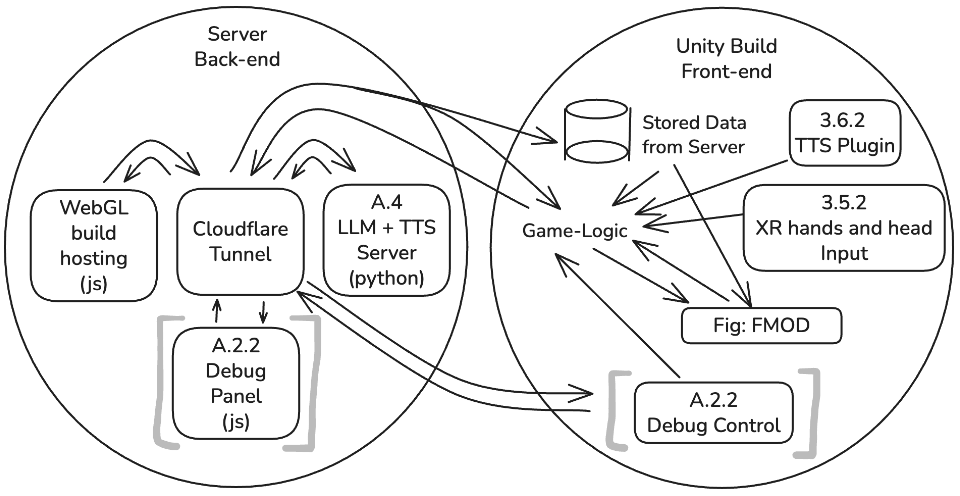


Figure 3.1: Schematic of the architecture

[2]The Unity package newly called InferenceEngine (https://docs.unity3d.com/Packages/com.unity.ai.inference@2.2/manual/) makes it possible to infer neural networks directly inside Unity using the Open Neural Network Exchange (ONNX) standard, see https://onnx.ai/

[3]FastAPI is a modern and fast web framework for building APIs with Python. See https://fastapi.tiangolo.com/

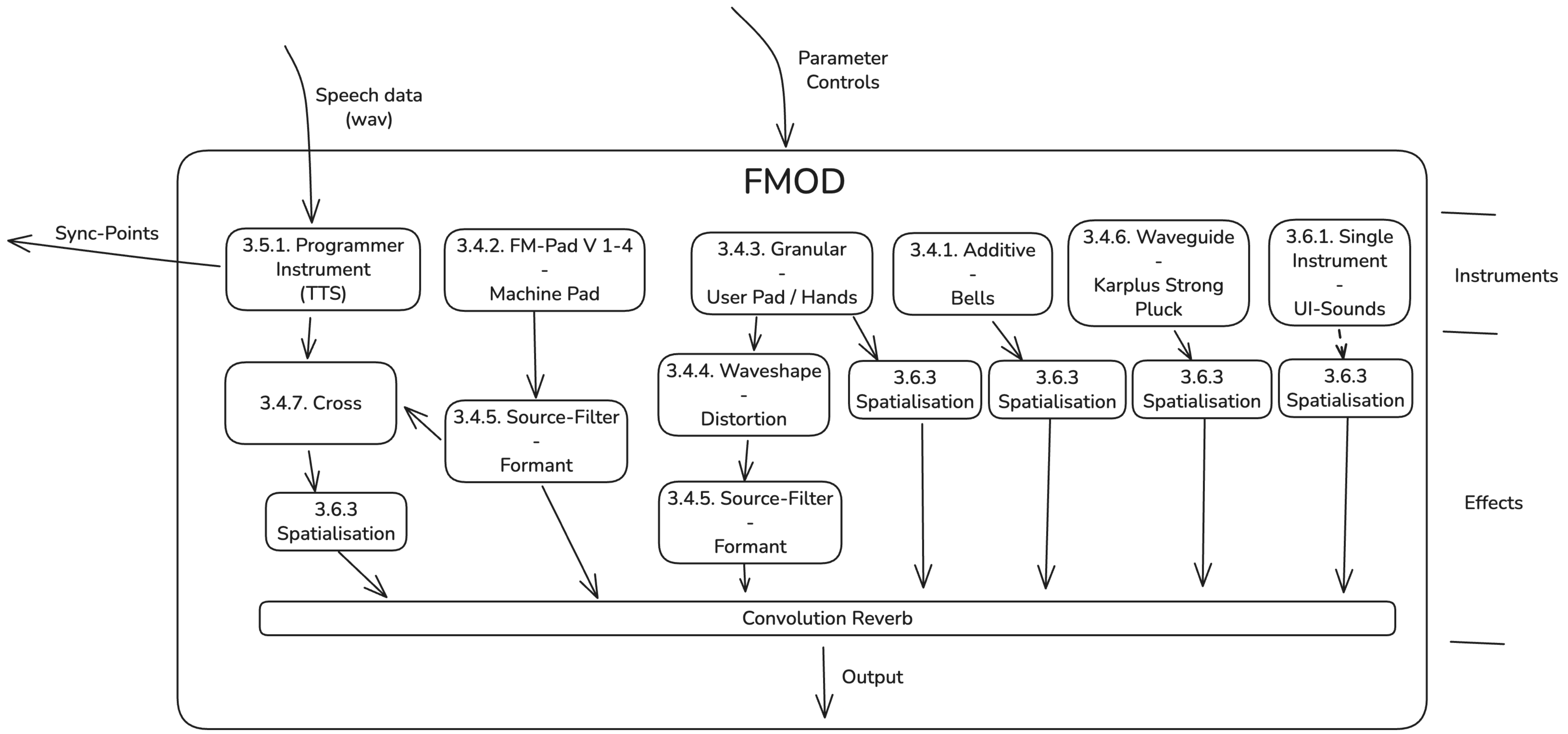

Speech data (wav)
Parameter Controls
FMOD
Sync-Points
3.5.1. Programmer Instrument (TTS)
3.4.2. FM-Pad V 1-4 - Machine Pad
3.4.3. Granular - User Pad / Hands
3.4.1. Additive - Bells
3.4.6. Waveguide - Karplus Strong Pluck
3.6.1. Single Instrument - UI-Sounds
Instruments
3.4.7. Cross
3.4.5. Source-Filter - Formant
3.4.4. Waveshape - Distortion
3.6.3 Spatialisation
3.6.3 Spatialisation
3.6.3 Spatialisation
3.6.3 Spatialisation
3.6.3 Spatialisation
3.4.5. Source-Filter - Formant
Effects
Convolution Reverb
Output

Figure 3.2: Schematic of a simplified FMOD signal flow with index numbers reflecting the section numbers

3.3 Preliminary Prototyping in Python and Max

During my residency, when I first saw a visual representation of token embeddings as seen in Figure 3.3[4], my first instinct was a spectrogram. Through vocoding[5] I tried turning them into sound, using Librosa[6] in Python. However, experiments resulted in unpleasant FFT artefact noise, and my interest was more in turning one token embedding into sound instead of hundreds.

Very similar to this approach is the use of sinusoids to represent any audio signal (IRCAM, 2025). In a simplified adaptation, I had the idea to map the strongest 20 indices of the embeddings to frequencies and play them back with sines to simulate a similar effect. And with that the first version of the BERTophon[e] was born. A demo-video of the Max prototype can be found here [8] .

(8) BERTophone V1

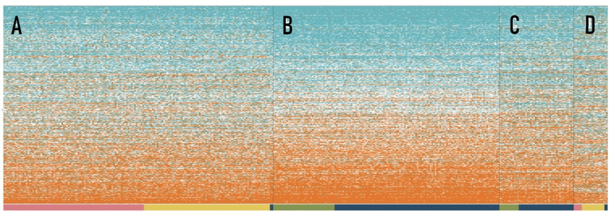


Figure 3.3: Spectrogram-like embeddings representation from hundreds of adjacently placed tokens.

All of the later developed techniques and tools that have led to the final (modern)BERTophon I also use in the final master project and will be discussed later. However, after the residency ended, I tried going back a few steps, rethinking the sonification approach, since the first happened rather intuitively and quickly. Other Max prototypes followed, and I tried other audification techniques (See Audification (2.2)). In the first one, I tried to play the embeddings slowly or quickly by converting each embedding value to a frequency. In the second, most literal audification technique, I played back the (normalised) embeddings as if they were audio files. I am certain that much more can be done in this direction, for instance by following the section about signal conditioning for audification (Dombois & Eckel, 2011, pp. 313–315). However, time constraints and liking the aesthetics of the first approach led me to go back to the existing techniques and build on them.

As all plug-ins are done in Cycling 74 Max RNBO, the following descriptions were all started with a Max patch.

[4]Sevastjanova et al., 2023.
[5]Term used here as in turning images/spectrograms into sound.
[6]See https://librosa.org/doc/latest/index.html

3.4 DSP Plugin Implementations

The following sections describe the plug-ins that I made for this project and how and in what part of the sonification they are used. I have not invented anything new here, as all these synthesis techniques have been used for a long time; however, it has never been easier to directly implement them into a game engine.

3.4.1 Additive Synthesis: Making Embeddings Audible (Transient)

To give each token in each layer a clearly perceptible sonic fingerprint, I needed a sound with a sharp attack and wide spectral range, so that it would cut through a potential sound bed. A bell-like sound, created with additive synthesis, would be a great fit. The rich variations and colourful interferences of the odd overtones of bells make them one of my favourite sounds. This plug-in therefore directly originates from the first BERTophon, the only addition being a characteristic percussive envelope that turns it into a bell sound, also known as *Risset Bell.*

Technical Description

In BERTophon, the indices of the highest values of the token embedding are linearly mapped[7] to a frequency range and played back with sine waves. The percussive envelope has a short attack (60 ms) and a long exponential decay (3000 ms) and is directly driven by the ADSR parameter modulator in FMOD.

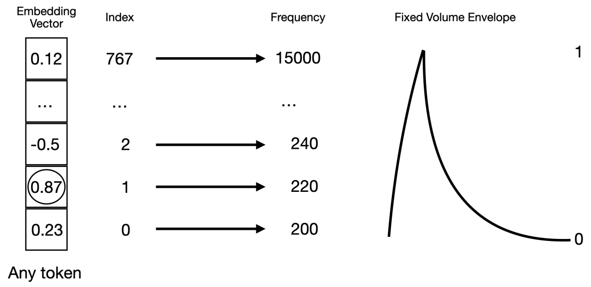


Figure 3.4: Schematic of the Risset Bell mapping.

I also experimented with a similar approach, first calculating the FFT of the token embedding and similarly mapping the indices of the strongest amplitudes to the frequencies. I ultimately chose this FFT-based approach, as the relationships between the tokens seemed to improve. For example, punctuation tokens now had an audibly similar characteristic in some layers, which suggests a better mapping of the underlaying data relationships.

[7]In *Dialog*, they are logarithmically mapped to favour higher frequencies over lower.

To solve this using RNBO, Unity sends the indices in quick succession every frame to FMOD. In RNBO I use a polyphonic patcher where each frequency parameter change sets and locks a new instance of a playing sine wave.

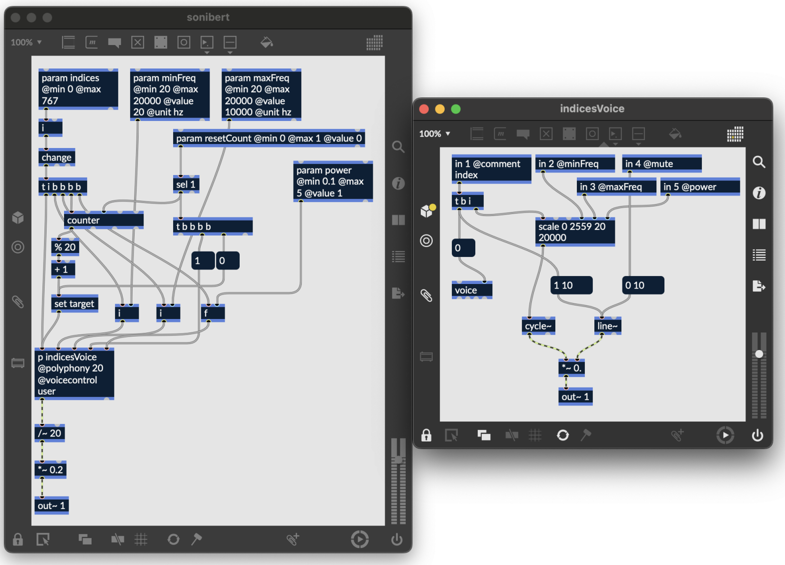

sonibert
100%
param indices @min 0 @max 767
param minFreq @min 20 @max 20000 @value 20 @unit hz
param maxFreq @min 20 @max 20000 @value 10000 @unit hz
i
param resetCount @min 0 @max 1 @value 0
change
param power @min 0.1 @max 5 @value 1
t i b b b b
sel 1
counter
t b b b b
% 20
1
0
+ 1
set target
i
i
f
p indicesVoice @polyphony 20 @voicecontrol user
/~ 20
*~ 0.2
out~ 1
indicesVoice
100%
in 1 @comment index
in 2 @minFreq
in 4 @mute
in 3 @maxFreq
in 5 @power
t b i
scale 0 2559 20 20000
0
voice
1 10
0 10
cycle~
line~
*~ 0.
out~ 1

Figure 3.5: Implementation of the bell in RNBO with main patch on the left and single sine voice on the right.

Sonic Description

As no tuning or quantising to a frequency is performed and the spread or clustering of frequencies is purely based on the embeddings, the resulting pitch(es) either match or contrast the other elements of the sonic composition. The only controls in fine-tuning are the mapped range and mapping power, which determine where the bells sit spectrally in the mix.

3.4.2 FM-Synthesis: Making Embeddings Audible (Continuous)

Still in the time of the residency, I only made the bells so far and wanted to create a sonic bed for them. Frequency Modulation (FM)-Synthesis seemed promising, as it can be potentially used in a versatile way due to its wide range of timbral and harmonic abilities. The bells could reflect short discrete points in time, and the pad could be used to display more gradual or overarching states and provide a context.

In my initial approach in the (modern)BERTophon web app, I used the FFT of the embeddings of the [CLS]-tokens. The FFT bins were grouped into eight, averaged, and then mapped to the pitch and intensity of four independent synth voices as seen in Figure 3.6. When a sentence was focused, the pad was heard as a representation of the whole sentence, while the bells provided insight into individual tokens.

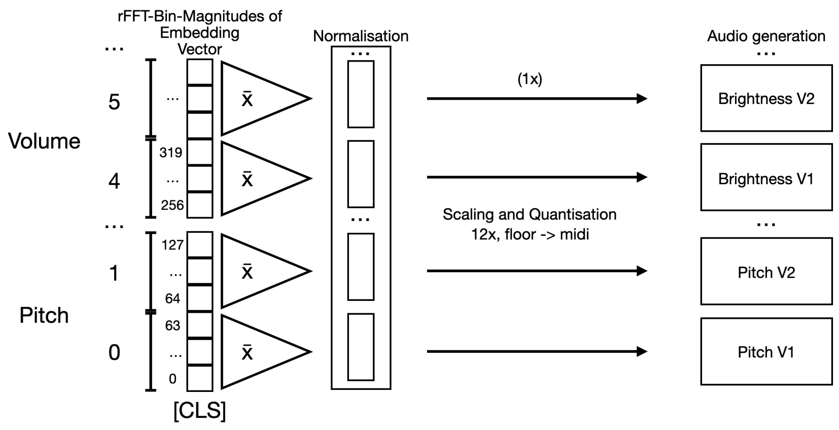


Figure 3.6: Schematic of the (modern)BERTophon Pad mapping.

To adapt or evolve this approach for the generative model, I faced different challenges. First, as the tokens are streamed one by one, the whole sentence is only apparent when it has already finished, making it impossible to normalise the values in a meaningful way without having all the tokens at hand from the beginning. Therefore, static normalisation boundaries based on the first 20 prompts of the Alpaca dataset[8] are calculated beforehand. This means that all future tokens that are generated during the experience are scaled based on the maximum and minimum values of those precalculated tokens.

A second challenge was the absence of a [CLS]-token in this model. Each new token is based on all the previous ones, and no additional embedding is generated. In the interest of time and getting a first prototype to work, I ignored this initially and doubled the playback of the bells with the changing chords by using the same token embeddings. However, as the doubling with the bells based on the same data makes little sense conceptually nor from a sonification standpoint, I finally used a moving average of the values, which results in a pad where chords interpolate constantly between the embeddings played back in a given timeframe.

The next challenge was that the new embeddings somehow sounded worse and more uniform. As the token embedding values were much noisier, this resulted in the voices mostly moving in tandem. Instead of grouping FFT bins, I used an arbitrary set of spectral descriptors, namely Centroid, Rolloff, Bandwidth, High-Low-Ratio, Peak Frequency, Flatness, Skewness, and Kurtosis[9]. Using the same normalisation and mapping strategy for these values as above resulted in much more musical results[10].

As the mapping strategy with grouping FFT bins or spectral descriptors is quite crude,

[8]See https://huggingface.co/datasets/tatsu-lab/alpaca

[9]The initial idea came from an exchange with Gemini 2.5-pro inside Cursor upon asking for a better way to turn a spectrogram into a set of numbers

[10]See UMAP vs. Spectral Descriptors (A.6) for more details

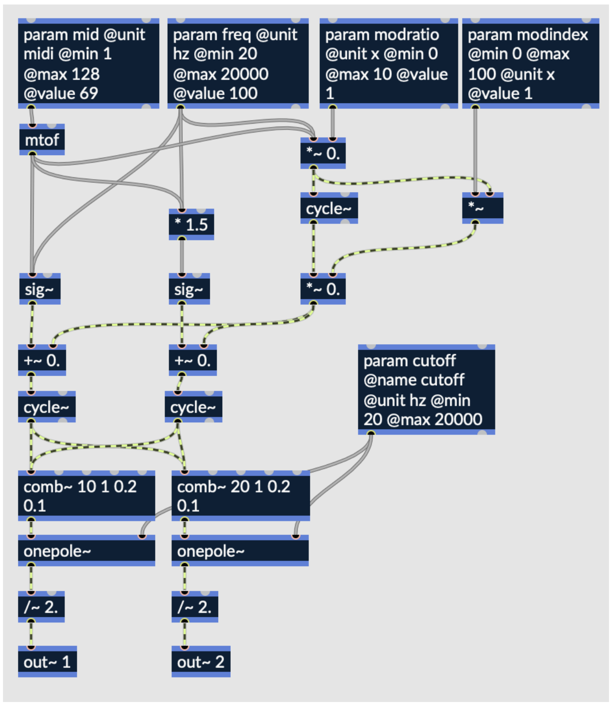
arbitrary, and does not take advantage of all the possibilities of FM synthesis, I made other attempts at building a sonic *fingerprint* using the same eight values and four voices but with a different mapping. Each voice I assigned four parameters instead of two: pitch, intensity, mod-index, and ratio. However, linearly mapping values to pitch in hertz and to mod-index and ratio yielded unsatisfactory results. Small changes in ratio can have drastic harmonic and timbral results depending on the index. Furthermore, the resulting cacophony of pitches and uneven harmonics resulted in a nauseating listening experience. To account for this, I also tried a hybrid approach, which involved quantising again to a scale similar to above for pitch, and also quantising the ratio to multiples of 0.5 to have more predictable harmonic outcomes. Still not satisfied with the result I also attempted to take the spectral descriptors analysed data into account, for instance, modulating the ratio by the flatness or the pitch by the centroid or peak frequency parameter instead of using a purely aleatoric mapping. In this version, all voices together were also filtered on the basis of the roll-off parameter. However, also this approach, I ultimately rejected in favour of aesthetics and control over the timbral note qualities.

In the end, I stuck with using the spectral descriptors in the same way as the FFT-groups were used in (modern)BERTophon, as it yielded the most musically interesting results. The evolving chords and melodies had the most variation, while the resulting patterns were still noticeable. This resulted in noticeable musical motifs that repeated throughout the playback, for instance, I noticed that in layer 19 for the currently used model, the melody always ascends and plays b, c, d at the end of a sentence.

In addition, what I call the *sonic momentum*, the volume of the pad is driven by the activity of the model or the interaction with it. When new tokens appear or a token is focussed by the user, an envelope with a short attack and slow decay is triggered that emphasises the pad and gradually fades it out. The decay time is linked to the overall playback speed and a visual fade-out of the tokens.

Technical Description

The instrument is a basic two-parameter FM-Synth with harmonicity ratio (R), controlling the ratio between the carrier and modulator frequency ($f_m = R * f_c$) and mod index (I), controlling the ratio of the modulator amplitude and the modulator frequency ($A_{modulator} = I * f_m$). To add harmonic richness, a perfect fifth is layered with the original signal and this combined sound is spread across the stereo field. Each channel then runs through an independent comb filter with slightly different settings (10 and 20 ms delay, 0.2 feedforward, 0.1 feedback) to introduce subtle variations in timbre and pitch, creating a wider, more pleasant character. Finally, a low-pass filter is added to control the *intensity* of the voice. The carrier frequency can be set with both MIDI and hertz.



param mid @unit midi @min 1 @max 128 @value 69
param freq @unit hz @min 20 @max 20000 @value 100
param modratio @unit x @min 0 @max 10 @value 1
param modindex @min 0 @max 100 @unit x @value 1
mtof
*~ 0.
cycle~
*~
* 1.5
sig~
sig~
*~ 0.
+~ 0.
+~ 0.
param cutoff @name cutoff @unit hz @min 20 @max 20000
cycle~
cycle~
comb~ 10 1 0.2 0.1
comb~ 20 1 0.2 0.1
onepole~
onepole~
/~ 2.
/~ 2.
out~ 1
out~ 2

Figure 3.7: RNBO patch of one synth voice.

Sonic Description

From a compositional standpoint, I used twelve notes from a mixture of d-dorian/mixolydian that I offset by a fifth, octave, and twelfth to give each voice their register, which, when randomly set, results in mostly pleasant chords. Through the varying intensities of the voices, especially if similar intensities stretch over multiple tokens, some voices stand out and sometimes form a melody. After having implemented the moving average interpolation, this became highlighted more as the voices form evolving chords that have a more or less noticeable note glissando in-between. Through the changing bass tones, this breaks the composition out of the modal bed that is established by the other components.

The chosen FM parameters give the voices a similarity to a lingual register of an organ such as the *vox humana*. Although this was not an active decision, in hindsight, it further blurs the line between artificiality and human nature.

3.4.3 Granular Synthesis: Manipulating Organic Material

This plug-in was made for the User-Sonification (3.1), translating the user's physical movements into sound by manipulating organic source material. I chose to add granular synthesis as it is a good way to manipulate existing recordings in a dynamic way. I could use this to satisfy two goals, stemming from the root concept of having an elongated pad and short transient figures, like the machine pad and bells. First, to sonify the user's general movement, and second, one way to make their immediate movements audible.

First, I wanted to have an organic counter part to the machine-pad using a real instrument. For this, I recorded a cello pad, overdubbing multiple layers, starting quietly, with slow and uniform bow strokes (played *flautando* and *flageolet*) and ends in rapid irregular bow changes played loudly near the bridge (*sul pont*). The flageolet notes oscillate between a, b, and c, resulting in a d-dorian or -mixolydian feeling. When the user moves more and the granular pad becomes stronger, it shifts to the notes g, d, and a creating a soundscape that opens up harmonically in addition to the volume increase.

For the mapping, the user's total movement is averaged and filtered with a rate-limited parameter with a slow speed. This value is normalised and assigned to the playback position of the granulator. A user that generally moves more therefore hears a louder, brighter, and rhythmically more complex pad, reflecting their overall activity. This is what I call *sonic momentum* for the user pad.

Figure 3.8: Photo from the koto recording session.

For the immediate movements, I chose another instrument, the koto, tuned to a d-major pentatonic scale. I recorded arpeggios applying varying levels of string damping with the free hand to capture a range of muted tones. For the mapping, the average velocity of the user's finger curvature was assigned to the volume of playback and the angle of the wrist to the sample position. Moving the fingers with the palm facing upward results in a more muted sound than with the palm facing downward. This transforms their hand into an invisible multi-dimensional instrument they can *play* in the air.

Technical Description

For this plug-in, an adaption of the RNBO-FMOD-Wrapper was necessary to support the loading of samples, which was not possible until then. This involved creating a data parameter for every buffer~ in RNBO. This results in file drop zones in the plug-in that pass the raw bytes (See Figure 3.9). Using dr_libs[11] this data is then read and/or decoded and passed to RNBO.

For the architecture, I closely followed a YouTube tutorial by *toneparticle*[12], which implements a granular synth in gen~. To make it slightly more performant, the maximum number of simultaneous grains was halved to 15. The controllable parameters are: position, grain length, pitch, stereo-spread, spray (randomised position offset), spread (randomised pitch offset).

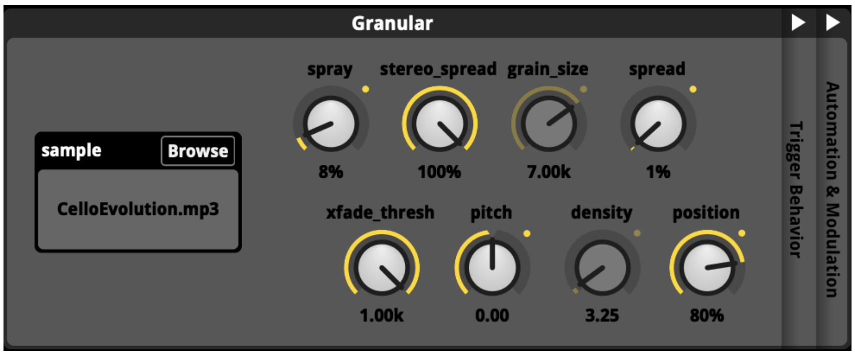


Figure 3.9: Granular plug-in, displayed inside FMOD Studio.

3.4.4 Waveshaping: Pad Accentuation

The preceding plug-ins were designed for the generation of either the Machine-Sonification or User-Sonification. This and the next ones are effects that either highlight or try to combine and blur the lines between organic and digital.

This plug-in is only used on the cello pad and complements the koto instrument by adding another type of immediate feedback: the user's head movements. It temporarily distorts the signal and makes the pad stand out while the user moves. Faster head movements therefore result in a louder and brighter pad.

Technical Description

This plugin is very simple, as it uses the predefined tanh~ object of RNBO. The distortion is based on what is also called *soft-clipping* and distorts the signal using the hyperbolic tangent formula (3.1)[13]. The only parameter is a pregain that is automatically compensated for after

[11]See https://github.com/mackron/dr_libs
[12]See https://www.youtube.com/watch?v=VU2TQmxte9A
[13]The MathWorks, Inc., 2025.

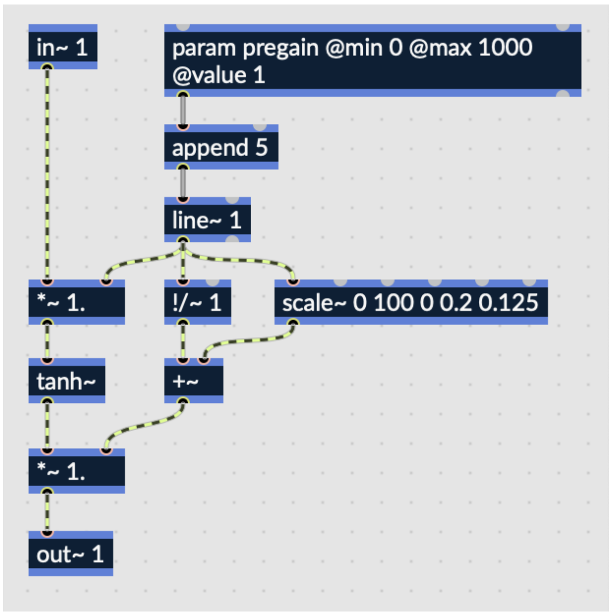


Figure 3.10: Distortion plugin with automatic gain compensation.

tanh~. Compensation takes place in two steps. As an inversion of the pregain curve ($1/pregain$) and a correction boost curve to account for the lost energy of the clipping that I have experimentally found out using pink and white noise (See Figure 3.10). The user's head movement, positional and angular velocity combined, is mapped to the pregain. To ensure that outliers in the tracking are not heard, the parameter is still filtered slightly, but only to the extent that it still feels reactive and imminent.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{3.1}$$

3.4.5 Source-Filter Synthesis: Adding a Human Touch

This plug-in serves my goal to give the synthetic sound of the machine a *human* touch and to blur between the user and the machine. Inspired by the vowel / formant filter section in Grond and Berger's PMSon chapter (Grond & Berger, 2011, pp. 378–379), I developed a formant synthesis plug-in, which boosts frequencies on the machine pad based on the characteristics of human vowels.

The mapping is designed to be directly tied to the user's interaction. It is laid out in such a way that the vowels are linearly spread across all the layers of the model. As the user goes through the layers from top to bottom, the vowel changes through the sequence A-O-U-E-I. Through the nature of the vowels, the vowel *A* passes the most frequencies and *I* the least, resulting in a diminishing volume and narrower focus going from higher to lower level. To create more variation depending on the user's behaviour, the voice register is controlled by the user's behaviour on how long he listens, and the dry/wet mix is increased the longer the user spends interacting with the model, gradually revealing the vowels.

Additionally, I also added formant filtering in the Resulting Design Iteration (4.4) to the user pad and translating movements of the hand, also played by the cello. The longer the user listens, the more those elements become filtered, controlling the register and vowels with their hands.

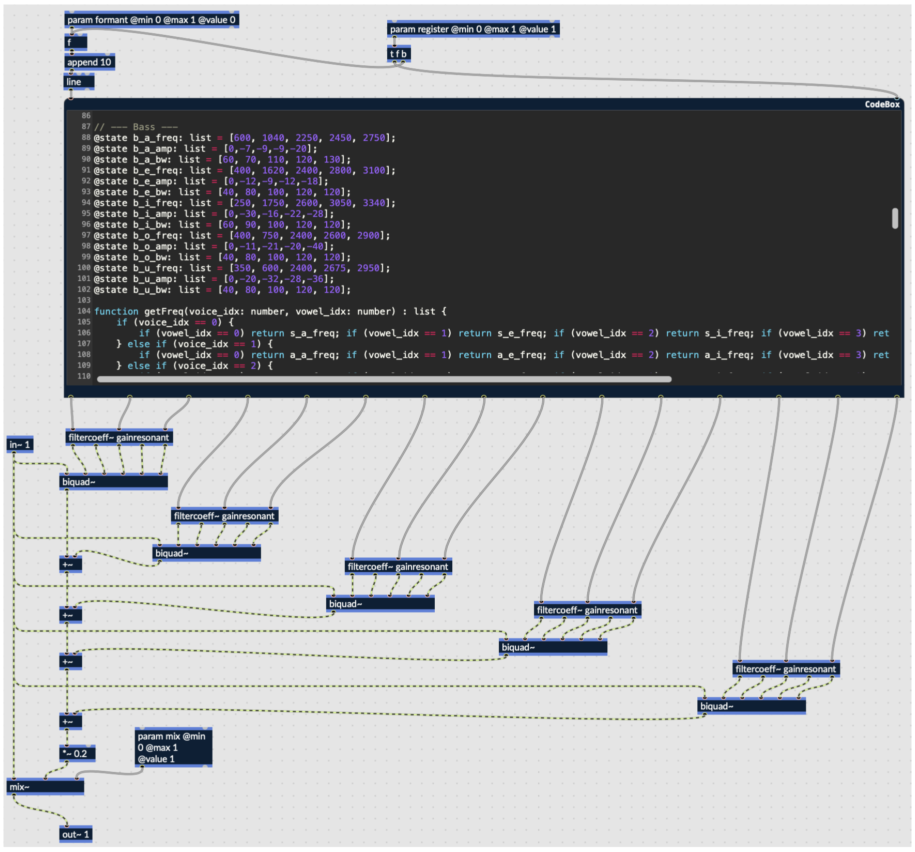
Technical Description

Formant filtering is based on the concept of formants, resonant frequencies that give spoken vowels their characteristic sound. To do this digitally, a set of parallel resonant filters can be used. Although the frequency (f), amplitude (a), and bandwidth (bw) of the formants could be experimentally determined, a standard data table from the Internet was used[14]. It offers the five strongest formants for each vowel four registers: soprano, alto, tenor, and bass. However, the fun really starts when interpolating between vowels and registers. This is done by doing a piecewise bilinear interpolation for all the filter parameters. To create smooth transitions between vowel sounds and voice types, the system continuously calculates intermediate filter settings using a two-step interpolation process. The register parameter is divided into three segments, the vowel parameter into four segments. Depending on where you are, a lower and higher register is selected along with a lower and higher vowel. Then it interpolates between the lower and higher registers of the lower vowel and between the lower and higher registers of the higher vowel. Those two intermediary sets are then linearly interpolated on the basis of the vowel parameter. This is done independently 15 times for all filter parameters (f, a, bw) of the five filters. For this, I used the codebox feature as I found it more appropriate to do it this way instead of relying on nodes. As the type of code is close to Java script, I was also able to ask an LLM (Gemini 2-5-pro) to help me with the interpolation logic, which would not have been possible with node-based programming.

3.4.6 Waveguide Synthesis: Complementing the Koto Digitally

Also concerning changing the layers of the model, this plug-in provides direct tactile sonic feedback. On the conceptual level of blurring the lines, it is a *digital twin* of the koto, simulating a plucked string through Karplus Strong synthesis. Moving their right hand up and down, the sound is triggered on each layer change. The speed of the user's hand during that change acts on the loudness and brightness of the pluck. Faster movements therefore result in a more

[14]Piché and Nix, 1997.


param formant @min 0 @max 1 @value 0
f
append 10
line
param register @min 0 @max 1 @value 1
t f b
CodeBox
in~ 1
filtercoeff~ gainresonant
biquad~
+~
*~ 0.2
param mix @min 0 @max 1 @value 1
mix~
out~ 1

```
// --- Bass ---
@state b_a_freq: list = [600, 1040, 2250, 2450, 2750];
@state b_a_amp: list = [0,-7,-9,-9,-20];
@state b_a_bw: list = [60, 70, 110, 120, 130];
@state b_e_freq: list = [400, 1620, 2400, 2800, 3100];
@state b_e_amp: list = [0,-12,-9,-12,-18];
@state b_e_bw: list = [40, 80, 100, 120, 120];
@state b_i_freq: list = [250, 1750, 2600, 3050, 3340];
@state b_i_amp: list = [0,-30,-16,-22,-28];
@state b_i_bw: list = [60, 90, 100, 120, 120];
@state b_o_freq: list = [400, 750, 2400, 2600, 2900];
@state b_o_amp: list = [0,-11,-21,-20,-40];
@state b_o_bw: list = [40, 80, 100, 120, 120];
@state b_u_freq: list = [350, 600, 2400, 2675, 2950];
@state b_u_amp: list = [0,-20,-32,-28,-36];
@state b_u_bw: list = [40, 80, 100, 120, 120];

function getFreq(voice_idx: number, vowel_idx: number) : list {
    if (voice_idx == 0) {
        if (vowel_idx == 0) return s_a_freq; if (vowel_idx == 1) return s_e_freq; if (vowel_idx == 2) return s_i_freq; if (vowel_idx == 3) ret
    } else if (voice_idx == 1) {
        if (vowel_idx == 0) return a_a_freq; if (vowel_idx == 1) return a_e_freq; if (vowel_idx == 2) return a_i_freq; if (vowel_idx == 3) ret
    } else if (voice_idx == 2) {
```

Figure 3.11: Excerpt from the RNBO-patch showing a codebox in conjunction with resonant filters.

pronounced note. The user is free to use this mechanic for its intended layer change action or *abuse* it as an additional transient instrument.

Technical Description

This plug-in implements the Karplus Strong algorithm. Developed in gen~, a short burst of pink noise is passed through a very fast delay-feedback loop, resulting in a sound that resembles a plucked string. Based on a YouTube video by tune4media[15], I implemented an improvement to add a low-pass filter (LPF) that quickly sweeps down in the first few milliseconds of the transient to give a more pronounced and brighter pluck sound. Furthermore, I implemented an improvement over the video approach that adds a uniform decay that is specifiable in milliseconds and independent of the pitch of the note. There, a first step towards it was described using the *t60* object, which calculates the per-sample feedback factor (d_{sample}) needed to reduce the input by -60dB (10^{-3}) over the desired duration of T samples(3.2).

$$d_{sample} = 10^{\frac{-3}{T}} \tag{3.2}$$

[15]See https://youtu.be/Yb8JuFHJxQQ?si=dtvSmxcEoEOtfLEn

However, the delay is only calculated for each feedback cycle (with the length in samples N). To account for this, the final per-cycle feedback factor (d_{cycle}) is calculated by raising d_{sample} to the power of N as seen in (3.3)[16] and Figure 3.12.

$$d_{cycle} = (d_{sample})^N = \left(10^{\frac{-3}{T}}\right)^N = 10^{\frac{-3 \cdot N}{T}} \tag{3.3}$$

Taking also from the video, I introduced a velocity parameter that determines on the one hand the start frequency of the LPF to create a brighter transient when *plucked* harder and, on the other hand, sets the overall volume.

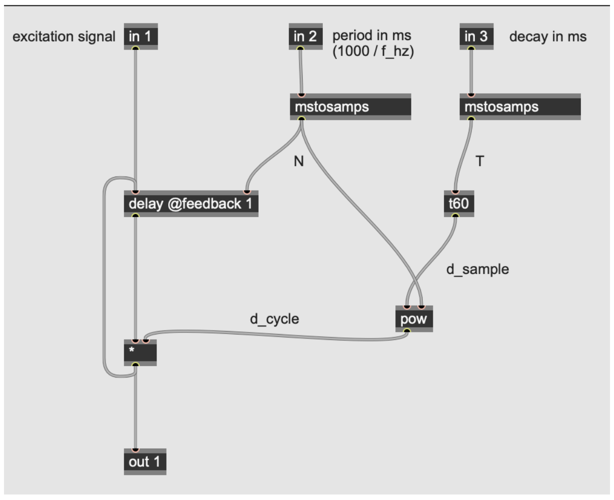


Figure 3.12: Gen-patch for the Karplus Strong synthesis

Sonic Description

The layer indices are mapped to a d-dorian scale that fits both the chords played by the machine pad and the open fifths by the user pad.

3.4.7 Cross Synthesis: Blurring the Lines Further

This plug-in was made to further blend different elements of the sonification by being able to explicitly fuse two signals together taking the timbral content from one and the harmonic from the other. With it, it was possible to combine the text-to-speech voice (See Text to Speech (TTS) Pipeline (3.6.2)) with the machine pad. Although the plug-in existed early on, it was not implemented until after the user tests in the last design iteration. There, the mix in how much the voice is blended is determined by the user's behaviour, the more the user spends speaking, the less the model is affected by the vocoder and the more it sounds natural. Originally, I was

[16]Problem determination and needed correction found out through chat with Gemini 2.5-pro

also experimenting with using this effect to blend the machine and user pad together, but I rejected the idea because the sound quality was not high enough in this case for unspatialised signals and too taxing on the processing power.

Technical Description

It implements a simple phase vocoder by calculating the FFT of two input signals, converting the real and imaginary parts of the FFT-bin to amplitude and phase (Cartesian to Polar), taking the amplitude of one signal and the phase of the other, and reversing everything by converting back to Cartesian and computing the inverse FFT. The FFT is windowed with a Hann window and calculated twice, the second offset by $nFFT/2$, which results in what is called the overlap-add method where the FFT frames overlap. This effect can thereby merge two signals into one, taking the timbral character of the first one and the pitch information of the second.

3.5 Managing Input in Virtual Reality

The following two sections describe techniques that help to make the interaction as intuitive and direct as possible in VR. Using the own voice as text input with text-to-speech and the own hands as means of interacting with the model with hand tracking.

3.5.1 Speech to Text (STT) Plug-In

As FMOD does not provide an option to use microphone input in the web-target, another solution had to be found. To minimise the required processing power, Unity's audio engine was left disabled, and I built a JavaScript plug-in for Unity based on the Web Speech API[17][18]. The plug-in implements the interface for *SpeechRecognition* which works in modern browsers except Firefox. In Unity, it is then possible to select the language, start and stop the recognition and receive the final as well as the interim transcriptions as an event.

3.5.2 Tracking Head, Hands and Gestures

When developing for VR using a device that is tracked in six degrees of freedom (6 DoF), you get the position and rotation of the head for free as this is needed for driving the camera movements.

Hand-tracking allowed me to get on one hand more information about the user's movement and, on the other, let the user control the application with their own hands. Those interaction

[17] Mozilla Developer Network, 2025.
[18] With help from Gemini 2.5-pro in Cursor.

possibilities also enable it to become more intuitive and intimate, without having to rely on controllers.

Unity XR Hands is currently marked as a *pre-release* package[19] that enables interactions with the hands tracked by a VR headset and is already supported by the WebXR exporter. It gives access on a low level to 25 reference points on each hand, called joints, from which position and rotation are exposed. They come with some precomputed measurements, such as their velocity or distance between the palm and the thumb. On a higher level, finger shapes (curvatures of fingers) and hand poses (finger shapes + hand orientation) can be defined, and actions can be run as soon as the user holds the respective shape or pose for a specific amount of time, called static *gesture*. I adopted the term *gesture* from the XR Hands documentation, although I find it misleading, as I mainly think of gesture as evolving poses over time. The work with dynamic gestures would require the use of some interpretative machine learning algorithm such as that provided with the motion analysis tools of the AI-Toolbox of Daniel Bisig at ICST[20]. Similarly to the data from the LLM, however, I wanted to work with the raw data instead of an interpretation of it. This resulted in relying on the static gestures provided in conjunction with joint translation or rotation. Additionally, I used manually calculated low-level descriptors like linear joint velocity (movement/time), acceleration (velocity/time) or jerk (acceleration/time).

In the final experience, the following hand gestures are defined:

- **Both open palms up**: Start listening to speech (lowering disables it)
- **Open palm down right**: Moving palm up and down switches through the layers.
- **Open palm down left**: Moving palm up and down modifies speed; moving left and right switches between the two playback modes.
- **Pointing index finger of either hand**: Activates a pointing ray with which tokens can be selected in the interaction state.
- **Both index finger pinched**: distance between both index fingers scales model; average translation of both hands, translates model; rotation around the centre point between both index fingers, rotates model.
- **Double shaka**: Resets the view to default.

The positional and rotational velocities of the palms and fingers are used in addition to those of the head to determine the general activity, all weighted in equal parts.

[19]See https://docs.unity3d.com/Packages/com.unity.xr.hands@1.7/manual/index.html

[20]See the GitHub repository here https://github.com/bisnad/MotionAnalysis/tree/main/MocapAnalysisPython

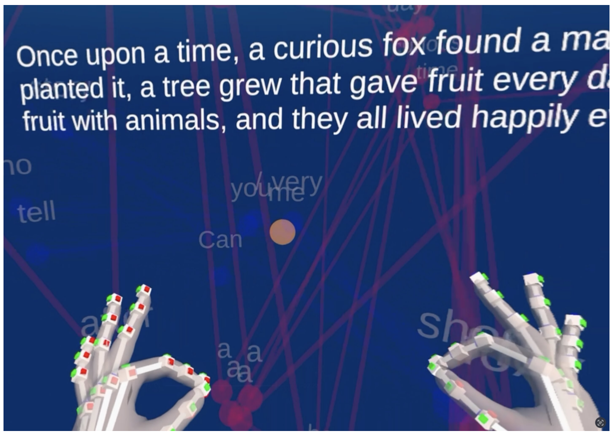

Once upon a time, a curious fox found a ma
planted it, a tree grew that gave fruit every d
fruit with animals, and they all lived happily e
tell
you
very
me
Can
a
a
a

Figure 3.13: Double pinch gesture for manipulating the position, scale and rotation of the tokens

3.6 Further Audio Implementations

Moving from the DSP plug-ins and their usage, this section elaborates on further audio elements that formed the whole experience. This includes additional user feedback through separate sounds, giving the model a voice through an audio generative neural network, and placing sounds in the space through binaural rendering. .

3.6.1 UI-Sounds: User Feedback

Two gestures received additional acoustic feedback. First, the start and stop of speech recognition and the switch between playback modes, which have been implemented only after the user tests.

The first one indicates to the user when the gesture is recognised and they can or cannot speak. The second provides additional feedback to the visual switch of the symbols.

Sonic Description

The signals are tuned to the general harmonic landscape. The start sound is a highly reverberated auditory signal with a fifth e-b ascending. It is a mixture of two notes from the karplus-strong synthesiser in quick succession and a layered woosh sound. Harmonically dominantic, it creates some tension that is released by the stopping sound, indicated with two notes a-d quickly descending in the same instrumentation. However, with a longer tail and accompanied by a metallic shimmer, which evokes sending a message *out and away to the model.*

The mode switch is played by an aggressive snap pizzicato from the koto alone, playing the note a. The note also has an unresting quality, being the fifth. It has a wide spectral footprint and is heard well through the mix, which is also helpful as an acoustic indicator if the user changes the mode by accident.

3.6.2 Text to Speech (TTS) Pipeline

For the first step in reaching the design goal of providing the model with a voice, the formant plug-in was built. However, to make it more concrete and make the experience more accessible, the generated text of the model is read aloud. As tokens are meant to be played back one by one here and aren't always full words, this becomes an interesting challenge. While it would be possible to convert each full word to audio, the natural flow of speech that the TTS model has learnt would be lost. The chosen solution is to generate the text in full length and cut it afterwards based on full words. This results in some tokens not having any text, as they're either punctuation or a continuation of a word.

The audio pipeline for TTS consists of the following three-step process. First, the fully generated text is sent to a TTS model called *Kokoro* by Hexgrad[21], which is one of the fastest and most expressive open-source speech generation models to date, that can be run locally. Second, the generated audio is again analysed by *Whisper* by OpenAI[22] to find the timestamps of the words. The timestamps are on the one hand imprinted onto the full audio file as cue markers, and on the other hand used to split the full audio into single word chunks. (More details on this are provided in Time-Based vs. Audio-Based Playback (4.4.1)) All audio files are hosted to be downloaded by the front-end. Finally, the token indices are matched with the timestamps, and download links are sent to the front-end, which downloads them immediately. The audio file is then played back through a programmer sound in FMOD, taking advantage of its signal chain with spatialisation and reverberation.

3.6.3 Spatialisation in the Web: Steam Audio

At the time of writing, the HRTF spatialiser shipped with FMOD called *Resonance Audio* from Google[23] is not compatible with the web target. It has been deprecated and is inflexible as all spatialised objects are routed to a single *listener* plug-in. *Steam Audio* by Valve Corporation[24] is a free and since early 2024 open source spatialiser, which also features ray-traced reflections and reverberations, spatialises per object directly in the signal chain, and even provides the option to choose personalised SOFA files to calculate the HRTF. As its Web support is experimental with no official binaries, the library had to be built from source. Therefore, the binaries had to be built from source. This also involved removing some lingering code for multithreading, which is not supported in web assembly.

[21] See https://kokorottsai.com/
[22] See https://openai.com/index/whisper/
[23] See https://github.com/resonance-audio
[24] See https://valvesoftware.github.io/steam-audio/

4. User Testing and Design Iteration

After three months of development, I conducted a small qualitative design study in July. It was based on a minimum viable product that provided all planned interactions but was missing a large part of the planned features of the User-Sonification (3.1). So far, I had only implemented the general movements of the head and fingers that affect the granular cello and koto pad and the change in layers. The behavioural impact and other planned movements were still missing. To give an idea of what the participants experienced, I first provide The User Experience of *Dialog* (4.1) at the state before user tests. Then, I describe the Goals and Design of the User Tests (4.2). It could not be postponed in order to respond to feedback, which turned out to be highly valuable, for which more information is found in Findings (4.3) and Resulting Design Iteration (4.4). In the end, the Final Prototype (4.5) after the iterations is presented using a video.

4.1 The User Experience of *Dialog*

In this section, I describe the entire experience with all the interaction and sonification elements in context, which I finished before the user tests to gain a better understanding of what the users experienced.

The user was surrounded by a blue environment that is darker at the bottom and lighter at the top. The own hands as a stylised model could be seen. Two quiet pads from the machine- and user-sonification could barely be heard.

The experience was structured into sections or states that will later become the basis of calculating the user behaviour sonification. However, this was not completed prior to the user study. These states are described linearly here, but could be interrupted by the user, for instance, by switching to the speaking state at any moment.

Speaking State Holding both palms up activated the Speech to Text (STT) Plug-In (3.5.1) indicated with a reverberated auditory signal. Lowering one or both palms resulted in stopping the *machine's* listening, indicated with another signal. While the two-palm gesture was held, the user could speak, which was transcribed and sent to the back-end, as soon as the gesture was released.

Listening State The tokenised input was immediately sent back to the front-end together with its calculated descriptors and 3D feature projection. The token playback started, and one token after the other was placed into the scene. The input tokens were displayed as blue spheres, animated flying from leaving the user's mouth to one metre in front, the bell for its embedding was heard, and the pad's voicing was adjusted. The token hovered for a brief moment there until finally flying to its 3D position on the basis of the embedding's feature projection. This happened for all the input tokens sequentially, with the addition that an animated line was drawn between the tokens.

At the same time, the back-end calculated the whole response, generated the speech and all required steps thereof, sent the data to the front-end, which also downloaded audio files. This should have ended all before the input tokens finished animating. However, depending on the length of the input or the length of the response, it might have led to a pause where not much happened. The existing token visuals could be seen to slowly fade, and the pad also gradually became quieter again.

In this pause, the user was able to still interact with the model and experiment with the sounds they could make. I use this here to describe the present user-sonification elements.

All translocations and rotations of the hands, its finger joint angles, and head movements were averaged and used to drive the positional parameter of the granular synthesiser. Rotating the head resulted in amplitude envelopes that resembled waves with shorter attacks and longer releases. Bending the fingers caused the koto pad to be heard. The layer change interaction was also available, holding the right palm down and moving the wrist up and down, which resulted in playing the Karplus Strong synth and manipulating the position of the existing tokens for that layer.

The back-end now finished and all data received, the main token-playback started. One by one, the reasoning tokens in green appeared first, followed by the response tokens in red. This happened with an animated line with the same colour from the current to the next token position. Additionally, from the three most attended tokens, white lines extended to the final position. At the end of the animation the sphere appeared, the pad was adjusted, and the bell and the spoken word were heard from that position. However, the spoken word was only heard for tokens that are the beginning of words.

Interaction State After the playback had finished, the user was able to interact with the model by pointing one of the index fingers at tokens, which lighted them up, the word was spoken, the bell was heard, and the pad's voicing adjusted again. By pinching the index finger and thumb of both hands, the tokens could be scaled, translated, and rotated, so that it was possible to explore tighter token clusters that appear in lower layers. It was also possible to become fully immersed by placing them around oneself.

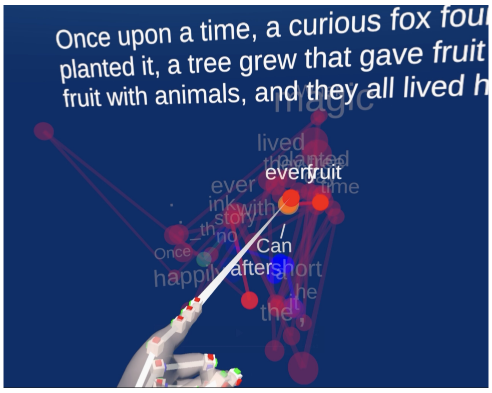

Once upon a time, a curious fox fou
planted it, a tree grew that gave fruit
fruit with animals, and they all lived h
magic
lived
planted
ever
fruit
time
ink
with
story
no
Can
Once
happily
after
short
he
the

Figure 4.1: Pointing at tokens for repeated playback

4.2 Goals and Design of the User Tests

The aim was to figure out what interactions work and how the whole experience or the elements of it are perceived. The main interest also was how users respond to the Machine-Sonification (3.1) and User-Sonification (3.1).

The 13 participants had different backgrounds and expertise in all age groups except children. Game designers, musicians, AI researchers, and other people from unrelated fields with a broad difference in knowledge about sound synthesis, LLMs, and virtual reality. The group consisted of family members, colleagues from the University of different disciplines, and from the IVIA lab.

A testing session took thirty minutes to an hour, where possible interactions were explained first. After setting up the headset, the participants freely tested the experience as they preferred. If they had questions during the experience, I was there to answer them. After they were no longer interested, a small interview followed that incorporated the same questions for everyone and individual questions based on what the participants did or said during the experience. The session ended with a form that included demographic questions, the self-assessment tool (SAM) (Bradley & Lang, 1994) to have a simple way to evaluate, how the experience was perceived using a visual manikin as reference, and additional optional feedback. This form was filled out by 11 participants.

4.3 Findings

As seen in Figure 4.2, SAM showed a high convergence for the pleasure and dominance parameters, describing the experience as mostly pleasant and tending towards full control. For arousal, the experience was generally described as calm, but some participants tended towards having had an activating experience. This can be explained in the diverse user reactions, with some people moving around more and even standing and walking around (2) or trying to make music using focussing or changing through different tokens and layers (2), while others were locked in on the model, trying to understand the logic of the word clusters (3) while being unaware of their effect on the sound (9). Therefore, most of the participants did not actively perceive their effect on the User-Sonification (3.1).

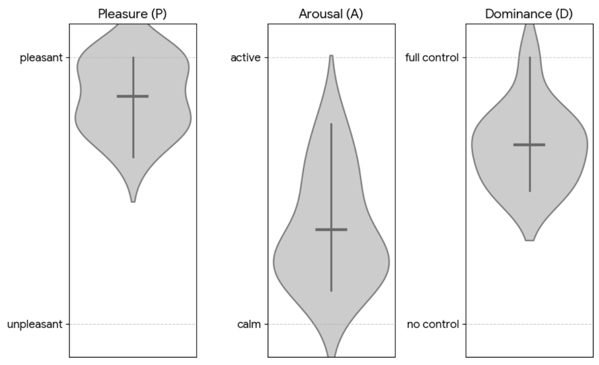


Figure 4.2: Violin plot with results from SAM.

The fixed-time token-per-token playback was described by three participants as jarring or incomprehensible, while one explicitly described it as a pleasant aesthetic comparing it to spoken-word poetry. Two participants mentioned the unnatural cuts in the words, criticising either the understandability or missing the ability to reliably recombine words ”to make the model say something different”. However, most of the participants (8) did not pay close attention to the content of the responses, as it was too slow (6), not understandable enough (3) or illegible due to vision impairments (1). The playback of the models reasoning tokens were also way too slow, resulting in one conversation iteration taking two to four minutes. Furthermore, the final output was not interesting enough, which gave little or no incentive to continue a conversation. Seven only spoke one prompt and spent the rest of the time exploring the scene. However, four tried multiple different inputs, either patiently waiting for the model to finish (2) or rapidly trying multiple different prompts in rapid succession (2).

Five participants explicitly mentioned that they felt the character of this model entity, feeling sorry for *it* (3) because of the tone of voice (2) and/or its over-the-top/overthinking reasoning effort for simple questions (2). Two participants stated that they appreciate the representation of AI through audio (1) and the point cloud (2) to see or hear what is happening internally, favouring it over explicit depictions of animated personas or abstract animations like the moving watercolour approach for ChatGPT Voice-Mode or Apple's Siri.

For the first participants, the speech-to-text gesture and the general input procedure were generally confusing. They left the hands resting in the input gesture until the internal timer just fired the prompt, presumably through an undocumented timer from the Speech API. This was likely due to a lack of direct feedback on what was registered or understood by the app. To focus on the other interactions and avoid frustration, in subsequent tests, the participants were therefore explained to rest their hands again after speaking, which worked well. However, user feedback would need to be improved here.

The favourite interaction was the double-pinch navigation. Two people immediately mentioned feeling like *Tony Stark*. Other participants mentioned that this interaction was valuable for exploring (1) or to immerse themselves in a vector model/being surrounded by an LLM (3).

Three participants criticised not having any incentive or *quest* to do, needing to come up with some input without any prior context. Similarly, two missed having some form of evolution of the character or the sonic landscape, which made them lose interest.

Despite some criticism, the tests were satisfactory in showing that the application provided the affordance to explore the experience in different ways, where varying personalities could take something different from it. However, in order to reach the user sonification goal, the user's action should be more reflected in the sonic landscape. There should be an alternative to the fragmented word playback. The responses should ideally occur faster or be shorter in length, and the playback speed of tokens controllable by the user.

4.4 Resulting Design Iteration

This section describes the changes made or planned based on the user tests. The main focus is the sonic part, and potential technical aspects are hereby mentioned but not elaborated further:

- Facilitated model switch (implemented)
- Improved feature projection and normalisation (implemented)
- Performance improvements for faster response time (implemented)
- Different languages (interaction not yet implemented)
- Chunked TTS streaming response for drastic reduction in response time (not implemented)

4.4.1 Implemented Changes

User Behaviour Impact

The three states mentioned in The User Experience of *Dialog* (4.1), *Speaking*, *Listening* and *Interacting* are tracked in time and divided by the entire runtime. The resulting percentages are assigned to parameters. The longer the user spends talking, the more the voice sounds natural by adjusting the mix of cross-synthesis between the generated voice and the machine pad. Cross-synthesis was now newly added. The voice is taken as the carrier and the pad as the modulator, resulting in a classic vocoding effect where the voice *sings* with the chords of the pad. My idea was that the user influences the model. The more the user speaks, the more natural the voice of the model becomes.

The longer the user spends interacting, the more the machine-pad becomes affected by the formant filter. Similarly to the above, the more the user interacts with it, the more the sound takes up a human quality.

Lastly, the more time the user spends listening, the more formant filtering of the user pad and hand movements is added. The formant filters that affect the user sonifications can be controlled by the elevation of both hands. The right hand controls the formants similarly to when the formants are changed for the layer interaction, the left hand also controls the register of the formants. With this, I wanted to shift the focus of the user's attention to his own sound and invite experimentation there. As the listening state is finite and bound to the amount the model has to say and then switches back to interaction, the effect fades out gradually, leading the focus more again on the interaction.

Unfortunately, these effects have not been evaluated. Through their subtle nature, I fear that they might be not noticeable enough.

Additional User Sonification

Together with the user pad recordings, I also recorded improvisations of varying registers and speeds and bow strokes of different intensities.

The phrases are spatialised to be heard coming from the user's hands with the volume bound to the hand's speed. Different heights of the hands result in different registers. Increased velocities result in faster movement, at least in the middle register so far. To also give a more transient and punctual feedback, when a certain velocity and jerk threshold is met, bow strokes at different intensities based on the acceleration are played back, also coming from the hands. The note is pitched to the notes *a'*, *d'*, *a*, and a gradient goes 2 octaves lower, depending on the elevation of where the trigger occurred. These are the above-mentioned hand elements that are also passed through a formant filter that is controlled with the height of the hands and its mix determined by how long the user listens.

Time-Based vs. Audio-Based Playback

After the testing, I developed an additional mode that links the playback of the tokens to the uncut spoken text. The user is able to change between the two modes as they want, by a newly introduced hand gesture moving the left hand facing down, left and right. An icon is displayed above the hand to show the current mode. Here, both modes are compared and described further.

Time-Based The main goal of the token-by-token time-based playback was to allow the user to perceive the sonic characteristic of the token. This makes a lot of sense for BERTophon, where the user might be curious how the input, and therefore known words to the user, might be turned into sound. Here, however, where the interaction is more in the main focus, waiting for the output to advance becomes too tedious. Additionally, the word-stamp predictions from Whisper are sometimes quite imprecise, resulting in cut-off or conjoined words, feeding this issue further.

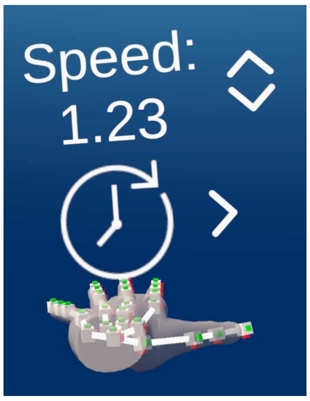


Figure 4.3: Time-based icon and hand interaction

Audio-Based In this mode, I trigger the generation of the visual tokens and sonifications based on the markers imprinted on the whole generated speech, based on the same Whisper analysis on which the cutting for time-based playback is based. However, only the onsets are needed, which are generally more reliable in this case. Among the single-word audio snippets, the whole audio file is downloaded by the front-end and played back by an FMOD programmer instrument. The token playback now occurs through FMOD's sync point callback, following the rhythm of the voice.

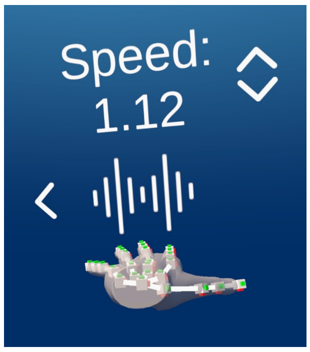


Figure 4.4: Audio-based icon and hand interaction

Speed Change Gesture

As users became more familiar with the system, the default token playback was too slow and participants lost interest. Upon realising that during the tests, I offered them to accelerate it through the Live Tuning Panel (A.2.2), which they appreciated in hindsight.

In response to this feedback, I first considered gradually making the tempo faster over the course of the experience, but ultimately decided to give the user the option to change it themselves. I implemented a hand gesture moving the left hand up and down with the palm facing down, similar to the layer change in the right hand. Also tied to the playback speed is the pitch of the spoken voice, which serves as feedback and to avoid overlapping speech. This pitch shift is also applied for the audio-based playback so that the voice can be made talking faster or slower, which also results in faster or slower token appearance without any additional work.

The *sonic momentum* and the interpolation window of the machine pad are also affected, so that the pad chords decay and move faster.

To Think or not to Think...

With the model used[1], controlling the length of the chain-of-thought (CoT) process is not reliable by modifying the system prompt[2], and only the response is kept short. It is also not possible to specify it through its architecture, such as the newly released gpt-oss, which includes three levels of reasoning effort[3].

As the model also supports responses without reasoning with dynamic switching, I implemented hardcoded prompt keywords called ”please do not think” or ”please don’t think” to disable reasoning and ”please think” to re-enable it. This is done by converting the words inside the

[1]See Inference in *Dialog* (A.4)

[2]A set of text instructions that can be set in the back-end. It includes, for instance, the request to ”Keep your responses concise and to the point.”

[3]See https://openai.com/index/introducing-gpt-oss/

prompt using regular expressions to "/think" and "/no_think", which the model understands. Having thinking disabled therefore results in quicker and shorter responses that allow for a more direct conversation.

4.4.2 Incomplete or not yet Implemented Changes

Focus-Based Playback Improvements

After the main playback is done and all the tokens are visible, the user can play and focus on them, pointing with the index finger, to replay them in the different layers. The embedding bell and the generated voice, where present, sound at the same time. In addition, the voicing of the pad is updated. This is potentially too much information and too overbearing, and it would be preferable that the user could select which type of sonification should be played back. This could be done by assigning a technique to a hand, so that pointing with the left hand would result in hearing the bell, muting the pad, while pointing with the right would only update the voicing of the pad.

Impact of Conversation History (No Result Found so Far)

It would be preferable if the longer the conversation goes, the general mood, tonality, be it scale or timbral qualities or additional parameters of the machine-pad would shift in some way, based on the whole conversation history. This could potentially help bring movement to the now static modal material.

An approach could be to repurpose the machine pad for this. This would also conceptually make more sense here to have a bed based on the context of everything and punctual bell sounds for the single tokens. The current moving average of the pad takes the last few seconds into account, as anything that is much longer results in hardly moving chords, which makes it compositionally less interesting. This suggests that the average, especially when taking all the previous tokens into account, is on a much more subtle scale than the current chord quantisation can make audible. Those differences would be interesting to sonify using delicate ratio or index settings; however, no solution has been found yet that is also aesthetically pleasing, and more time and resources would have to be spent here.

Interim Speech To Text Feedback (Not yet Implemented)

The acoustic feedback at the start and end when the front-end listens was not enough to make the interaction intuitively understandable. To avoid confusion after speech input, an option would be to automatically stop the listening process automatically to avoid further involuntary input. To further improve the feedback on what is actually perceived by the *machine*, a good option would be to temporarily display the interim speech, which is provided by the API[4], which could display the text in front as it is spoken.

4.5 Final Prototype

The final prototype incorporates the changes made based on the user test findings. Through additional hand gestures with the left hand, the user is able to switch between Time-Based vs. Audio-Based playback and influence the playback speed. The impact of the user's behaviour is implemented to some degree, filtering and merging elements of the sonification. In addition, the user is able to change between thinking and non-thinking mode of the model by directly telling it to do so. Finally, the range of sonified movements of the hands has been extended.

Here I am providing a video with timestamps [9] , that document *Dialog* at the time of the thesis hand-in. This video or any follow-up actions can also be found on my website: www.jrfsound.ch/dialog

(9) Video Demo

Table 4.1: Video Timestamps of key events.

Time	Event
00:00	Hand and Finger Movements Initial
00:21	First Prompt (Non-Thinking) with Tokens
01:57	First Answer (Time-Based)
03:39	Interacting with Tokens (Navigating, Pointing Bells and Chords in Different Layers)
06:17	Reset View Gesture
06:33	Second Prompt (First One Fails)
06:55	Second Prompt (Non-Thinking, Working) with Tokens
08:17	Second Answer (Audio-Based)
08:38	Third Prompt (Non-Thinking) and Audio-Based Answer
10:37	Fourth Prompt (Thinking) and Tokens
11:29	Waiting for Response with Hand and Finger Movements (Now Filtered)
12:11	Fourth Answer (Audio-Based)
15:10	Program gradually dying because of too many tokens
16:01	Fifth Prompt (Non-Thinking) and audio-based answer
17:50	Final Explorations, Layer-Change Melodies, Pointing, Movements in Exchange

[4]See Speech to Text (STT) Plug-In (3.5.1)

5. Discussion and Reflection

This chapter first reflects on the Goals and Answers (5.1). In Reflection on the Sonification (5.2) I question musical decisions and sonification techniques. After the Reflection on the Process (5.3) I will state the limitations and give an outlook for possible further work and research in Limitations and Future Work (5.4).

5.1 Goals and Answers

Through this project, I was able to test and improve the existing pipeline to create plug-ins, from which seven made it to the final prototype of *Dialog*. There, I was able to use them creatively, enabling an exchange with an LLM on a sonic and textual level. The project uses state-of-the-art publicly accessible models, which can be replaced if desired with little effort [10] [1]. Furthermore, the used (VR)-Web technologies are going to be developed further, and their advantages of being easily accessible and platform independent are demonstrated here.

(10) Apertus vs Qwen

Overall, I am happy with the result and was pleased that a lot of the participants enjoyed their experience. The invitation and affordance for exploration showed through how the different personalities interacted with it, with some trying to analyse patterns and with others that were trying to make music with it.

However, it has not been confirmed whether the experience leads to a more *conscious* interaction with LLM, as I was focussing more on getting a functional prototype working and such an evaluation would have been beyond the scope.

[1] For example, I was able to test the newly released model *Apertus* from the Swiss AI Initiative (See https://huggingface.co/swiss-ai/Apertus-8B-Instruct-2509), released on 2 September and put it side-by-side with a Qwen model of the same size, that I used during development (See https://huggingface.co/Qwen/Qwen3-8B)

5.2 Reflection on the Sonification

In this section, I will look at the final sonification through the lens of the two different functions of sonification mentioned in Sonification (2.2): As an *artistic form* and for *data exploration.*

5.2.1 From an Artistic and Musical Perspective

The general sonic aesthetic of the machine emerged through iterative experimentation rather than a fixed concept from the beginning. This apart from the ground concept of a tonal bed and transient events. The sounds for the user arose from the way I describe my musical origin and expression with playing the cello since I was a child and my appreciation of the sound qualities of the koto that have followed me throughout the past decade. Combined with my fascination for bells and the often heard albums "Zen Koto" and "Zen Glocken" by composer and artist Michael Vetter, it is no coincidence that the project moved in a similar direction, especially when dealing with my questions about the conscious use of or the impact on artistic originality by LLMs.

The organ-like characteristic of the machine pad, combined with a long three-second reverb, evokes a church-like space. This highlights the bells from another perspective. Together with the slightly provocative double palm-up gesture in order to speak with the model, it navigates a semantic space of surrender and worship and asks the question of what role the LLM is given by us.

From these ideas, the sonification is kept modal for most elements. The exceptions are the bells that separate themselves from the dorian/mixolydian framing together with the cello improvisations from the hands. Additionally, the machine pad establishes through different chords different harmonic functions, but within the same tonal range. Although I am pleased with the overall sonic aesthetic that has been synthesised from the ground up, having a bachelor degree in composition, the lack of harmonic complexity or at least evolution leaves me somewhat dissatisfied in that regard. For example, evolution could have been established throughout the different states of interaction or throughout time.

While the first version of project, the *(modern)BERTophon* fulfils all the requirements of an instrument by being deterministic, (more or less) expressive, and learnable, *Dialog* now extends the idea to an immersive experience that sits somewhere between being an interactive composition and a multifaceted instrument that is played through conversation and interaction. The many interconnecting parameters make it hardly *learnable.* This complexity can be seen both as a strength and as a weakness, as the user hardly understands what is going on behind the curtains and has a difficult time creating a mental model.

The goal of creating a *personalised* composition, depending on the user behaviour, has only been somewhat achieved. The influence of behaviour is solely heard by different types of formant filtering and the amount of cross-synthesis of the voice. The adjustable speed of playback has an effect on the release time of the machine pad's envelope and, therefore, on its presence. In addition, the interpolation rate between the chords changes according to the playback speed, so the chords and *melodies* move faster. However, this can hardly be seen as a fully *personalised* composition. My argument against more compositional diversity during development was data interpretability, which leads to the next section.

5.2.2 From a Data Explorative Perspective

During my residency, I always shifted between creating a product that could be helpful for AI explainability, by being able to sonically explore and possibly understand the underlying data of a machine learning model that goes beyond a visualisation, or a more artistic project that allows people to interact with a model in a novel way. I eventually shifted towards the latter. However, there are still points that can be said about data exploration.

In *(modern)BERTophon*, the text material is provided completely by the user. It is therefore easy to reproduce the same conditions by inputting the same sentence and choosing the same layer. By optionally disabling the sentence pad, it is possible to focus only on the token embeddings or to focus only on the pad by changing through the sentences using the arrow keys. However, even there through the mapping technique, the sonic result can only be interpreted through similarity or emergent harmonic evolutions that most likely do not lead to a valuable new insight into the inner workings of the model. At least it is not a particularly efficient way, as the listener's recalling of different chords and timbral structures is limited.

In *Dialog* now, exploration has become more difficult. Although it is still the case that the same input leads to the same output, more factors come into play. The text is now generated by the user and the machine, and every token that is newly displayed depends on all the previous ones throughout the whole experience. It is still possible to replay tokens after the generation has finished, however, it is more cumbersome to reproduce the same conditions. This becomes further complicated by the user behaviour, which while not influencing the mapping of the sonification, affects its filtering, leading to a slightly different sonic impression.

The addition of a voice playback, while yielding better feedback on the currently treated text content and giving the model some personality, leads to masking of the sonifications and should be considered further.

After the design iteration the bells and machine pad have been decoupled for that the bell makes single tokens audible and the pad the averaged tendencies over the played-back tokens of the last few seconds. This is a better solution than to purely sonify the same data with two sonification approaches, but at the end its efficacy has not been evaluated. However, recurring harmonic and or melodic motifs can be picked up. For example, when listening to the words ”no emojis, no extra words”[2] under the following link [11] at timestamps 0:05 and 0:20. Although not identical, the underlying harmonic evolution is the same, suggesting similar data.

(11) Recurring Audio Motifs

For the user sonification, it is possible to hear the amount of total movement the user made in the past seconds, immediate movements of head and hand, and to some extent the jerkiness of hand movements. Movements of the fingers can be heard in combination with the direction in which the palms are generally facing. However, this is so subtle that nobody noticed it during the tests. For the interactions, it is possible to hear in what layer the user currently is and how fast the playback speed is set to. The only noticeable parameters driving the sonification were the velocity and its derivatives. In summary, the term user-sonification sets the bar too high for what I achieved in this direction, where I would expect a higher resolved acoustic representation of movement and behaviour.

5.3 Reflection on the Process

The nature of being able to rapidly prototype and preview DSP concepts in an editor like Max and having them minutes later available in an audio middleware like FMOD made it possible and simple to try out, whatever came to my mind. Unfortunately, becoming apparent in mid-May, a possible collaboration with an AI scientist that would have programmed a large part of the back-end and would have provided valuable insights was not possible. As a result, in doing a *solo* project, I spent a lot of time researching not only audio technologies and coding, but also LLMs, the different possible frameworks for VR development, and the other mentioned technologies, which led to less time spent researching artistic variety. This also led to not having implemented all planned elements at the time of the user tests and it would have been interesting to see how the user sonification is perceived. This suggests that the scope of the project in the given time was too ambitious.

I am grateful for having had the possibility to spend two months in the IVIA lab, which inspired me to many potential project ideas. One of which I was now able to follow through. However, I am not an expert in this field and needed help in navigating this complex domain. While the

[2]An often repeated mantra as it is contained in the system prompt. See System Prompt (A.5)

matter of the resulting product is a dialogue with an AI, the process of creating it itself was also one. Without it, I would not have achieved a project of this scope in this time frame and along the workload of other projects. Also, taking the *conscious* interaction out of the product and to the process of making it, this comes with pressing questions about originality, independence, authorship, and quality of work that I want to discuss and highlight here, as transparent as possible. Concrete examples of how and which models were used can be read in the Declaration of AI Use (7). To digress as little as possible, I exclude environmental considerations, although they are important to consider.

Originality: The overarching concept and goals were always in my hands. In regards to design, but also architecture. Instead of a single prompt that gave me the whole product, through countless iterations, I implemented new modules and their features one by one. Sometimes through AI, sometimes by me. However, this also does not mean that I invented everything.

For instance, the Risset Bell idea originally stemmed from both a Supercollider and a Sound Synthesis lesson about the bells themselves and sinusoids, respectively. After having had the spectrogram idea for *(modern)BERTophon*, I asked for a LINQ expression to get the top 20 indices inside Unity. In *Dialog*, this is done in the back-end using some Python wizardry. Both solutions I would have had to browse for in Stack Overflow and were probably forgotten by now.

For the machine pad, this is more complex. Although the use of FFT came from a researcher at IVIA, the idea for using spectral descriptors came from the collaboration with a chatbot. The mapping to pitch and voice grew from many different iterations, where in the end only the musical goal, the scales, and the use of intensity are clearly from my side, as this happened in FMOD, where I was on my own.

However, a chatbot has also been used for the plug-ins directly with aiding in Max for the bilinear interpolation logic of the formant filter in an RNBO codebox and in the search for a solution for the decay logic in the Karplus Strong algorithm, indirectly or directly affecting a sonic result.

The benefit is on the one hand to not having to learn every implementation nuance of the many existing, rapidly evolving libraries, but rather to focus and guide the general architecture. On the other hand, fine-grained control is given to an algorithm that misses the overarching picture.

Independence: If, for some reason, the availability of the models had ceased, I would have had to learn the existing codebase on this fine-grained level to be able to adapt for new changes. This would have slowed me to such an extent that I would have had to cut my goals shorter.

However, since some initial work was done, I would argue that I would have completed the entire project faster than if I had to research all the required steps from the ground up. Through my previous experience, I would have done a similar project, but focussed more on working with frameworks that I already knew. This would have resulted in a different project with a smaller scope.

Authorship: This point goes into the discussion of plagiarism in the use of generative AI, which is controversially discussed, especially in the arts. In my opinion, there is a gradient of authorship and originality, to its classification I do not have an answer. The fact is that through the use of an LLM, my codebase was populated by code, for which its generation the model was trained on data that were likely not intended to be trained on by their authors. However, through the context of my prompts, the curation of their results, and my own written code, this has become a new, unique code base.

Quality of work: The description unique for the code base fits, as it is in its current structure not a work of art, considering its coding conventions, modularity, and ultimately robustness. While I was guiding the structure along the way, by separating concerns and refactoring nonsensical and over-complex attempts from the AI, the final codebase became a mess nevertheless. Arguing against that, it is still better than the one from my bachelor project, which formed the beginning of my Unity programming journey. However, in a professional context that allows for less experimentation, I would spend more time either learning required libraries myself or monitoring the creation more closely, where here, I sometimes waived through new code additions without checking every detail. The wrappers for both RNBO and Heavy Pd are available on my GitHub page. In order to comfortably switch the source code for *Dialog* to a public repository, I will want to go over everything again before. If, at the time of reading, the repository [12] is not public, I am happy to answer questions through my contact details or grant private access.

(12) Dialog Repository

In conclusion, I think I managed to maintain the balance of control and freedom in the work with LLMs with respect to the result that can be experienced and noticed by the user. The lack of fine-grained control in the code base was in favour of accelerating development and elevating the result to a level that I could not have done without.

5.4 Limitations and Future Work

The project combines many fields. Traditional and new synthesis in game engines, sonification of large language models, sonification of movements, and the technicalities that allow all this to happen in a browser in virtual reality, combining it to a functional user experience. It provides an idea of what is and becomes possible with increasingly powerful hardware, accessible frameworks, and tools. In creating the experience with all its components, combined with the endless possibilities I had from the plug-in pipeline, the depth of the single elements is limited and only scratches the surface of what can be done. This thesis only makes a first step towards the sonification of LLMs and how users interact with it. Almost every aspect could be explored extensively in isolation and in more detail.

For instance, taking the approach of representing the embeddings through bells could be taken as its own research. A model-based sonification concept that uses physical modelling with many more parameters could be tested. The bell could be excited, by hitting it or bowing it at different spots, making the embedding explorable in a more profound way. In addition, the use of simply the embeddings could be extended to the multi-head attention mechanism, maybe a whole bell tower, comparing different heads sonic semantics or word families. Such attempts may also provide more insight for the field of explainability, as this is very limited here.

While the used model is one of the most topical, it would also be interesting to explore MoE models, possibly sonifying active experts or the data that are responsible in choosing them. Furthermore, implementing the MCP protocol, the model could control or access info from Unity directly, enabling natural interactions with the environment or elements in the scene. Finally, switching to a completely different model architecture and exploring, for instance, a generative audio model through audio could potentially be very interesting, especially when its output would also be combined with the corresponding sonification.

The sonification of movement uses primitive movement data such as joint velocity, acceleration, and rotation of head and hands, making a first step towards movement sonification. Not only could the sonic palette and variety of sonified movements have a higher number and resolution, spectral descriptors for segments of movement or the use of established concepts such as Laban Movement Analysis (LMA) could improve the capture of the intention of movement of the user (Bouchard & Badler, 2007, p. 4).

The single-thread runtime on the Web currently makes tracking less reactive than a native build. It would also be possible to further extend the tracking to other forms, such as the addition of IMUs that could be directly connected to the headset for increased precision and reactivity.

Ultimately, the concept of user behaviour is tracked here through the time spent in different states, which only gives a crude estimate. A chapter could be opened on user focus or engagement measurement and the control or influence of sound on those.

6. Conclusion

The question of how to utilise the potential of custom plug-ins in game engines allowed me to participate at the residency at IVIA. Inspired by their work, I built a first prototype that led me to build *Dialog*, an immersive, sonic interaction with a Large Language Model.

The thesis produces two outcomes. First, the plug-in pipeline has been improved and, through its more accessible custom DSP making process, can potentially enable sound designers to shape more detailed and responsive audio for complex, interactive systems. *Dialog* itself shows this in an approach to sonify the inner workings of an LLM and how it is interacted with. It sets a foundation on which the described methods and tools could be expanded in future work. For example, further user tests could investigate how users interpret or react to the elements of *Dialog*, exploring and improving the application itself. Or, the sonification of LLMs, including the attention mechanism or any other neural network at a more granular level, could lead to different or greater insights there.

List of Figures

Icon References

Clock Figure 4.3 and wave Figure 4.4 Icon by https://icons8.com/.
The arrow used in all interactions comes from https://www.flaticon.com/

Bibliography

Bouchard, D., & Badler, N. (2007). Semantic segmentation of motion capture using laban movement analysis. *Proceedings of the 7th international conference on Intelligent Virtual Agents*, *4722*, 37–44. https://doi.org/10.1007/978-3-540-74997-4_4

Bradley, M. M., & Lang, P. J. (1994). Measuring emotion: The self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry*, *25*(1), 49–59. https://doi.org/10.1016/0005-7916(94)90063-9

Cho, A., Kim, G. C., Karpekov, A., Helbling, A., Wang, Z. J., Lee, S., Hoover, B., & Chau, D. H. (2024). Transformer explainer: Interactive learning of text-generative models. https://arxiv.org/abs/2408.04619

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805

Dombois, F. (2002). Auditory seismology – on free oscillations, focal mechanisms, explosions, and synthetic seismograms. *Proceedings of the 8th International Conference on Auditory Display*, 27–30.

Dombois, F., & Eckel, G. (2011). Audification. In T. Hermann, A. Hunt, & J. G. Neuhoff (Eds.), *The sonification handbook* (pp. 301–324). Logos Publishing House. http://sonification.de/handbook/chapters/chapter12/

Grond, F., & Berger, J. (2011). Parameter mapping sonification. In T. Hermann, A. Hunt, & J. G. Neuhoff (Eds.), *The sonification handbook* (pp. 363–397). Logos Publishing House. http://sonification.de/handbook/chapters/chapter15/

Hermann, T. (2008). Taxonomy and definitions for sonification and auditory display. *Proceedings of the 14th International Conference on Auditory Display*.

Hermann, T. (2011). Model-based sonification. In T. Hermann, A. Hunt, & J. G. Neuhoff (Eds.), *The sonification handbook* (pp. 399–427). Logos Publishing House. http://sonification.de/handbook/chapters/chapter16/

Hermann, T., Hunt, A., & Neuhoff, J. G. (2011). Introduction. In T. Hermann, A. Hunt, & J. G. Neuhoff (Eds.), *The sonification handbook* (pp. 1–6). Logos Publishing House. http://sonification.de/handbook/chapters/chapter1/

Morales, E., James, K., Horst, R., Takeda, Y., & Yung, E. (2021). The Sound of Our Words: Singling, a Textual Sonification Software. *Proceedings of the 26th International Conference on Auditory Display (ICAD 2021)*, 164–168. https://doi.org/10.21785/icad2021.038

Sevastjanova, R., Cakmak, E., Ravfogel, S., Cotterell, R., & El-Assady, M. (2023). Visual Comparison of Language Model Adaptation. *IEEE Trans. Vis. Comput. Graph.*, *29*(1), 1178–1188. https://doi.org/10.1109/TVCG.2022.3209458

Torre, G., Andersen, K., & Baldé, F. (2016). The Hands: The Making of a Digital Musical Instrument. *Computer Music Journal*, *40*(2), 22–34. https://doi.org/10.1162/COMJ_a_00356

Walker, B. N., & Nees, M. A. (2011). Theory of sonification. In T. Hermann, A. Hunt, & J. G. Neuhoff (Eds.), *The sonification handbook* (pp. 9–39). Logos Publishing House. http://sonification.de/handbook/chapters/chapter2/

Online Resources

20kHz Podcast. (2025). *Solar symphony*. Retrieved August 13, 2025, from https://www.20k.org/episodes/solar-symphony

IRCAM. (2025). *Sinusoidal analysis and synthesis*. Retrieved August 11, 2025, from https://support.ircam.fr/docs/AudioSculpt/3.0/co/General%20Principles.html

Kyriakides, Y. (2024). *Hands*. Retrieved August 22, 2025, from https://www.kyriakides.com/hands.html

Minimi Dance Theatre. (2023). *Biodata sonata blog*. Retrieved August 16, 2025, from https://minimi.fi/en_GB/biodata-sonata-blog

Mordvintsev, A. (2022, December 23). *Particle Lenia and the energy-based formulation*. Retrieved August 13, 2025, from https://google-research.github.io/self-organising-systems/particle-lenia/

Mozilla Developer Network. (2025). *Web speech api*. Retrieved August 7, 2025, from https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

OpenAI. (2025, August 7). *Introducing gpt-5*. Retrieved August 14, 2025, from https://openai.com/index/introducing-gpt-5/

Piché, J., & Nix, P. J. (1997). *Table iii: Formant values*. University of Chicago. Retrieved April 6, 2025, from https://www.classes.cs.uchicago.edu/archive/1999/spring/CS295/Computing_Resources/Csound/CsManual3.48b1.HTML/Appendices/table3.html

Raschka, S. (2025, February 5). *Understanding reasoning llms*. Retrieved August 14, 2025, from https://magazine.sebastianraschka.com/p/understanding-reasoning-llms

Schmitt, P. (2020). *Mark ii (convolutional neural network)*. Retrieved August 14, 2025, from https://philippschmitt.com/work/mark-ii-convolutional-neural-network

The MathWorks, Inc. (2025). *Hyperbolic tangent - matlab tanh*. The MathWorks, Inc. Retrieved August 2, 2025, from https://www.mathworks.com/help/matlab/ref/double.tanh.html

Video Resources

Alexander, R. (2024, December 9). *Listen to space! the parker solar probe flies through venus's magnetic field.* [Video]. Retrieved August 3, 2025, from https://www.youtube.com/watch?v=pdB-z0K1foo

Hutchinson, S. (2024, June 24). *Artificial neurons for music and sound design* [Video]. Retrieved August 15, 2025, from https://www.youtube.com/watch?v=jNcj-XQwIFg

7. Declaration of AI Use

...in Writing the Thesis

Use for Formulation and Orthography: As my native language is not English and to get advice on formulation, Overleaf's **Writefull** language suggestions were used, using their own Writefull model. Its *supercomplete* or any other generative features were not used.

Use for Feedback in Structuring: Furthermore, **Gemini 2.5-pro** was used iteratively on the website to structure the Table of Contents (ToC) to provide a stronger scaffold to write on. This was done by copying and pasting the ToC into the chat and approving and editing my list accordingly or rejecting the advice listed by the AI.

Chapters with which I was not happy, I copied alongside my intentions for the section and asked for feedback. Statements from AI were considered and resulted in my own rewriting or were rejected. No generated text was copied, nor suggestions paraphrased.

Finally, in the same setting, the model was also used for latex-specific formatting. The code for title formatting was directly copied or adapted and used in the editor.

Use for Learning Concepts: **Gemini 2.5-pro** was also used to explain concepts about LLMs and the Transformer structure. Those were either then used in the project, verifying its correctness or backed by sources, that are cited.

Translation: Finally, **DeepL Translate** was used for single-word translations.

...in Creating the Experience

The IDE **Cursor.ai** was used in conjunction with the following models, ordered by frequency of use: **Gemini 2.5-pro**, **Claude 3.5-sonnet**, **Claude 4.0-sonnet**, **gpt-5** and **gpt-4**

The application has evolved substantially during the project, allowing for agentic interaction on the code base in supervised creation and deletion of files. Each edit from the AI is presented as a visual diff, highlighting code changes that can be accepted or rejected section by section, for the whole file or all files together. This has been extensively used, by validating the written logic and suggesting changes for separation of concerns, readability, and maintainability. Furthermore, AI is able to write its own test queries, running the code itself, and iteratively hunting bugs in the code.

However, the whole project spans 85 separate chats, which followed an even larger amount of subgoals that I wanted to achieve and design throughout.

Most of the audio implementations I have done myself using **JetBrains Rider** or started them and iterated over it with AI, where I would normally have had to consult online resources. (For instance, tracked instance pooling for the layer change.) In Rider I additionally used **Codeium**, that rebranded to **Windsurf** for line completion.

A. Additional Technical Descriptions

A.1 Platform Independent Developing

Through the residency, the use of using a Web-Build became more and more interesting, as the final product can be easily shared with anyone using a simple link that can be opened with almost any platform that supports WebGL, which is the majority of modern browsers nowadays. Furthermore, WebXR[1] and additionally the WebXR exporter for Unity by de-panther[2], is actively being developed on, which is open source and allows creating immersive, and most importantly platform-independent Web content. Looking in the near future, it is highly likely that WebGPU will be integrated into WebXR, which will expand the performance and processing capabilities that are quite limited compared to native builds. Testing a build with hand tracking on an Apple Vision Pro just worked easily out of the box, which led to the decision to use this approach.

However, using Unity's OpenXR API, it is comparatively easy to export the experience to various platforms.

The choice to aim for a Web-build still influenced the development by having to write a speech-to-text browser plug-in for Unity, the solution for spatial audio for the web, and a way to live-tune sonic or visual parameters in the final build.

A.2 Tools

Different tools were created to facilitate development.

A.2.1 Plugin Compiler Offline Pipeline

To simplify the management and updating of a large number of plug-ins, I made a cross-platform offline batch compiler using Python[3], which compiled for all available architectures and copies the build results into the respective folders. This offers a speed increase over the online pipeline that uses single runner instances from GitHub and requires manual downloading and moving afterwards.

Ideally, I would like to develop a standalone hybrid application similar to my first GUI compiler attempt using JUCE[4] that, however, connects to the online database. This would make development easier and faster.

A.2.2 Live Tuning Panel

Building the web target in Unity can take anywhere from a few seconds to many minutes and through the development on a mac it was not possible to do a live preview on a headset. I

[1]See https://immersiveweb.dev/
[2]See https://github.com/De-Panther/unity-webxr-export
[3]Through the use of Gemini 2.5-pro in Cursor
[4]See https://github.com/JFuellem/RNBO-FMOD-Compiler

had to find another way to manipulate variables while running the game to fine-tune or adjust parameters. I built an implementation in which a field in a Unity script could be given the attribute [Remote Editable] or [Remote Visible], which made it editable/visible in a remote tuning utility using reflection and an experimental web socket setup. Without going into more details, this was, for instance, essential when trying to determine how reactive a parameter has to be set to achieve a desired effect.

A.3 RNBO - Memory Leaks, and How to Find Them

A significant challenge was during testing the web build, where memory is more limited. The application started to crash after some time and the last memory allocation was always made from the RNBO initialisation method, which allocated a required signal buffer. This method is fired every time an event with a plugin plays, which can be quite often, as this happens for every token playback. When reproducing it with a test case, the application successfully crashed after firing around 10'000 plug-in instances. I finally was able to isolate it to being RNBO (v1.3.4), which did not free some of the objects that are allocated.

In the above-mentioned Offline Pipeline, this is corrected automatically by altering the source code to free the objects in the destructor of the top-level patcher class.

A.4 Inference in *Dialog*

For *Dialog*, I used the Huggingface Transformers[5] framework, which has become an industry standard for working with transformer models. To retrieve all the embeddings and attentions, the model is inferred manually instead of using the standard model.generate() pipeline provided, where such an extraction is not possible. This means that the model is inferred token by token, calculating the next token *by hand* and storing past key-value pairs manually. This is magnitudes slower and requires more memory, but to my knowledge is necessary here. To maximise reproducibility, calculating the next token is done using torch.argmax on the logits, picking the most probable next token. However, a temperature approach was also implemented, which picks the next token based on its probability using torch.multinominal. The temperature can be specified for each prompt request.

Through the use of the Transformers library, it is quite simple to switch to different models. However, the output of the models can still be quite different, which means that the stored normalisation parameters have to be recalculated. For this project, the Qwen model from Alibaba is used. First, with Qwen 1.5[6], and later with Qwen 3.0[7], which was the reasoning model that performed the best during development. The exact models used at the end are "Qwen/Qwen3-4B", "Qwen/Qwen3-4B" if only few prompts are made or with one client and "Qwen/Qwen3-0.6B" with a smaller footprint[8], released in April 2025.

[5]See https://huggingface.co/docs/transformers/en/index
[6]See https://qwenlm.github.io/blog/qwen1.5/
[7]See https://qwenlm.github.io/blog/qwen3/
[8]See same blog entry as above

A.5 System Prompt

A system prompt is a set of words that are initially provided to the model. Located at the beginning, they are valued more than later tokens and can be used to give instructions on how the model should *behave* or what to say and not say. During reasoning, those instructions are often picked up and result in a kind of mantra, where the model repeats its instructions. The system prompts of current chatbots like Claude often contain thousands of tokens that explain how the model should respond or what to omit. A prompt example for the Claude models can be seen on the Anthropic website, as they make those publicly available[9].

Theoretically, this can be used for storytelling or other creative explorations. However, I was not interested in trying to stage the model beyond the generic assistant role.

For *Dialog*, a generic prompt was used, which was extended during development and reflects the problems that I had such as the length of output or the use of emojis.

You are a helpful assistant with the following characteristics:

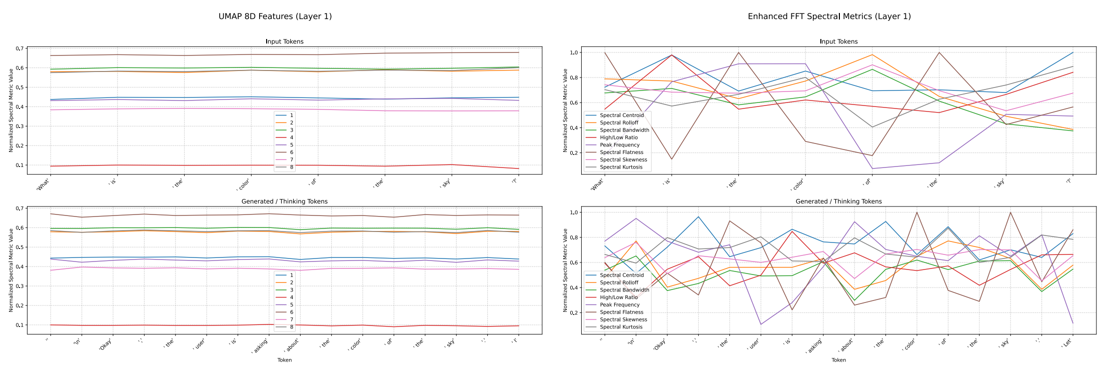
1. Please keep your reasoning short.
2. Keep your responses concise and to the point.
3. Avoid using emojis in your responses.
4. You can be casual and friendly, but don't be too verbose.
5. Focus on providing accurate and relevant information.
6. Break down complex topics into simple explanations.

[9]See https://docs.anthropic.com/en/release-notes/system-prompts

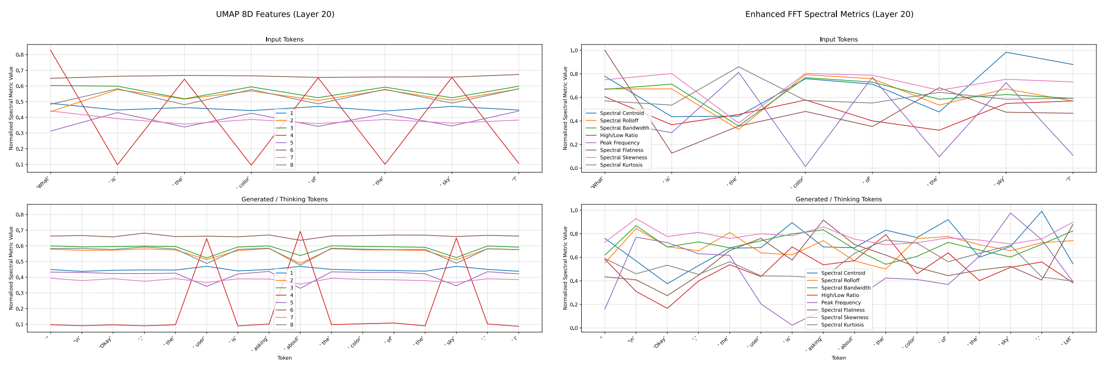
A.6 UMAP vs. Spectral Descriptors

The spectral descriptors used are experimental and are an unusual way to do what feature projection algorithms are meant for. However, I favoured it over the use of a more traditional, more precise approach like UMAP because it results in an increased variety of chords. It would also have been possible to tweak the feature projection parameters or use another method such as t-SNE. However, I liked the results of the chords and the use of an approach that is more rooted in the audio domain.

Followed are three comparisons of the resulting values in different layers, with UMAP on the left and the Descriptors on the right that were assigned to pitch and intensity.

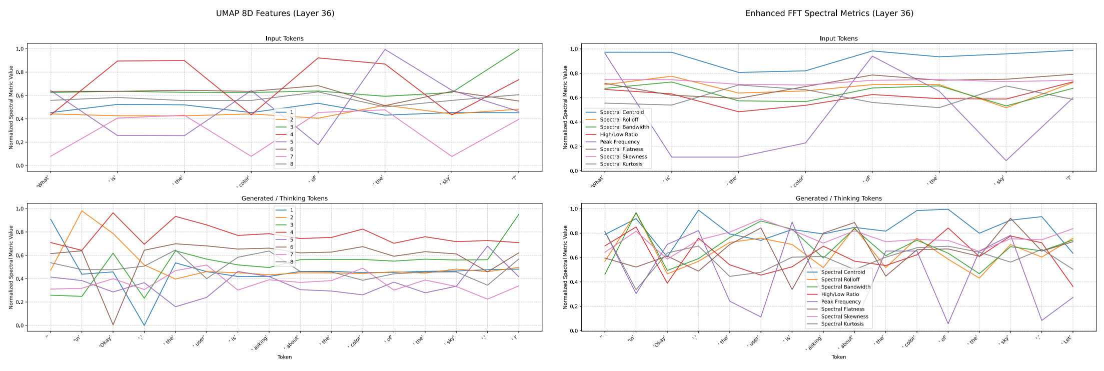


(a) Layer 1



(b) Layer 20

(c) Interestingly enough, some shapes of features are preserved, for example, looking at the input tokens (top row), comparing UMAP value 4 with *peak frequency*.



(d) Layer 36

Figure A.1: UMAP vs. Spectral Descriptors at different layers

Declaration of Autonomy Master-Thesis

I hereby confirm that I have written this Master’s thesis entitled "Dialog - A sonic encounter with a large language model in VR" independently and without outside help.

Any content taken from other sources such as texts, images, audio, graphics, software, etc., whether verbatim or analogous, is correctly cited with full attribution of authorship and source. In addition, all passages created with the help of AI-supported programs are clearly marked and provided with the exact name of the program used and the prompt applied. It is declared how AI tools were used for the translation of my own text, idea generation, brainstorming or similar.

Furthermore, I confirm that the thesis has not yet been published and has not been submitted in an identical or similar form as an examination or final project at another university, educational institution or in another degree program.

I acknowledge that a violation of these requirements may have legal and disciplinary consequences in accordance with § 26 Regulatory Framework for Bachelor’s and Master’s Degree Programmes at Zurich University of the Arts in conjunction with §§ 8 ff. Ordinance of the Universities of Applied Sciences Act.

With my signature I confirm the accuracy of this information:

First name: Jonas   Last name: Füllemann

Swiss matriculation number: 18-874-826

Date: 25.08.2025

Signature: [illegible]