In cooperation with

MASTER THESIS

Tone Tutor: A virtual teacher for music theory

Presented on 29. November 2024

Rodrigo Silvetti Murillo

Erstprüferin: Prof. Dr. Eva Wilk
Zweitprüfer: Yvan Grabit

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

Dieses Projekt befasst sich mit der Herausforderung, Menschen zu motivieren, in ihrer Freizeit etwas so schwieriges wie Musiktheorie zu lernen, indem Gamification-Prinzipien und immersive Geräte wie die Meta Quest 3 genutzt werden. Ein Spiel wurde entwickelt, um Musiktheorie zu lehren und den Nutzern das Spielen auf einem MIDI-Controller zu ermöglichen, wobei Werkzeuge wie Unreal Engine und Cubase verwendet wurden, um das Engagement zu erhöhen.

Das Spiel verfügt über einen virtuellen Lehrer, dargestellt durch einen Roboter „Tone Tutor“, in einer Mixed-Reality-Umgebung. Gamification-Elemente, einschließlich einer Erzählung und eines Punktesystems, schaffen herausfordernde Aufgaben. Der Roboter liefert alle Unterrichtsinhalte und ersetzt traditionelle schriftliche Tutorials. Das Spiel bietet zwei verschiedene Spiele, um das Nutzerengagement zu steigern.

Trotz der Komplexität des Setups ist das Spiel angenehm zu spielen, und die Integration der Komponenten war erfolgreich. Wertvolles Feedback wurde für zukünftige Verbesserungen gesammelt. Dieses Projekt stellt einen vielversprechenden Ansatz zur Musiktheorie-Ausbildung durch Gamification dar und bietet eine solide Grundlage für zukünftige Versionen.

Abstract

This project addresses the challenge of motivating individuals to learn something as difficult as music theory in their free time by using gamification principles and immersive devices like the Meta Quest 3. A game was developed to teach music theory and allow users to play on a MIDI controller, utilizing tools such as Unreal Engine and Cubase to enhance engagement.

The game features a virtual teacher, represented by a robot “Tone Tutor”, in a mixed reality environment. Gamification elements, including a narrative and a points system, create engaging challenges. The robot delivers all instructional content, replacing traditional written tutorials, and the game offers two different games to boost user engagement.

Despite the complexity of the setup, the game is comfortable to play, and the integration of components was successful. Valuable feedback was collected for future improvements. This project presents a promising approach to music theory education through gamification, providing a solid foundation for future versions.

Table of Contents

List of Abbreviations

DAW	Digital Audio Workstation
OSC	Open Sound Control
VR	Virtual Reality
AR	Augmented Reality
MIDI	Musical Instrument Digital Interface
XR	Extended Reality
SKI	Steinberg Kernel Interface

List of Figures

1 Introduction

After working or studying for many hours a day, everyone feels tired and wants to disconnect from their daily activities by playing something and relaxing. Imagine being able to learn something effortlessly while playing. Education can sometimes be a challenging and frustrating journey, especially when dealing with complex or abstract topics. In addition, depending on the subject, education can be very expensive. Often a person will begin to learn something with interest, but during the process they may find that the subject is not what they expected, causing them to drop out or change subjects, wasting money in the process.

Music theory, with its extensive content, is a significant challenge for many. It requires the memorization of a vast amount of information and can often be tedious. The implementation of gamification offers a potential solution to the problem of motivational deficits in learning, especially for tasks that are repetitive and require extensive theoretical reading.

An affordable solution is a program that costs less than attending an institute, but it can still be difficult to study alone at home. If technology can help reduce the cost, it could also address the problem of boredom in learning.

One solution is to approach the problem in a game-like way, known as gamification, by incorporating elements such as points to be earned or a narrative to develop. This can increase enjoyment, excitement, and engagement in the process. Keeping the user's motivation high as they continue to learn music can be a positive stimulus to maintain interest in music. Ultimately, this could lead to joining a music institute for professional training if the learner wishes to pursue a music career. This solution could help reduce initial expenses and increase motivation for the subject.

Emerging mixed reality technologies have the potential to revolutionize personal engagement in everyday tasks, particularly in education. Similar to how computers and the Internet have transformed education by introducing a new dynamic, these novel virtual devices could further enrich the learning experience. They allow users to engage with a variety of virtual tools and navigate through virtual environments. Immersive technologies such as virtual reality (VR) and augmented reality (AR) provide opportunities for interaction, compensate for real-life limitations, extend reality, and even provide cognitive training and promote brain activity. Mixed Reality combines both VR and AR, understanding the real environment and adding 3D models to it, making it possible to interact with the world. These new devices provide a high level of immersion, but sometimes cause discomfort when used. Therefore, tests are needed to determine how uncomfortable they are.

One objective of this thesis is to propose an interactive approach to learning music theory that enhances engagement. Instead of overwhelming learners with exhaustive details and hours of reading in music books, it provides them with a tool to learn music theory in a more entertaining way. This approach seeks to increase participants' interest in learning theoretical concepts and musical notation. To achieve

this, a 3D character named "Tone Tutor," a virtual teacher of music theory, will be created. This robot will explain all the concepts, making the learning experience more enjoyable.

This project will utilize mixed reality to facilitate the learning of musical notation and rhythm. Users will engage in a game designed to teach the fundamentals of music theory and enhance their ability to follow the beat of a simple melody. The game will also offer finger dexterity exercises and provide insights into understanding basic major and minor scales, as well as forming chords on a keyboard. All the necessary tools for users to engage in short, simple improvisations will be supplied. Users will have the capability to play with a backing track, simulating a virtual jam session.

To achieve this objective, it will be necessary to utilize a mixed reality headset, specifically the Meta Quest 3, a physical MIDI controller, and a digital audio workstation (DAW) such as Cubase to support all sound-related operations. This study could benefit existing Cubase users with little or no musical background by providing an opportunity to learn within a familiar environment, utilizing some of the same instruments and effects tools.

The project will focus on functionality, gaining insights about player engagement, their learning progress, and how the project can be improved for future versions. Feedback on the best resources presented and necessary changes will be collected. Additionally, assessing player discomfort while using these emerging devices is a priority. All results must be documented to inform future versions.

Since this project needs to be finished in six months, the music lessons will be kept short. Additionally, as the project is not being undertaken by a professional music teacher, the focus will be on fundamental concepts to make sure it can be completed on time.

1.1 Music Theory

The field of music theory encompasses a vast array of definitions, extending from the historical development of music to the physics of sound frequencies. Given the complex nature of this subject, misunderstandings can arise, making it necessary to clarify what is meant by music theory. The goal of this project is to provide an overview of the fundamental principles underlying compositional techniques for a basic understanding of music creation.

This project focuses on a simplified theoretical music course in Western music. By limiting the amount of theory covered, it will be possible to avoid an overly large project that could exceed six months of development. The course will provide simple explanations of tones and semitones, using the 12-note system and its representation on a keyboard. It will also include elementary descriptions of major and minor scales, the construction of basic major and minor chords using the first, third, and fifth positions of a scale, and the various chords within a scale. A detailed written explanation that was used in this game is included in the appendix of this document.

2 Problem Analysis

2.1 Problem Identification

Music theory can be highly demanding, complex, and often frustrating. For individuals who do not acquire this knowledge during childhood, mastering it later in life can be particularly challenging. While playing an instrument is a cherished hobby for many, the theoretical foundation behind each piece and the ability to improvise remain significant hurdles. Nonetheless, the aspiration to learn persists. In the modern era, this challenge can be addressed through the integration of advanced technologies and the development of innovative educational methodologies.

2.2 Problem Structuring

To effectively address the challenges of learning music theory, it is essential to understand the current landscape and identify problems in existing solutions. This section will explore the following aspects:

Current Methods of Learning Music Theory: People typically learn music theory in traditional classroom settings, through private tutoring, using paper or books, and by accessing online courses and video tutorials.

Limitations of Existing Methods: When learning music theory through self-education at home, rather than in an institution, learners can experience a lack of interactivity and engagement, making progress discontinuous. Maintaining motivation and interest over time is challenging. Positive feedback helps keep learners motivated, while negative feedback corrects mistakes. However, the vast amount of information available can overwhelm users, making it difficult to find a clear learning path and leading to misinformation.

Technological Advancements: New technologies, such as mixed reality devices, offer significant advantages by increasing user immersion. Integrating these technologies into education is essential. Implementing MIDI controllers and Digital Audio Workstations (DAWs) in the learning process can make the experience more interactive.

Identifying Opportunities: New pedagogical methods, such as gamification, are more engaging and interactive. Advanced technologies help create immersive learning environments. Combining these with the implementation of DAWs and MIDI controllers presents an opportunity to create an educational game that makes learning music theory more enjoyable by combining fun with education.

2.3 State of the Art

Gamification Music Theory

Empirical studies in the field of gamification for educational purposes have demonstrated that it is an effective method for engaging learners and motivating them to learn. For example, the investigation

conducted as part of the Melody Mystery project (Lim et al., 2023) involved the creation of a music theory learning game that employed a puzzle-based approach. The objective of this approach was to enhance learners' motivation and engagement. The study's findings indicated that participants had acquired a basic understanding of musical concepts and achieved the learning objectives with positive responses to music theory. The results suggest that integrating gamification into educational environments can effectively enhance learner engagement and motivation. The study was subject to certain limitations, including the use of a virtual keyboard rather than a physical piano keyboard, which may be a subject for further investigation. Furthermore, technological constraints in learning applications prevent them from accounting for individual circumstances, such as varying learning speeds or attitudes towards technology.

The study “Gamified Learning Intervention to Promote Music Literacy and Creativity in Elementary Music Education” (Robert et al., 2023) highlights the decline in research on theoretical music and proposes a study that employs both online and face-to-face platforms. It acknowledges the potential of gamification in music lessons to enhance musical skills. The study employed the principles of gamification in board games to teach the principles of music theory. The study demonstrated the potential of board games in music education. The game was found to be enjoyable and engaging for learners in developing rhythmic sight-reading skills. The study concluded that integrating game design principles into the learning process could facilitate the development of valuable qualities in learners in the modern era.

It is a reasonable conclusion that integrating gamification for educational purposes effectively sustains student motivation. However, the prospective integration of augmented reality into these initiatives requires additional exploration.

Mixed Reality to Learn Piano

The project HoloKeys (Hackl & Anthes, 2017) developed with a head-mounted display (HMD) has been utilized for piano lessons and provides a comprehensive illustration of the potential outcomes of a project of this nature. It introduced a method for musical learning, a simulation of the game Beatmania (Deep Clear Eyes, 2017) from Konami, a rhythm game where participants press buttons in sync with approaching blocks, designed to help users acquire musical rhythm skills in a gamified manner. It presents compelling arguments for further research. Although not a commercial product, this project has demonstrated success. The study outlines various potential uses, such as teaching piano notes and as well as assisting in the learning of simple to intermediate musical compositions. Additionally, the method could be employed to enhance technical proficiency when playing the keyboard or to gain insight into chord structures, scales, and other musical concepts. A few considerations should be made regarding the project. These include the potential use of music sheets as markers and the monitoring of

learning performance. This concept could be integrated into the actual project to facilitate the learning of basic scales and chords through the use of music sheets.

Researchers at the Alexandru Ioan Cuza University of Iasi developed an application “An Augmented Reality Piano Learning Tool” in AR (Alexandru Ioan Cuza University of Iasi et al., 2021) to facilitate piano learning. Study participants demonstrated a high level of interest in utilizing the app for piano instruction. Based on these observations, the following recommendations have been made: the calibration of the virtual keyboard with the physical keyboard should be as straightforward as possible, and the option to alter the position after detection should be available. It is crucial to consider the potential impact of virtual reality on individuals who may experience dizziness. Some participants paid more attention to their hands on the keyboard than others, suggesting the need to display the hands at all times during execution. It would be beneficial to provide feedback indicating whether the correct note was played. A score system could be implemented to allow users to track their progress and improve their performance with each repetition. It would be preferable if the melody ceased until the user successfully plays the correct note. This could be incorporated into a training module, with the inclusion of a pause button to allow users to analyze the next note, adjust the tempo, or simply take a break due to dizziness. A user could engage in piano practice by focusing on the notes, with the keys of the piano subsequently illuminated in a different color. Nevertheless, the software does not provide instructions on how to read sheet music or identify the notes on the keyboard.

Furthermore, the paper Chord AR (Lu et al., 2022) describes the development of an app designed to facilitate the learning of piano skills in children. The efficacy of the app in engaging children in the learning process has been demonstrated. The researchers developed an augmented reality project with the objective of providing immersive experience in the learning of music theory for preschool children. The objective of this project was to teach fundamental chord knowledge. Given that the app was designed with children in mind, it was essential to incorporate elements that would encourage engagement. As a result, it significantly boosted their motivation to learn basic music theory and ensured they understood the content without losing interest. This paper indicates that the creation of multiple difficulty levels is beneficial for enriching interaction and maintaining enthusiasm. Additionally, providing guidance and dynamic musical effects is vital for personalized learning and stimulating curiosity.

The research presented in the International Journal of Instruction with the title “Adoption of Virtual Reality Technology in Learning Elementary of Music Theory to Enhance the Learning Outcomes of Students with Disabilities” (Maqableh et al., 2024) offers an alternative approach to teaching music to students. The study employs virtual reality to assess the differences in attitudes between students with and without disabilities as they learn the principles of music theory. The researchers postulate that the advantages of virtual reality will continue to expand as further research is conducted and implemented in the learning of music theory. The study indicates that virtual reality technology is beneficial and engaging in the learning process. It concludes with successful results, demonstrating that there are no

adverse effects on the health of students and the ability to achieve learning objectives. This study shows that students with disabilities have a greater intention to utilize this technology, which facilitates their overcoming of the physical disabilities they endure. It is notable that students with disabilities and their non-disabled counterparts exhibit a comparable acceptance of VR in music education. A larger-scale study could help identify additional tools to improve learning. The project provides numerous insights and recommendations, emphasizing that a simple method of interaction with virtual or augmented environments is crucial, as visually appealing 3D models can greatly increase user enjoyment.

DAW as a Learning Tool

The use of a digital audio workstation (DAW) allows music to be visualized and modified with visual feedback. As Mark Marrington notes in his project ”Experiencing musical composition in the DAW: the software interface as mediator of the musical idea” (Marrington, 2010), DAWs have transformed music production and the way artists conceptualize music, encouraging creativity among artists who can view DAWs as a virtual sketchpad for experimenting with new concepts and ideas, free from the constraints of traditional compositional methods. The looping feature is a particularly convenient tool that is widely used in DAWs. The idea is to use the computer as a musical instrument. The creative freedom afforded by this tool is significant. It also increases students' engagement with music. In conclusion, it is an essential tool that could be used in this project to cultivate creativity while learning music.

2.4 Goal Formulation

Recognizing that music theory can be extensive, this project will provide a concise version that covers essential concepts. There is potential to expand into a larger product involving several additional lessons and game levels, taking into consideration the results obtained from this analysis. It is crucial to assess whether the project is progressing correctly while ensuring that it remains user-friendly and engages users to keep playing this game.

Goals in the project**:**

1. **Functionality:** The proposed game should run smoothly, ensuring a seamless user experience. The technical aspect of the project needs to ensure the correct functionality and connection between the mixed reality glasses, the MIDI controller, the DAW, and the game, keeping players entertained.
2. **Learning Outcomes:** Players should be able to identify the C key on the keyboard or the C scale after one try. Although this aspect is hard to determine with just one use. It is necessary to determine if players learn something from the game.
3. **User Engagement:** Users should show interest in the game, continuing to play and learn music theory. The game should be entertaining, encouraging players to continue using it in the future

while maintaining their interest in music theory, which is crucial for engagement with the learning process.

4. **Feedback:** Another important goal is to gather feedback that helps gain insights, uncover trends, and generate ideas for future developments. Participants should be asked to evaluate the proposed ideas and provide feedback. The evaluation of this project can identify user needs, potential design flaws, and market gaps, ultimately guiding the development of a product that meets customer expectations.

3 Design a Mixed Reality Game

3.1 Concept Development

This study will focus on four main concepts: enjoyment, motivation, excitement, and emotions or feedback about the game.

The research conducted in the section before "State of the Art" has allowed us to explore three important concepts that could enhance music theory learning. These concepts aim to increase engagement during the learning process. Here are some key points for each of these ideas:

1. **Gamification:** Gamification is presented as an engaging method for learning music theory. By incorporating game elements such as challenges, rewards, and interactive exercises, learners become more motivated in their learning.
2. **Mixed Reality (Augmented and Virtual Reality):** Mixed reality applications can further enhance the learning experience by providing immersive and interactive content.
3. **Digital Audio Workstations (DAW):** DAWs allow users to experiment with different sounds, compose music, and explore various musical arrangements. The freedom to choose from a wide range of sounds and instruments makes DAWs both enjoyable and educational. Integrating DAWs into music theory education can increase creativity and engagement.

3.2 Idea Selection

Objective: To create a functional game for learning that keeps users engaged, several ideas have been developed, such as:

- **Block Falling Game:** In this game, players see blocks approaching a virtual keyboard while a song plays in the background. The player must press the corresponding keys, which the application recognizes. While this can help users learn a song mechanically, a significant drawback is that after playing, users often struggle to recognize notes on a physical keyboard and cannot play without the application. Additionally, using only a virtual keyboard may hinder the player's sense of velocity with the keys, making it difficult to control the song's volume. Consequently, players may learn to play the instrument but not understand music theory.

- **Escape Room:** This concept simulates a room where users must escape by solving puzzles based on music theory principles. While similar ideas have been explored in other projects, a limitation is that users do not engage with an actual instrument through this method.
- **Band Quest Game:** In this game, the user interacts with a music player who is losing band members and is on a quest to find new ones. While this can be engaging, it does not facilitate any learning for the user.
- **Music Note Reading Game:** This game focuses on teaching users to read music notes by simulating sheet music and indicating what should be played. However, it lacks elements of gamification, which could enhance engagement.
- **Story-Driven Keyboard Game:** This idea combines falling blocks with a music sheet and incorporates a storyline to motivate players.

The objective is to develop a game as an effective method for motivating students to learn music theory. The game must be enjoyable, accessible, and incorporate techniques to make learning easier. Furthermore, the game should include the principles of reading musical notation, a basic comprehension of music theory, and an opportunity to play the physical keyboard during the learning process. The best approach is to use some elements of all the presented ideas and create a game.

3.3 Concept Elaboration

The game contains four levels that will provide insights into rhythm and theorical knowledge of music. The game will provide a short story of a band that requires one participant to play keyboard with them. This will help to generate engagement with the game as well as a system of rewards and penalties.

The first level of the game will provide finger exercises, while the second level will introduce musical notation, because understanding the basics of a musical instrument and notation is crucial for fostering an interest in music. These two levels want to impart a basic understanding of melody in music. The third level will provide music theory lessons. The fourth level will focus on exploring the creativity of the player, building upon a backing track as a jam, providing a fun experience. Understanding music theory, including rhythm, could motivate the participants and help them engage in music theory. Participants will require them to achieve certain points to progress from level one to two. In the tutorial, participants will be able to pass to the next level by simply playing inside the game. And after the fourth level, the game will finish.

To decide how to create a new application for educational purposes using gamification principles, the enjoyment and effectiveness of the game will be evaluated. Each session is expected to last approximately 45 minutes. After the trial, participants will fill out a questionnaire about their experience with the game. The questionnaire will not only evaluate participants' enjoyment of learning music theory but also gather feedback about the game.

The analysis will use a combination of qualitative and quantitative analysis. The quantitative aspect will involve questions with five-point Likert scale (Joshi et al., 2015) options to measure specific aspects such as engagement and satisfaction. The qualitative aspect will include open-ended questions to capture participants' emotions, likes, dislikes, and suggestions for improvements. This combination will allow us to collect comprehensive data and understand the expectations of potential customers. The analysis in the project will uncover trends and identify potential issues early in the development process.

An application will be created using one of the accessible current devices for mixed reality. Due to the low price of these devices, they are expected to become widely adopted, meaning that the number of users will likely increase in the coming months or years. This new technology offers significant advantages for creating, learning, and exploring, making it a truly immersive tool that enhances the learning process. By augmenting the real world with new information using virtual objects, it becomes easier to experiment than with real objects.

3.4 Limitations

The purpose of this section is to outline the limitations for the research study considering a timeline of 6 months, which aligns with the timeframe for a master's thesis:

- Only one model of MIDI controller will be considered.
- The game will be limited to a basic understanding of musical theory, including the construction of a major scale for each key and the associated major and minor chords. The study of minor scales and other scales will not be included, nor will the study of chords such as diminished, or other forms.
- The musical notations employed in this project will be limited to whole notes, half notes, and quarter notes. Other notations will not be used as they are more challenging for beginners.
- The project will initially include only one simple song, not exceeding one minute in duration.
- This first version of the program will not create a database to store information from each session for every player. However, this could be considered for future implementation, as suggested in another project discussion.

3.5 Evaluation Concept of the Project

3.5.1 Evaluation Criteria:

Design Validation

To evaluate if this project is effective, it is essential to collect data from participants after they have completed the four parts of the game. The data will then be subjected to both quantitative and qualitative analysis using a series of prepared questions.

After testing the game, participants will complete a questionnaire created using Windows Forms. This digital form will help us collect all responses and export the data to Excel for analysis. The questionnaire includes questions designed to perform sentiment analysis, providing necessary data about enjoyment, motivation, and excitement in the game to answer the research question.

For the quantitative part, participants will rank their levels of enjoyment, excitement, and motivation on a scale from 1 to 5 for each game level. Thematic analysis will be used for the qualitative part to identify insights that will inform the development and efficacy of the tutorials and gather feedback for the project. Participants will give an opinion about the ease of setting up the game, which will help to formulate a new version of the game.

Another important aspect of every XR experience is simulation sickness, which will be assessed through a quantitative analysis with four different options to choose from. Participants will be asked to share their experience with XR devices before the game begins to evaluate their experience with the game, particularly new users of XR to understand the difficulty of the game.

Technical Validation

The game must successfully integrate the Meta Quest 3 with a MIDI controller and Cubase, ensuring seamless communication. This integration is crucial for providing an immersive and interactive experience. One important goal is to find a unified method to integrate all these devices and software programs. Figure 1 illustrates the necessary connections for the proper functioning of this project.

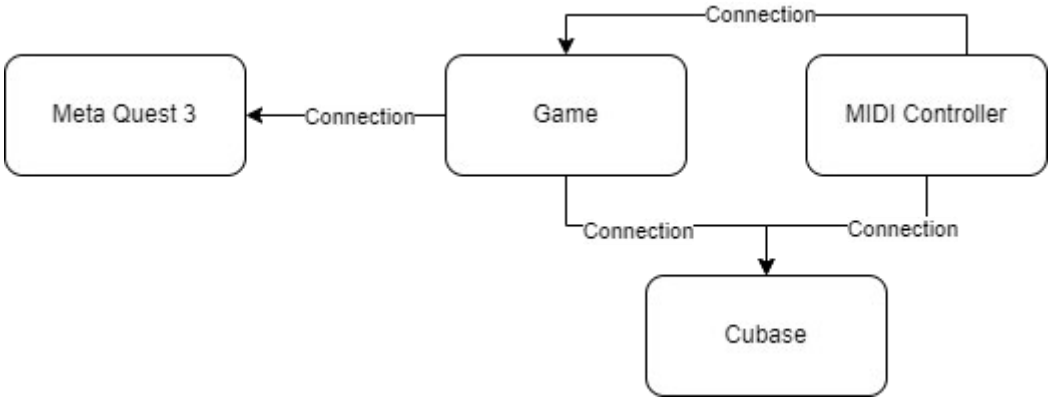

Figure 1 Desired integration of components in the project
Source: Own creation

The game should operate smoothly without any latency or drift, as these issues can significantly detract from the user experience. Drift, which refers to unintended movements or actions within the game environment, can disrupt gameplay and lead to frustration, particularly in a musical context where precision is essential.

Visually, the game is expected to leverage high-resolution graphics and immersive environments. Players should encounter a visually engaging interface that enhances their interaction with the MIDI controller and virtual objects. This could involve dynamic visual feedback that facilitates the creation of 3D models in the real world in an intuitive manner, ensuring that the experience remains enjoyable and not overly technically challenging.

3.5.2 Hypotheses for Validation

To help validate the design, hypotheses are created to evaluate the proposed goals. Organizing these hypotheses will make it easier to assess whether the project achieves its objectives.

User engagement refers to the willingness of the user to continue playing the game. It can be divided into enjoyment of playing the different levels, excitement about continuing the musical journey, and motivation to gain more insights into musical concepts. Additionally, aspects such as learning outcomes and the functionality of the game will be evaluated with the following hypotheses:

Willingness to play the game:

Ha0: The challenges presented at each level increase the overall enjoyment of the game.

Ha1: The challenges presented at each level do not increase the overall enjoyment of the game.

Hb0: Users are excited to continue their musical journey with more lessons in this game.

Hb1: Users are not excited to continue their musical journey with more lessons in this game.

Hc0: After completing the game, users are motivated to gain further insight into musical concepts.

Hc1: After completing the game, users are not motivated to gain further insight into musical concepts.

Functionality of the Game:

Ha0: The game is easy to set up and use.

Ha1: The game is not easy to set up and use.

Hb0: The use of the mixed reality game feels comfortable during use.

Hb1: The use of the mixed reality game produces discomfort during use.

Learning outcomes:

H0: Participants learn to identify the C key or C scale after playing the game.

H1: Participants do not learn to identify the C key or C scale after playing the game.

4 Creation Game

For the development of the project various tools are used, Blender is utilized for 3D graphics, either creating models ourselves or downloading them from the web. For audio is Cubase the tool, a comprehensive software for music production that will provide the audio for this project, LoopMIDI by Tobias Erichsen is used to connect the MIDI controller to other programs, such as Cubase and Unreal Engine. Additionally, MIDI-OX replicates the MIDI signal from one input channel to two outputs.

4.1 Programs in the project

Blender

For 3D creation tools that support 3D designs, there are many options available in the market, such as Maya, ZBrush, and Adobe Substance 3D Modeler. Many of these tools are similar, with only a few differences. Some offer more client support or make the process of exporting to game engines easier.

Blender is a free, open-source software that comes with many plugins. It is not heavy on computer resources, requiring less RAM and overall capacity. However, a significant issue with Blender is the creation of materials for exporting 3D models to a game engine. The export format is **.fbx**, which does not support procedurally generated textures due to the different render pipelines in Blender and game engines.

Because of this, it is better to find or create textures in Blender and add the materials separately in the game engine. The solution is to create the model in Blender, generate UV coordinates, and export it to the game engine. Alternatively, it is possible to bake textures in Blender and use these in the game engine. **Baking** is the process of pre-calculating shades and storing them in a static image texture. This process can incorporate features such as glossy effects, diffuse maps, bump maps, etc. Using baking may improve light quality and realism at the expense of dynamism, which can help enhance the performance of the game on some mobile devices.

Cubase

Cubase is a Digital Audio Workstation (DAW) produced by Steinberg, with many years in the market. It allows users to create music, record MIDI, sequence, and edit audio signals. It is very flexible and offers numerous tools for creating MIDI channels. Cubase operates on both Windows and macOS. Users can create music in mono, stereo, or ambisonics, using data from the headset to set the position, which is essential for 3D sound.

The main aspects that will be used for this project are the creation of a project in a MIDI channel that will be sent to the project in Unreal Engine and the use of ambisonics to achieve spatial 3D sound.

Cubase Plugin: It is necessary to create a plugin in Cubase to read the data from the channel and send it to Unreal Engine. OSC (Open Sound Control) connection in Cubase is one method to receive instructions and send data. Additionally, the rotation of the headset will be shared through OSC.

Loop MIDI

LoopMIDI, written by Tobias Erichsen, is a software that creates virtual looped MIDI ports on a computer, allowing several programs that read MIDI ports to be connected to one another in Windows. For this project, two MIDI ports will be created to connect two programs: **Cubase**, to read and play the corresponding sound from the MIDI controller, and **Unreal Engine**, to read the key played on the MIDI controller's keyboard and ensure that the correct note is pressed.

MIDI OX

MIDI-OX is software that can generate MIDI data or replicate data coming from one port. The problem is that Cubase takes control of the entire MIDI port bus as soon as it is opened, preventing Unreal Engine from receiving any data, even if it is opened first. MIDI-OX is necessary because it allows us to select one MIDI port as the data input and choose two virtual loop MIDI ports as the output ports. This way, MIDI data can be used in both Cubase and Unreal Engine.

MIDI OX will read the MIDI values from the MIDI controller and send them through the virtual MIDI ports created with LoopMIDI. Figure X illustrates the interaction between LoopMIDI and MIDI OX for better understanding.

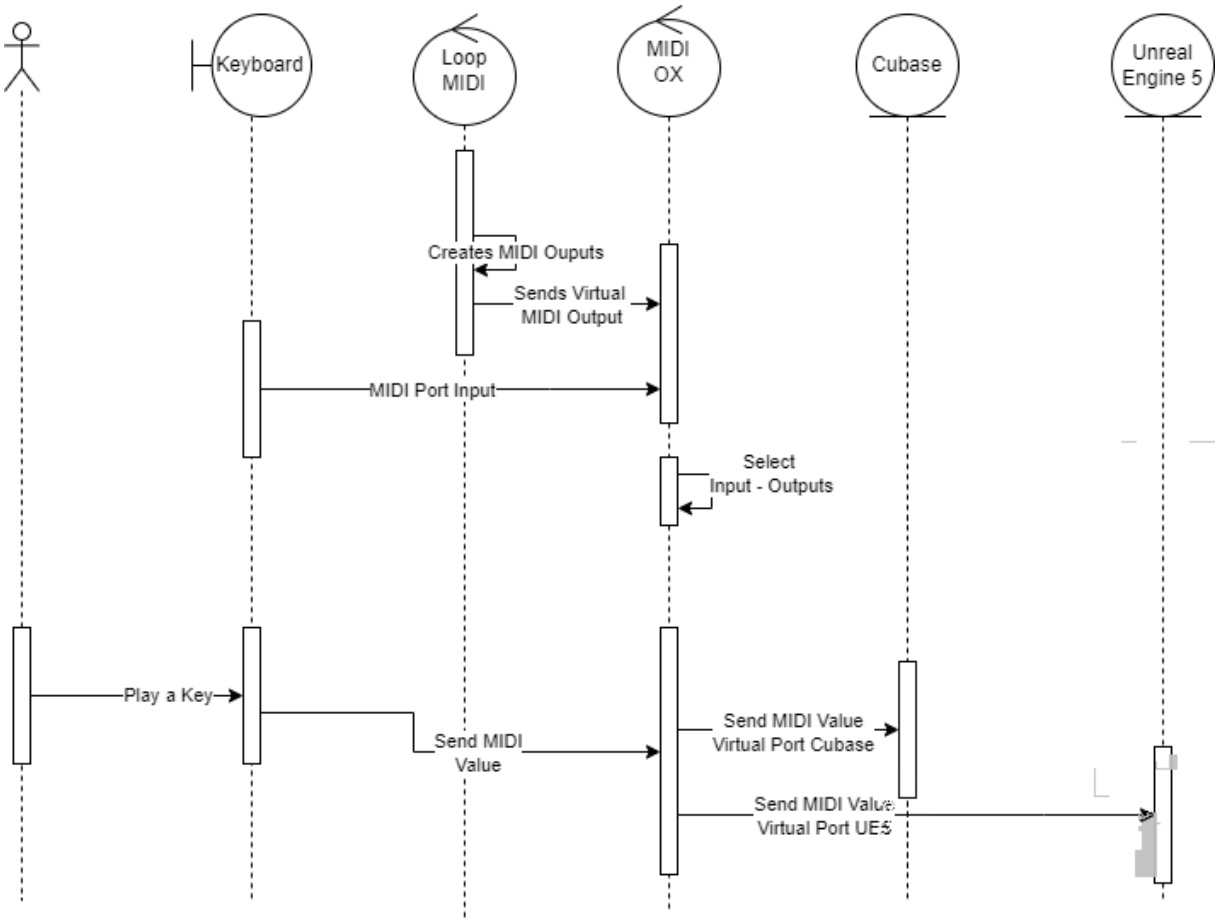

Figure 2 Loop MIDI - MIDI OX Interaction
Source: Own creation

Unreal Engine 5

This project will be done using version 5.3.2 of Unreal Engine. At the time of design, this was the latest stable version compatible with Meta XR, a plugin that facilitates the creation of VR apps for Meta products like Meta Quest 2 and 3. However, a notable limitation of this version is that it does not support the Open XR plugin, which connects VR devices from other companies, to function concurrently with Meta XR. Consequently, Open XR must be disabled, and only Meta XR components should be used. With the release of Unreal Engine 5.4.1 in August, this integration issue between Open XR and Meta XR has been resolved.

4.2 Plugins for the project

Meta XR

One plugin used in this project is Meta XR, which helps develop solutions for Meta Quest 2, 3, and Pro. To create applications for other headsets, it is necessary to use another plugin, such as OpenXR. However, OpenXR currently has a limitation: it does not support the use of the camera in the headset. This means it does not allow the passthrough function with the Meta Quest 3 and does not track hand movements.

Passthrough is essential for this project as it is crucial for a mixed reality application. The mixed reality solution is better on this project instead of a complete virtual reality solution because the main concept is to enhance a real instrument with 3D objects to help the user learn music concepts while using the instrument.

The Meta XR plugin includes key features such as hand tracking, compositor layers, passthrough layers, and spatial anchors. These features will be used to create immersive and interactive mixed reality experiences. The latest version of Unreal Engine, 5.4.1, has resolved the integration issues between OpenXR and Meta XR, allowing for better compatibility and functionality.

Open XR

OpenXR is a royalty-free, open standard developed by the Khronos Group. It aims to create a standard set of APIs for developing XR (Extended Reality) experiences. The goal is to develop an application once and deploy it across many devices without needing to create device-specific versions.

A better understanding of this concept is explained in the following graphic (Figure 3).

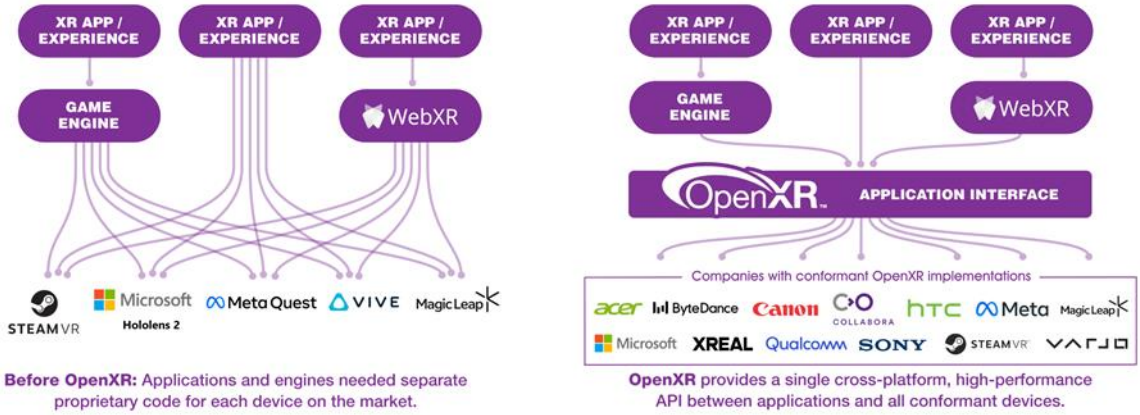

Figure 3 OpenXR Explanation Concept
Source: (OpenXR - High-Performance Access to AR and VR —Collectively Known as XR— Platforms and Devices, 2016)

This solution seems ideal, but some features are not available in the current API. For example, the Meta Quest 3 passthrough feature, which is essential for spawning virtual objects and creating a mixed reality experience. However, Meta has not yet opened this framework to OpenXR, which means that this feature can only be used through the tools provided by Meta.

MIDI Device Support Plugin

This plugin offers the possibility to read and send data from MIDI devices in real time. Currently, reading MIDI data is not supported.

This project will use MIDI input, so it is necessary to set the MIDI input device by selecting our MIDI controller MIDI events such as “On MIDI Note On” are used to listen for when a key is pressed on the

keyboard. One of the most common MIDI messages is the Note On message. This message consists of a status byte that both identifies it as a note on message and indicates the channel it is intended for (a value from 0 to 15). This is followed by a pitch data byte and a velocity data byte. Because the most significant bit of each data byte is used to indicate that it is a data byte, both the pitch and velocity data can only contain values between 0 and 127.

The MIDI protocol also specifies how these data bytes should be interpreted. For example, it defines how to translate a pitch value from 0 to 127 into a frequency in Hertz (Hz). While Unreal Engine does not directly translate MIDI data in this way, allowing users to use MIDI messages to control behaviour beyond audio, the Sound Utilities plugin contains several functions that can be used to perform MIDI data conversions.

The development uses the example (Figure 4) from the development website.

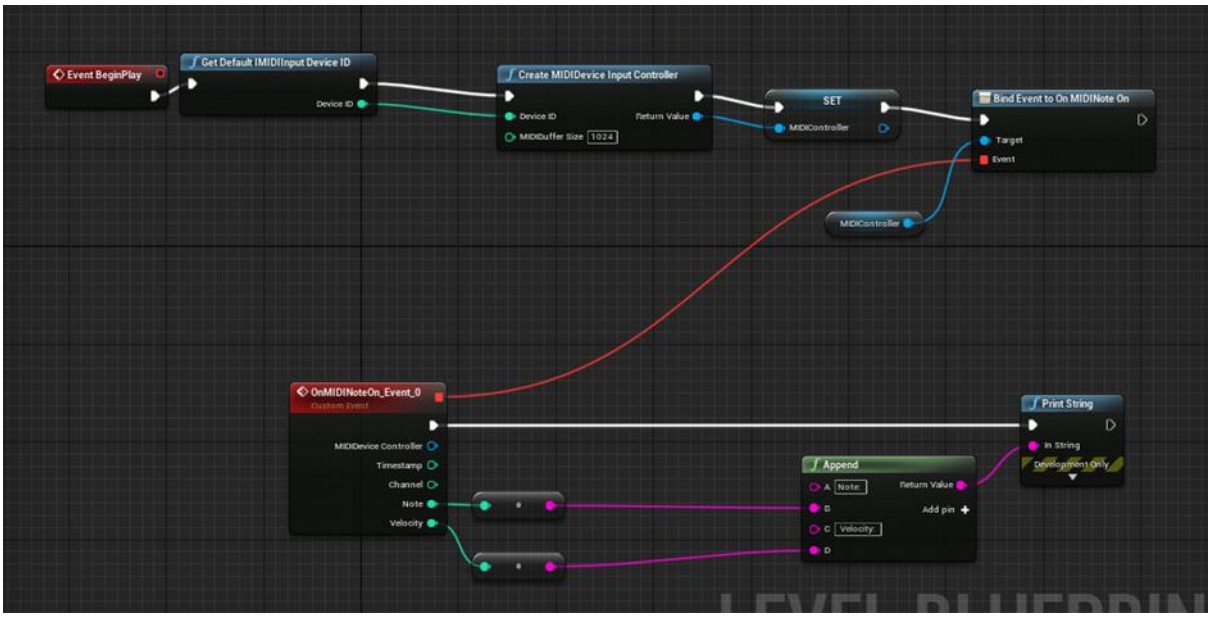

Figure 4 MIDI Device Events
Source: (MIDI in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community, n.d.)

OSC Plugin

Open Sound Control (OSC) is an open protocol used to communicate mainly audio data between clients, sending messages through a configured port. However, it can also be used to communicate non-audio data. In this project, it is used to communicate between Cubase and Unreal Engine. This plugin can create an OSC server and an OSC client.

4.3 Connection of Components

The connection is made through Meta Quest 3 and the computer using Quest Link, which makes Quest 3 function as Meta Rift in the interface. This means that the game will be rendered on the local machine. A MIDI controller will be connected to the computer, and the computer should have Cubase installed.

This project focuses on the Panorama T4 (Figure 5) by Nektar as the MIDI controller. The Panorama T4 features 49 keys, a drum pad, additional control buttons, and controllers for quick control of plugins.

Figure 5 MIDI Controller Panorama T4
Source: ("Panorama T4 and T6 ▷ MIDI Controller Keyboards | DAW Control | Plugin Control," n.d.)

Figure 6 will help to understand the external connection and the devices used in this project:

1. **Meta Quest 3**: Connected to the computer via Quest Link.
2. **Computer**: Running the game and Cubase, with the Meta Quest 3 working as Meta Rift through Meta Quest Link.
3. **MIDI Controller Panorama T4**: Connected to the computer for input.

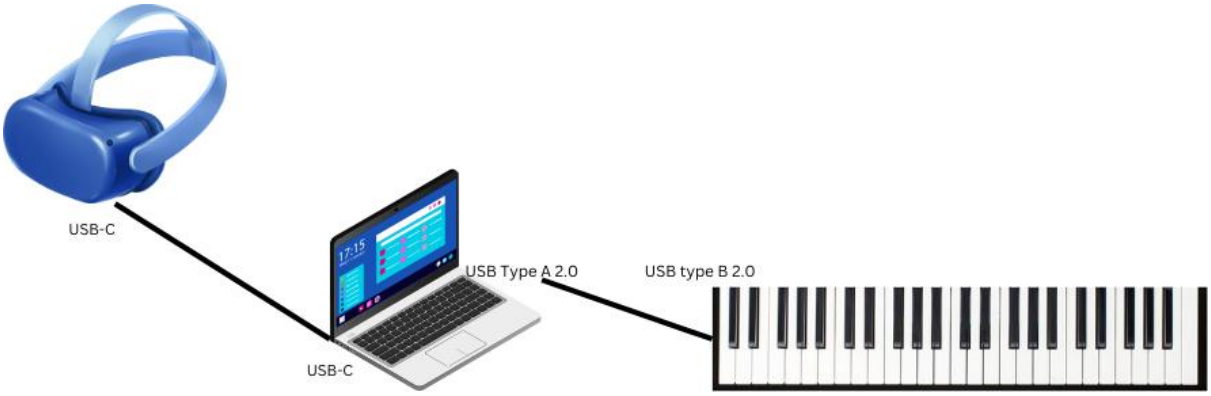

Figure 6 Connection Headset – Laptop – Keyboard
Source: Own creation

4.3.1 Game Process

The main idea is to have two games, a tutorial and an immersive experience of playing with a band. Figure 6 shows an overview of the options available to the player, now these options will be explained:

1. **Configure MIDI Keyboard:** The user will spawn a keyboard in the real world, laying the virtual keyboard over the physical MIDI controller and selecting the right MIDI port for it. At this point, the connection with Cubase will be established.

2. **Level 1**: A panel will appear in front of the user's view above the virtual keyboard. The rectangles needed for the level will be displayed moving from top to bottom. The user's goal is to press the corresponding key at the precise moment that the rectangle arrives at the corresponding key on the keyboard, as required by the song. The movement of the blocks will pause until the user hits the note. At this point, the targeted key will change color. If the user hits the key correctly, the song will resume, and a score will be awarded. If the user hits an incorrect key, it will wait for 4 seconds for the right key, and the points will decrease until reaching 0 points before continuing with the next key.
3. **Level 2**: The same pattern is presented to the user, but rather than blocks descending from top to bottom, the program displays a music sheet with notes and an indicator signaling whether the note will play. The music sheet has two clefs. The treble clef, used for the right hand, indicates higher pitch notes typically played with the right hand on the piano. The bass clef, used for the left hand, indicates lower pitch notes typically played with the left hand. Notes appear on the music sheet and move from right to left. When they reach the end, they disappear, and the corresponding note on the keyboard lights up in a different color. A scoring system records how quickly and accurately the user plays the notes, along with any mistakes made. And it is crucial to always display the name of the keys.
4. **The tutorial** will cover aspects of music theory with visual representations. No points will be awarded for this section.
    - Major scales
    - Minor scales
    - Chords
    - Chords within the scale
5. **Play with the Band:** An immersive experience playing the keyboard. The scene will change, a backing track will play, and the keyboard will display all the keys corresponding to the selected scale. The user will be presented with a virtual studio that offers the possibility to see the virtual keyboard and select the respective scales. Thanks to Mixed Reality glasses, it will provide information about the scale and all the corresponding notes of the scale will be highlighted on the keyboard. The user will have a feeling for a jam session listening to many instruments playing at the same time allowing them to feel the thrill of playing in a band. The audio can come from a variety of instruments such as a drum kit, a guitar, or a bass.

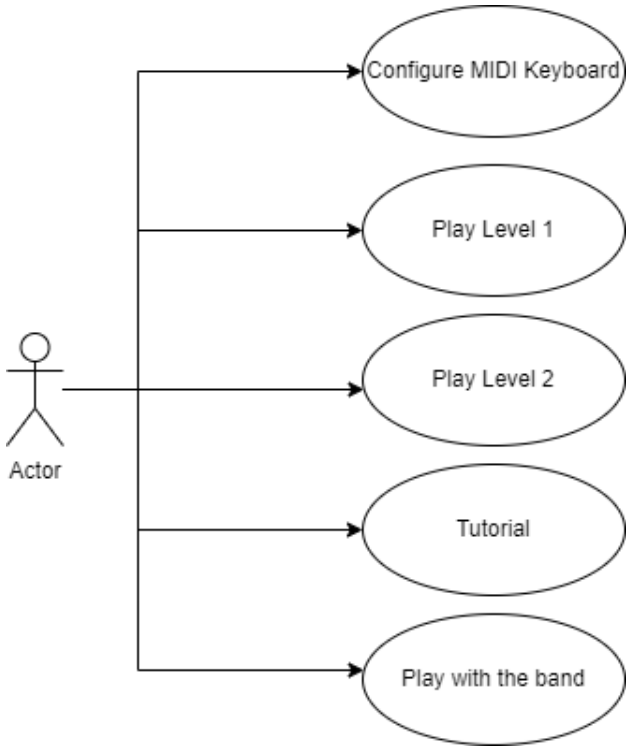

Figure 7 Use Case Main Menu
Source: Own creation

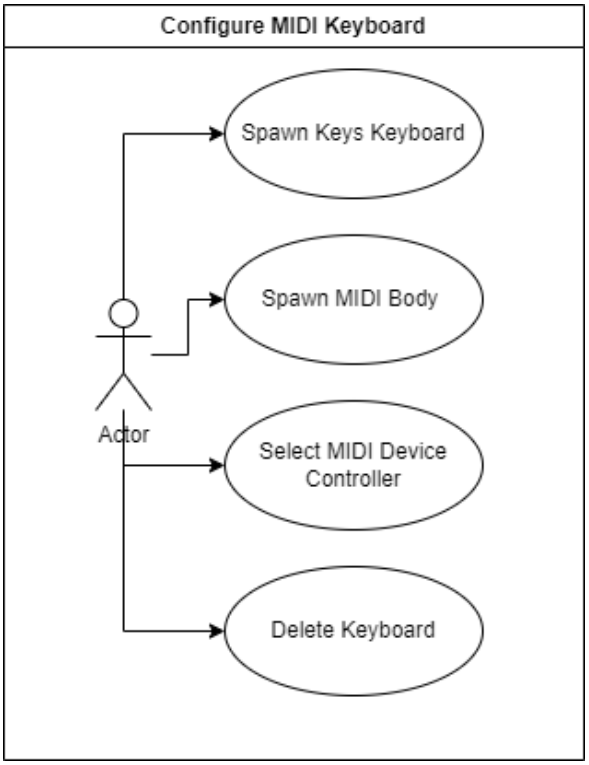

Figure 8 Use Case Configure MIDI Keyboard
Source: Own creation

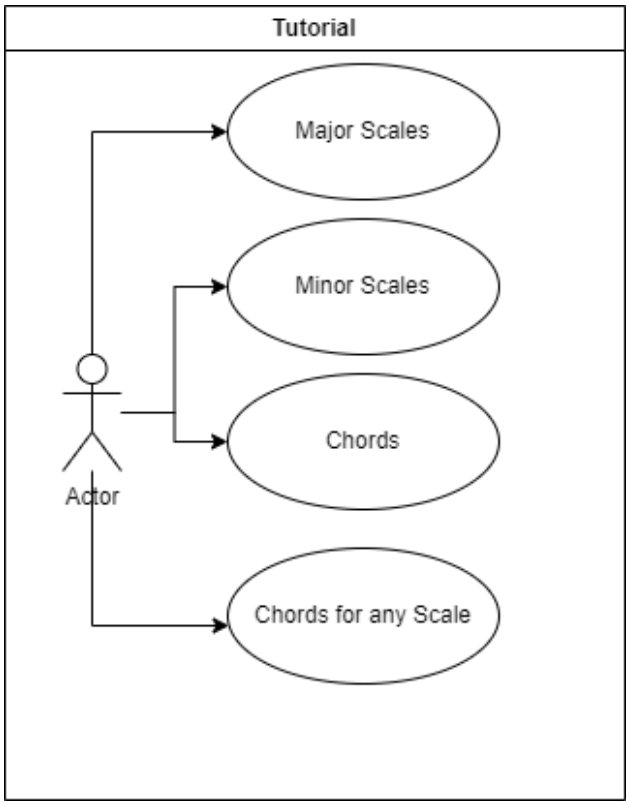

Figure 9 Use Case Tutorial
Source: Own creation

4.3.2 Cubase

Plugin for Cubase

To integrate Cubase with our project, it was necessary to develop a plugin. Here is a brief overview:

- **Language**: The plugin is written in C++.
- **Technology**: It uses Steinberg's Steinberg Kernel Interface (SKI) technology to gather all information from the channel named "Unreal" in Cubase.
- **Functionality**: The plugin converts this information into a string to be sent through the OSC client. It also incorporates an OSC server to listen to incoming commands.
- **Network Configuration**: Since Cubase and the game in Unreal Engine are on the same computer, it will use the internal local IP address (127.0.0.1) and just differentiate the ports by using 1338 to send data, 1337 to receive data, and 1340 to send head rotation data.

Configuration Project in Cubase

The project in Cubase is configured with multiple channels where the instruments for the backing track are recorded. These Audio Tracks Channels can use the Ambisonics plugin to create 3D sound.

- **Channel for Unreal Engine**: One channel, named "Unreal," is sent to Unreal Engine for use in the game. This must be a mono channel.

- **Head Rotation Plugin**: There is a plugin called “HeadRotation” that uses input from an OSC connection to change the position of the head. This means that the Quest 3 headset’s rotation is read in the game and sent through the OSC connection to port 1340 to adjust the sound of the backing track accordingly.

4.3.3 Graphic environment

To ensure the success of the project, it is essential to maintain a friendly atmosphere. To achieve this, it is necessary to provide the user with an interactive element. This will be achieved by creating the robot “Tone Tutor”, which will serve as an easily recognizable model for providing instructions to the player. The robot will provide all the necessary information about music theory.

The environment (Figure 7) for the final part of the project is inspired by a music studio, complete with guitar amplifiers, guitars, drums, and bass. Additionally, there are two types of robots. Other essential items for the game include a virtual keyboard, a music sheet, and notes that will appear in the game. All these items are detailed in the appendix.

Figure 10 Scenario "Play with the Band"
Source: Own creation

4.3.4 Development Game

In this type of game, assets are created whenever they are needed for the music. These assets (3D models or actors) perform an action and then need to be removed after a certain time or upon reaching a specific point. This process consumes many resources and can lead to memory leaks if garbage collection in C++ is not handled properly. Frequently deleting objects can also cause memory defragmentation. To

mitigate these issues, developers often use object pooling or the pool pattern, where a set of objects is created once and reused, rather than being created and destroyed repeatedly. This approach helps manage resources more efficiently and reduces the risk of memory leaks.

Pool Pattern:

The object pool pattern works as follows:

1. Initialization: A pool of objects is created at an exact moment, commonly at the start of the game. These objects are initially invisible and have their tick function disabled, indicating they are not in use.
2. Reuse: When an object is needed, it is taken from the pool, made visible, and its tick function is enabled. This object is then used in the game.
3. Return: After the object is no longer needed, it is returned to the pool, made invisible again, and its tick function is disabled. This makes it available for reuse.

Benefits of the object pool pattern include:

- Performance Improvement: By reusing objects, the game avoids the costly operations of creating and destroying objects, leading to smoother performance.
- Memory Management: It helps in managing memory more efficiently by reusing existing objects instead of allocating new memory for each object creation.
- Consistency: Ensures that objects are consistently available when needed, reducing the risk of delays or performance hiccups.

Some disadvantages of this pattern are:

- The project can consume more memory by creating objects that won't be in use, or creating too few objects can produce errors in the program. Finding the right number of objects can be challenging.
- It can increase the complexity of the code, as the lifecycle of these objects needs to be managed, ensuring they are reset and available after a certain time.
- The creation of the pool can temporarily increase resource consumption when it is initialized.

Unreal Engine 5

Unreal Engine uses Blueprints, which offer diverse options for creating elements, such as Actors, Pawns, and Actor Components. Blueprints have two main functions: **Begin Play** and **Tick**.

- **Begin Play** is a predefined event that is called once at the start of the game.
- **Tick** is called every frame, meaning it could be called 60 times per second or more, depending on the game’s configuration.

UE5 refers to Actors to the packets that contain code and they can include visual representations of objects. These objects can be spawned in the game, allowing us to interact with them. At the beginning of the game, there is an important section called the Outliner.

The **Outliner** is the section of the game that contains all the objects the game will use. Figure 8 shows a detailed view of the Outliner. When an object is created, it appears in this section. At the beginning of the game, the following objects are shown as invisible: **BP_scenario**, **BP_Drums**, and **BP_E-Guitar**. It also contains the following Actors with the Tick function disabled: **BP_GameSystem**, **BP_OSC** and **MRUKAnchorActorSpawner**.

The **Mixed Reality Utility Kit Anchor Actor Spawner (MRUK)** is a predefined blueprint created automatically. It contains all the information about the room. When a person uses the mixed reality option of Quest 3 for the first time, it requires scanning the actual room to detect tables, windows, floors, walls, plants, screens, and more. All this information is saved in the Meta Quest, with the possibility to scan other rooms and save their information as well. For every object, the coordinates of the position and dimensions are saved, along with any assigned material.

Player Start is an important component that holds crucial information about the game, as it references our **VRPawn**. It contains all the information for the camera, hands, and configuration for Virtual Reality on the Quest 3 for the experience. **VRPawn** contains a camera component and configurations to recognize the headset and controllers. It also includes configurations for the passthrough option and actions that the hands or controllers can perform. **VRPawn** has an actor component, **AC_VRManager**, which creates the Robot and the UMG Display in the virtual world.

BP_GameSystem is the blueprint that determines if the function should unpause after a 4-second pause. It is also responsible for deciding if the user wins or loses the game. **BP_GameSystem** calculates the number of points in the game: if the player plays the correct note within 4 seconds, the game adds 4 points to the total; after 3 seconds, it adds 3 points, and so on. If the player does not play anything, no points are added. The decision to determine if the player wins or loses is made as follows:

$$Points\ in\ the\ game > \left[\left(\frac{\{\{total\ number\ of\ notes\ to\ be\ spawned\} \times 4\}}{\{2\}}\right)\right]$$

Formel 1: Decision Win or Lose
Source: Own creation

This means that if the player correctly plays more than half of the total notes within 4 seconds before the note expires, they will win.

The following graphic illustrates the relationships and interactions between all the objects in our project. It provides a comprehensive overview of how each component is connected and functions within the system:

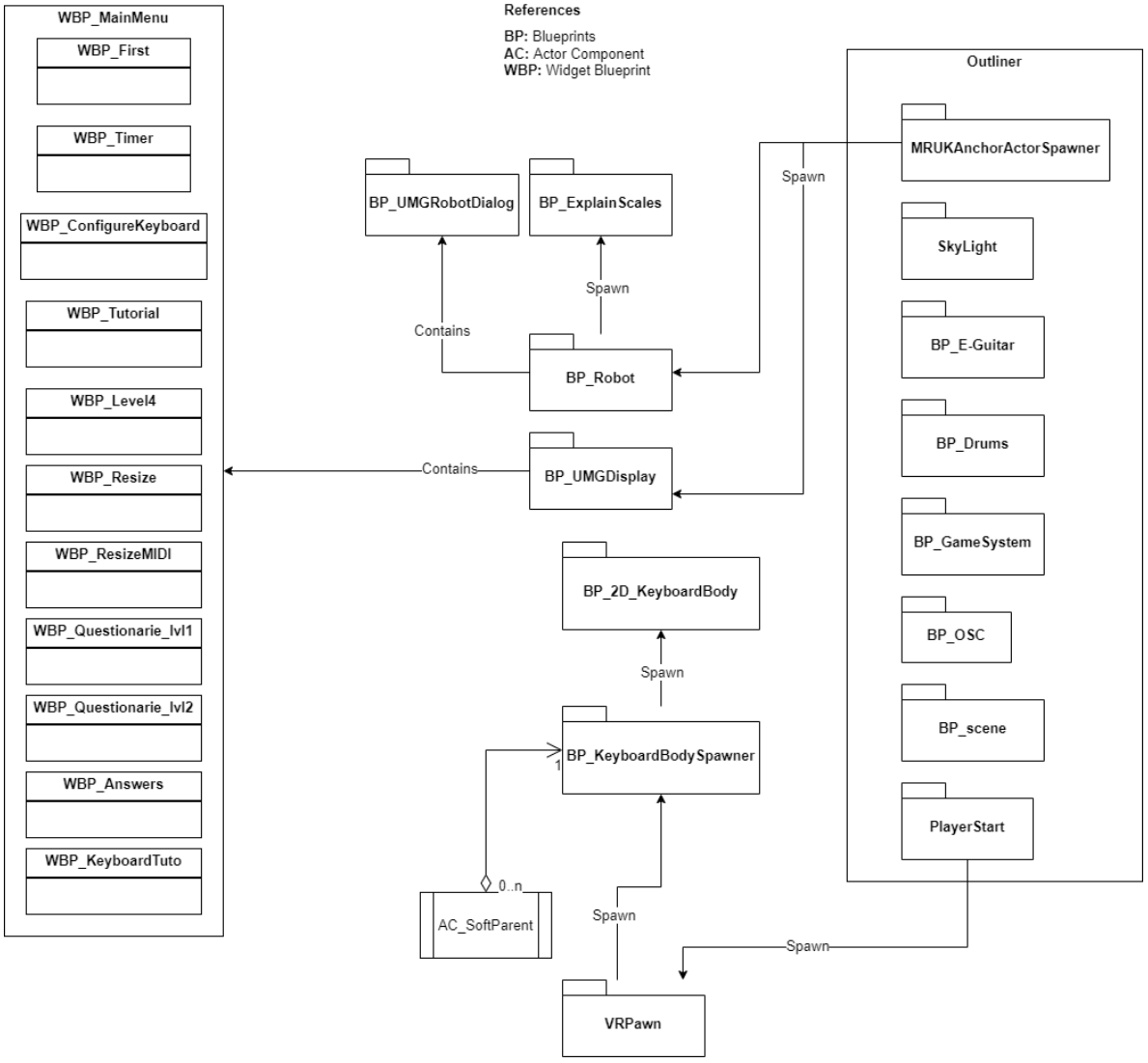

Figure 11 Blueprints Connection 1
Source: Own creation

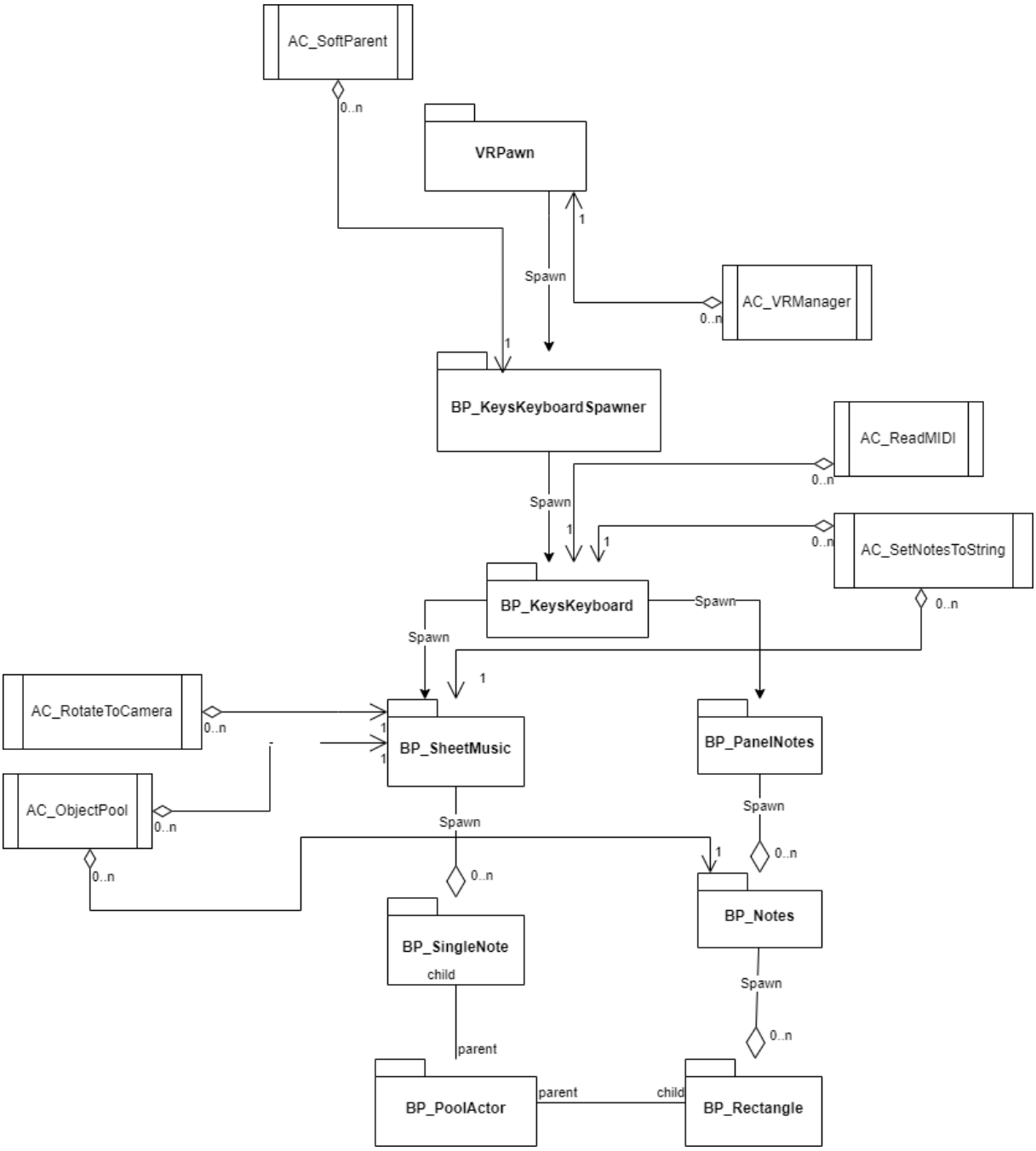

Figure 12 Blueprint Connections 2
Source: Own creation

BP_Robot uses the following information to be spawned: the nearest table and the direction the user is looking at. **BP_Robot** is an Actor blueprint that contains the meshes of the robot and has a child, **BP_UMGRobotDialog**. **UMG Robot Dialog** contains a widget that displays all the information the user will see. **BP_Robot** creates the actor **BP_ExplainScales** in the game.

When the blueprints **BP_KeyboardBodySpawner** and **BP_KeysKeyboardSpawner** are available, they have an actor component called **AC_SoftParent**. This component fixes the object to the controller. The object waits to be activated in the controllers. To activate it, press the trigger button on the desired controller, and the object will appear on that controller. Move it closer to the real MIDI Controller before

releasing the trigger. At that moment, the blueprint will be destroyed, and another blueprint will be created in the virtual world. If **BP_KeyboardBodySpawner** is chosen, a **BP_2D_KeyboardBody** will be created. If **BP_KeysKeyboardSpawner** is chosen, the actor **BP_KeysKeyboard** will be created in the virtual world.

BP_2D_KeyboardBody is a blueprint that contains visual representations of the drummer pad and other buttons on the MIDI controller. The design is based on the MIDI controller Nektar Panorama T4. It includes tags over the quick controls with their names.

BP_KeysKeyboard is the blueprint that contains 3D models of the keyboard, based on the Model Panorama T4, including the size of each key and the 49 keys present in that MIDI controller. This blueprint is crucial as it creates the two games within the project: **BP_SheetMusic** and **BP_PanelNotes**.

BP_KeysKeyboard includes two actors components: **AC_ReadMIDI** and **AC_SetNotesToString**. **AC_ReadMIDI** facilitates the use of the MIDI Device Controller plugin, while **AC_SetNotesToString** filters the received messages from the OSC, converting those strings into an array of structs with all the necessary data, and creates the notes for the MIDI data.

BP_SheetMusic contains a music sheet with a treble clef and a bass clef. This blueprint includes two actor components: **AC_RotateToCamera** and **AC_ObjectPool**. **BP_SheetMusic** receives information about the notes and the timing for their creation in the game. This information is sent at the appropriate moment to **BP_SpawnNote**.

AC_RotateToCamera is an actor component that rotates each object to face the camera.

When the blueprints **BP_KeyboardBodySpawner** and **BP_Keys KeyboardSpawner** are available, they include an actor component called **AC_SoftParent**. This component attaches the object to the controller. The object waits to be activated in the controllers. To activate it, press the trigger button on the desired controller, and the object will appear on that controller. Move it closer to the real MIDI Controller before releasing the trigger. At that moment, the blueprint will be destroyed, and another blueprint will be created in the virtual world. If **BP_KeyboardBodySpawner** is chosen, a **BP_2D_KeyboardBody** will be created. If **BP_KeysKeyboardSpawner** is chosen, the actor **BP_KeysKeyboard** will be created in the virtual world.

BP_2D_KeyboardBody is a blueprint that contains visual representations of the drummer pad and other buttons on the MIDI controller. The design is based on the MIDI controller Nektar Panorama T4 and includes tags over the quick controls with their names.

For objects that need to be created frequently, the project uses the **Object Pool Pattern**. This pattern involves maintaining a pool of reusable objects, rather than creating and destroying objects repeatedly. This approach improves performance by reducing the overhead associated with object creation and destruction.

AC_ObjectPool is an actor component that creates multiple objects in the game. It is part of the object pool pattern solution, along with **BP_PoolActor**. **AC_ObjectPool** creates many objects in the scene, which are shown in the Outliner when the game is running. These objects are initially created as invisible and with the tick function disabled, indicating they are not in use.

BP_PoolActor recognizes the blueprints that need to be created as part of the object pool. The desired object should be a child of **a Pool Actor**.

BP_SingleNote is a child of **BP_PoolActor** and is called from **BP_SheetMusic**, which activates these objects to be playable in the game. **BP_SingleNote** represents the music notes and has parameters for the duration of each note to create whole notes, half notes, etc. **BP_SheetMusic** reads the array of notes and activates each note at the correct time. It also includes the component **AC_ObjectPool**.

For the other game, **BP_PanelNotes** is the main blueprint. It receives information about the notes to be created and organizes this information in a dictionary, where the note is used as a key, and the spawn time and duration are the values. The game then creates one **BP_Notes** for each note in the dictionary.

BP_Notes is an actor blueprint with components such as **AC_ObjectPool**, a line for the movement of the block note, a beginning spot, and an end spot. **BP_Notes** activates the blocks at the appropriate moment, and the block moves along the line from the beginning spot to the end.

BP_Rectangle is the blueprint that represents the block note to be played. It is a child of **BP_PoolActor** and is activated in **BP_Notes**. **BP_Rectangle** receives information about the duration of the note and modifies the length of the rectangle accordingly. For example, a short rectangle could represent a quarter note, a longer rectangle could represent a half note, and an even longer rectangle could represent a whole note.

For navigating through the game, a menu appears directly in front of the user's camera view. It is contained in the Actor **BP_UMGDisplay**, which is created in the game using the position of the table closest to the user. This utilizes the information of the room contained in the **MRUK Anchor Actor Spawner**.

Inside **BP_UMGDisplay**, there are two objects: a Widget and a Niagara Component Particle System called **menu laser**. The Widget contains the menu, **WBP_MainMenu**. The menu laser is a line created using the Niagara particle component, starting at the detected controller and ending at the menu.

WBP_MainMenu contains a **Widget Switcher Component**, which manages the transition between different widgets. The entire connection is shown in Figure 9. It holds references to all the widgets within the project. The following graphic illustrates the relationships and interactions within the UI Menu, detailing how each component is connected and managed.

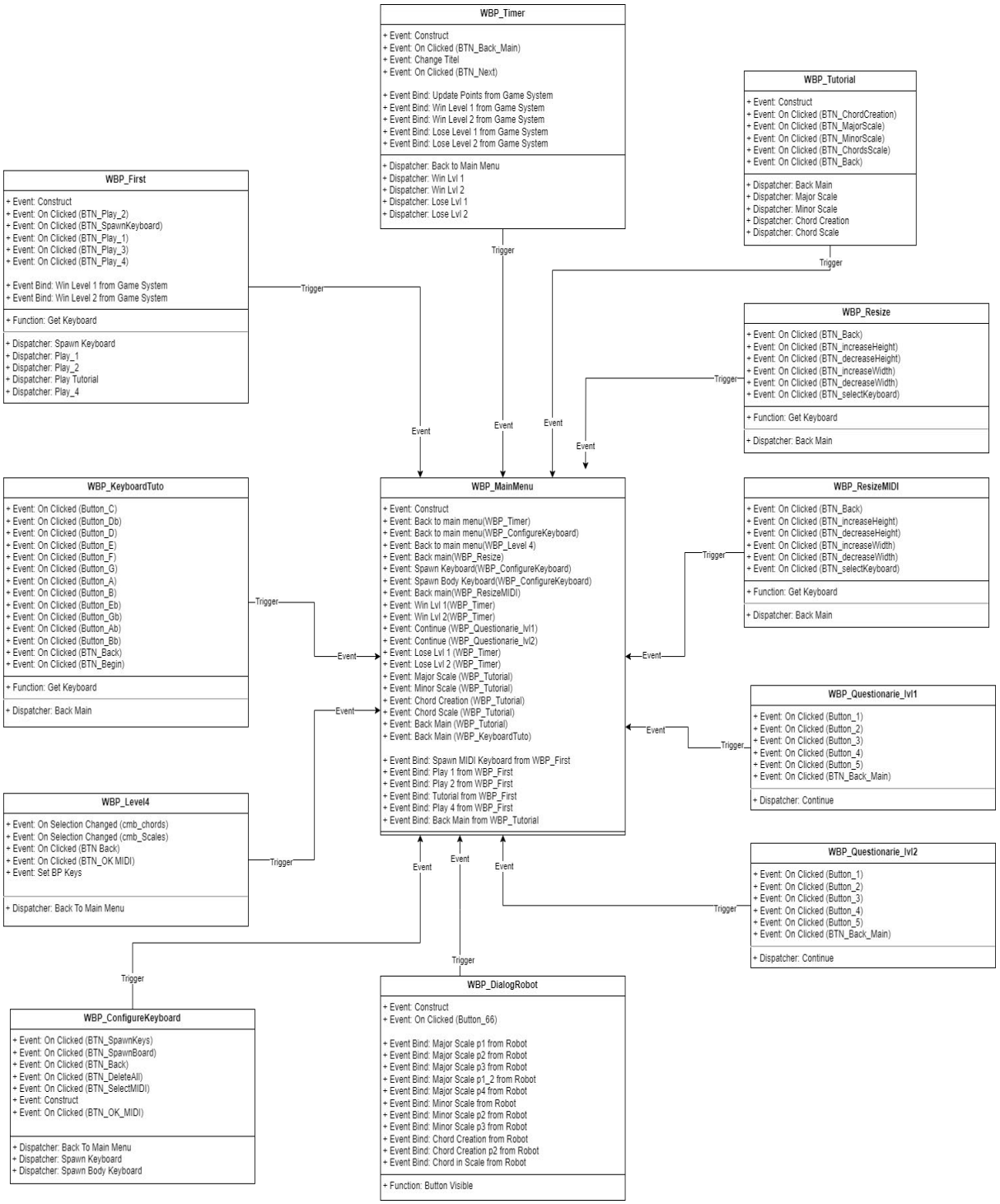

Figure 13 Main Menu Connection
Source: Own Creation

4.3.5 Development per Level

Development Configuration Keyboard

This is a possible idea for connecting the components during the configuration of the MIDI Controller. At this point, Quest 3 should send data from the scanned surface to Unreal Engine. The game will then use this room data to send virtual objects to the user.

To complete the configuration, it is important that the MIDI Controller sends all available MIDI ports for the device. The user should select the correct port, and the game will be ready to play.

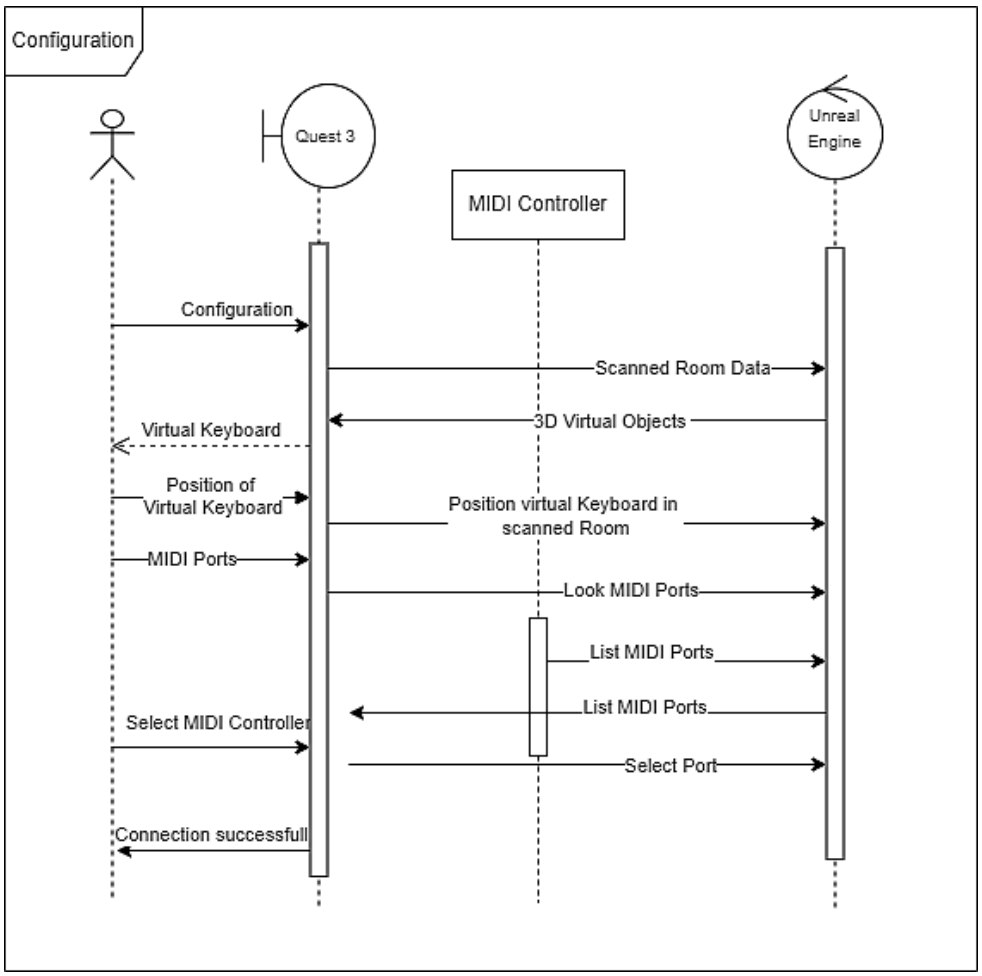

Figure 14 Configuration Keyboard Sequence Diagram
Source: Own Creation

The configuration of the keyboard is not just about spawning a virtual keyboard in the game. It is necessary to create an array of notes, which requires receiving a message from Cubase through the OSC connection.

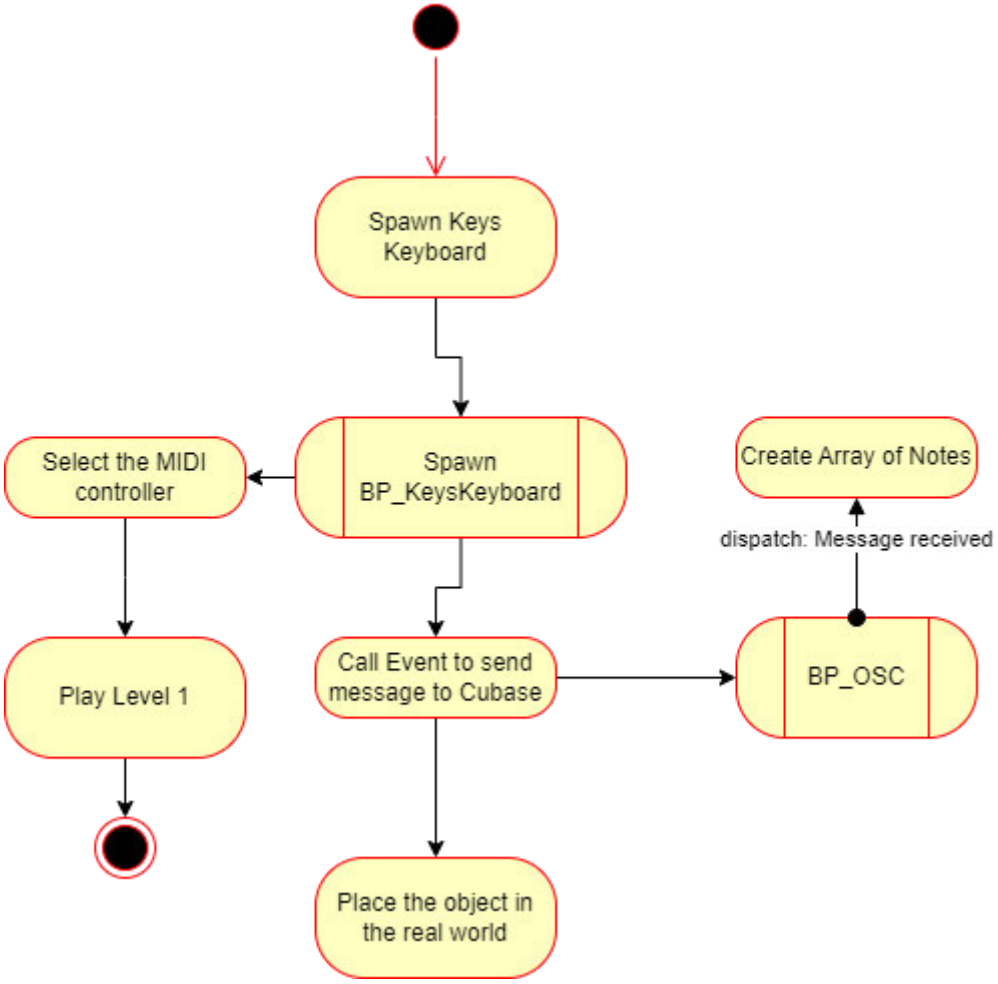

Figure 15 Configuration Keyboard Activity Diagram
Source: Own Diagram

The OSC connection takes place between the project in Unreal Engine and Cubase. Using SKI, Cubase can collect the MIDI data from the project, convert it to a string, and send it through OSC to the application.

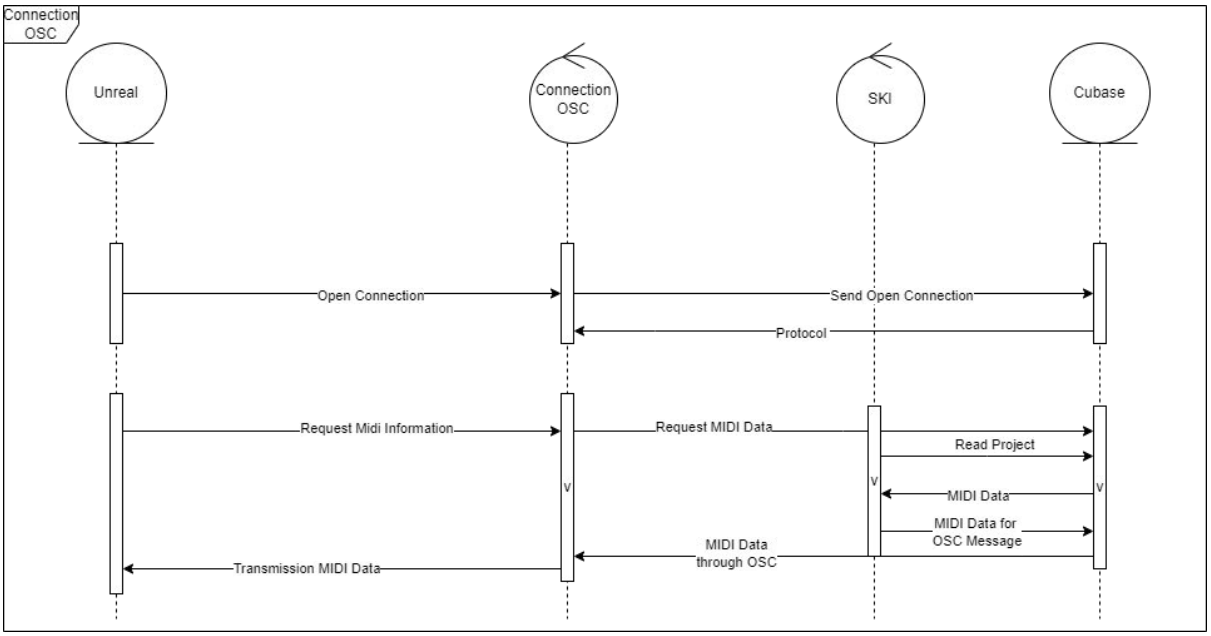

Figure 16 OSC Connection Sequence Diagram
Source: Own creation

Spawn Keyboard

When the game begins, Player Start calls VRPawn, which contains AC_VR Manager. VRPawn has the function to grab objects near the hand and display a small menu on the hand. This menu is predesigned in the template and includes an option to quit the game.

When the user configures the keyboard and selects the option to spawn keys, the blueprint BP_KeysKeyboardSpawner is created in the game. This blueprint includes an actor component, AC_SoftParent. AC_SoftParent looks for the player's game controller. After the trigger button on the controller is pressed, AC_SoftParent uses the tick function to attach the actor to the controller. Upon releasing the trigger button, the actor is destroyed, and the blueprint BP_KeysKeyboard is created. The same process is repeated with BP_2D_KeyboardBody.

Figure 17 Spawn Keyboard Class Diagram
Source: Own creation

Level 1 – Game with Rectangles

The idea is that the application will send notes to Meta Quest 3 for the user to play. Blocks will fall, and the key will change color. The system will wait 4 seconds while sending a pause signal to Unreal Engine. If the user plays a key, MIDI OX will receive the input and duplicate the output, sending the signal through the two loop MIDI ports to Unreal Engine and Cubase. Unreal Engine will use this information to determine if the played note is correct or incorrect. Cubase will receive the note to play the corresponding sound. Unreal Engine will decide if the note is correct and send this information to the game system to add points. At the end, Unreal Engine should send information to check if the user wins or

loses and communicate this to the user. The following graphic shows the connection between different actors in the game.

For the games presented in the application, Levels 1 and 2 follow this connection sequence. In the Loop section, the application runs the sequence at every moment for the games.

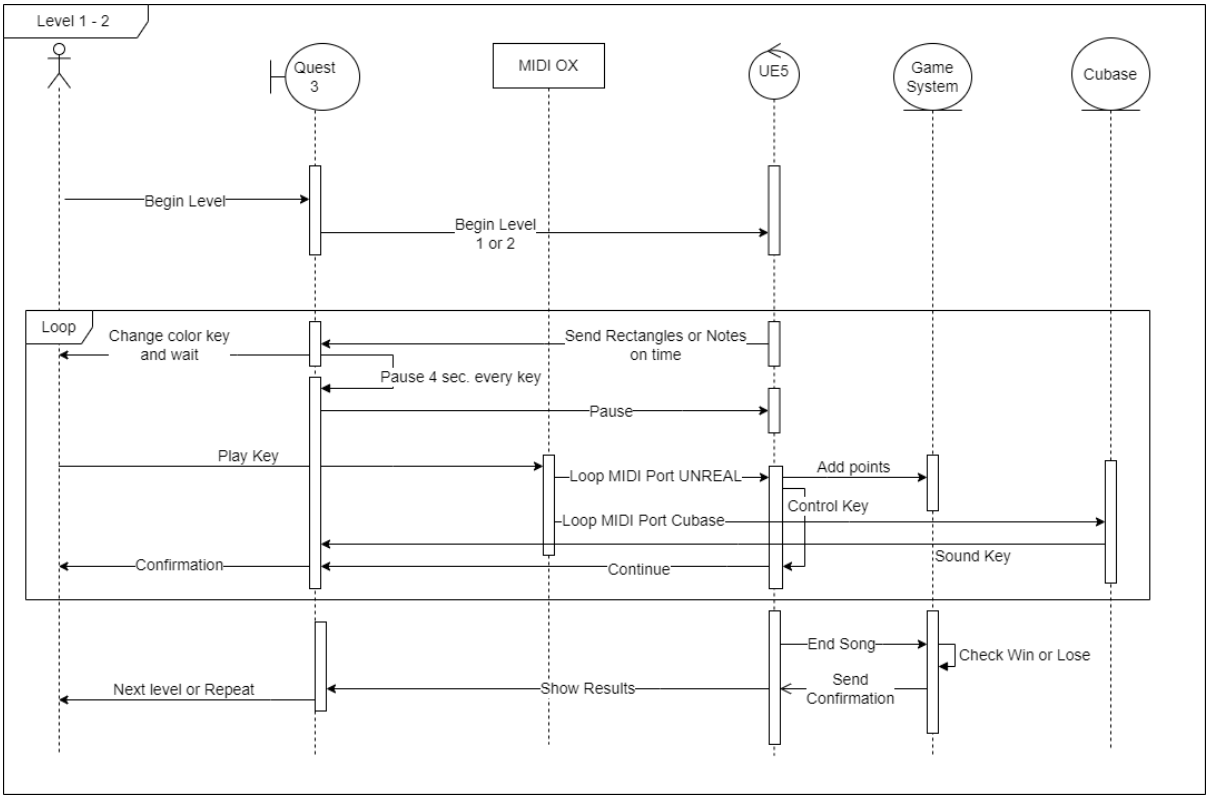

Figure 18 Level 1, level 2 Sequence Diagram
Source: Own creation

The following graphic illustrates the functionality of Play Level 1. It depicts the interaction between BP_Notes and BP_GameSystem and explains the creation process of BP_Rectangle. Key aspects include the game's behavior during pause and unpause states, as well as win and lose conditions, and the process of note creation.

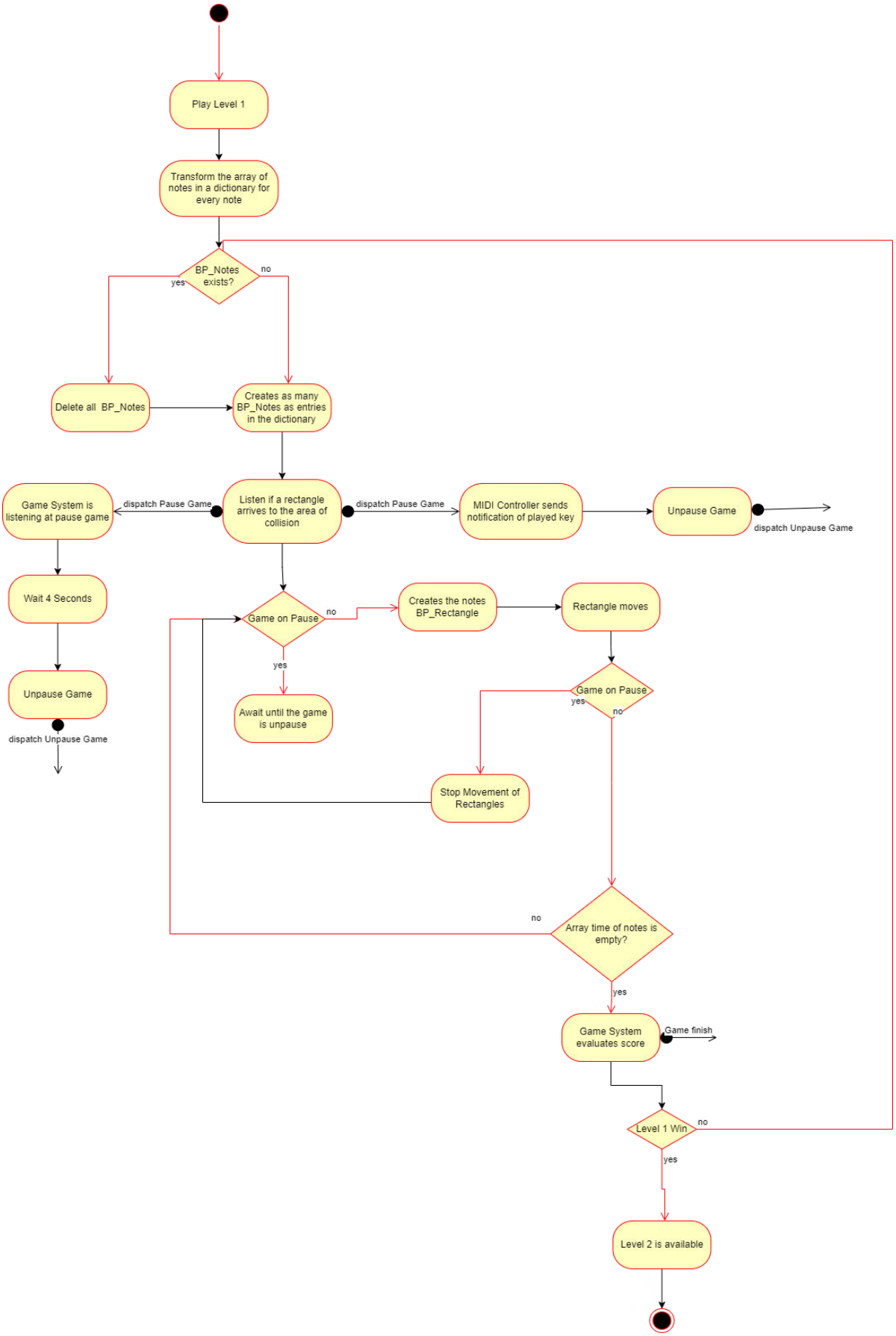

Figure 19 Level 1 Activity Diagram
Source: Own creation

Level 2 – Game with Notes and Music Sheet

The process in the second level is similar to that of Level 1, but there are fewer steps to create notes.

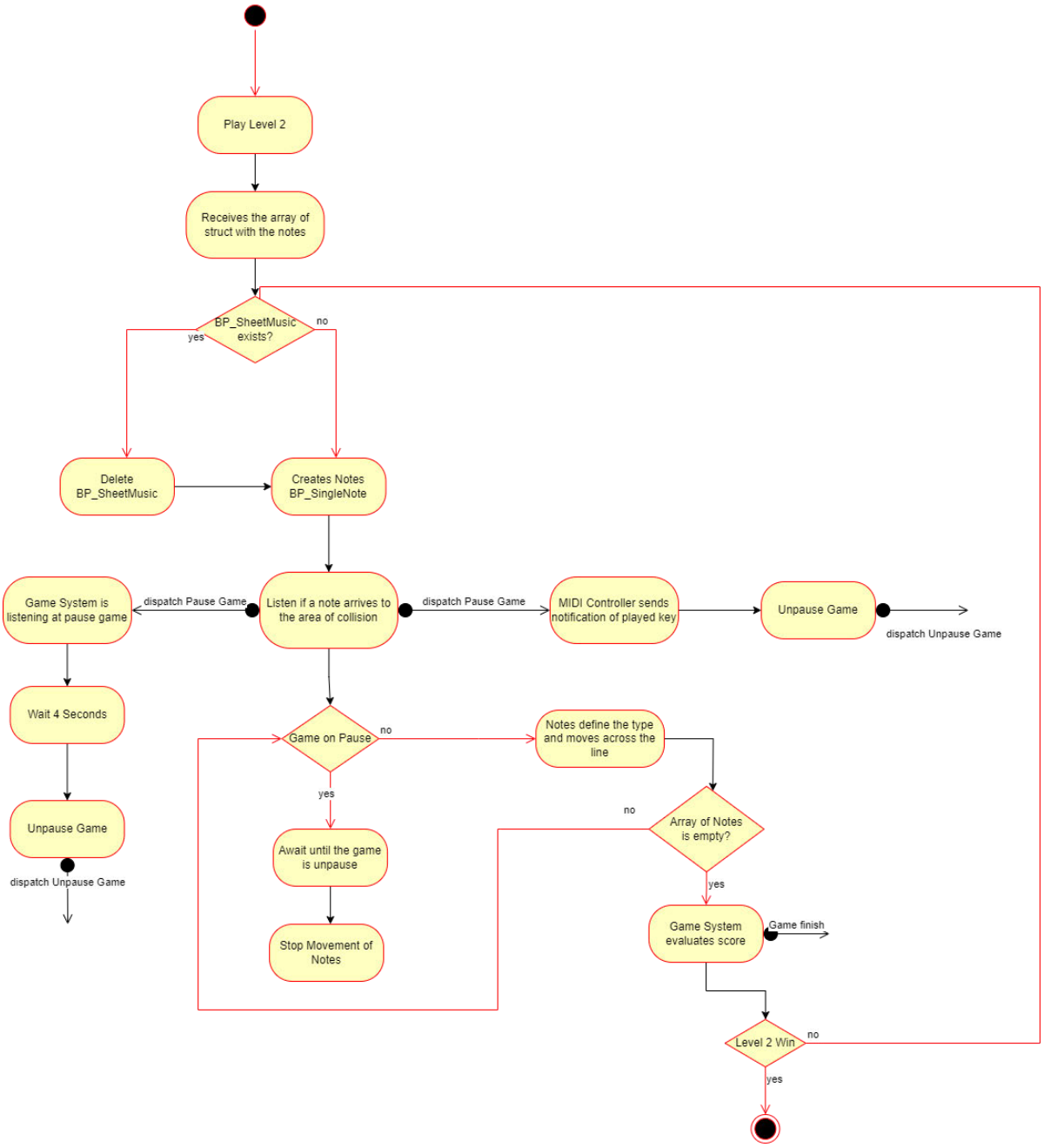

Figure 20 Level 2 Activity Diagram
Source: Own creation

Creation of Notes and Rectangles

As soon as BP_KeysKeyboard is spawned in the game, it will activate BP_OSC to send a message through the OSC connection to Cubase. Cubase will respond with a string containing all the notes, times, and durations created in a track named “Unreal.” When the response arrives, BP_OSC will dispatch a “Message Received” signal to BP_KeysKeyboard. BP_KeysKeyboard will listen for this signal and send it to AC_SetNotesToString to convert the string into an array of notes. SetNotesToString will process

the message into a list of notes and convert it into an array of struct notes. This array is then sent to BP_PanelNotes or BP_SheetMusic.

BP_PanelNotes separates and classifies all the notes from the array of struct notes, creating a dictionary with each note as a key and an array of floats for the values of time note and duration. TimeNote indicates when the note will be spawned, and duration indicates how long the note is played. BP_PanelNotes then creates a BP_Notes for each note in the respective position on the keyboard and sends the array of times and durations. BP_PanelNotes contains an actor component that creates BP_Notes, which uses the timing information to spawn the blocks. The duration is sent to BP_Rectangle to determine the length of the object. BP_Notes contains an actor component AC_ObjectPool, which creates all the objects in the background as inactive and invisible.

BP_Rectangle needs to be a child of BP_PoolActor to be recognized as an object to be spawned by AC_ObjectPool.

BP_SheetMusic is created when the user selects level 2. It receives an array of StructOfNotes and contains a mesh with lines for the notes, a treble clef, and a bass clef. This blueprint reads the timing for each note, selects the respective line for that note, and calls the BP_SingleNote object, sending the duration to determine the type of note. BP_SingleNote contains the actor component AC_ObjectPool, which creates the BP_SingleNote objects as inactive and invisible.

BP_SingleNote needs to be a child of BP_PoolActor to be recognized as an object to be spawned by AC_ObjectPool.

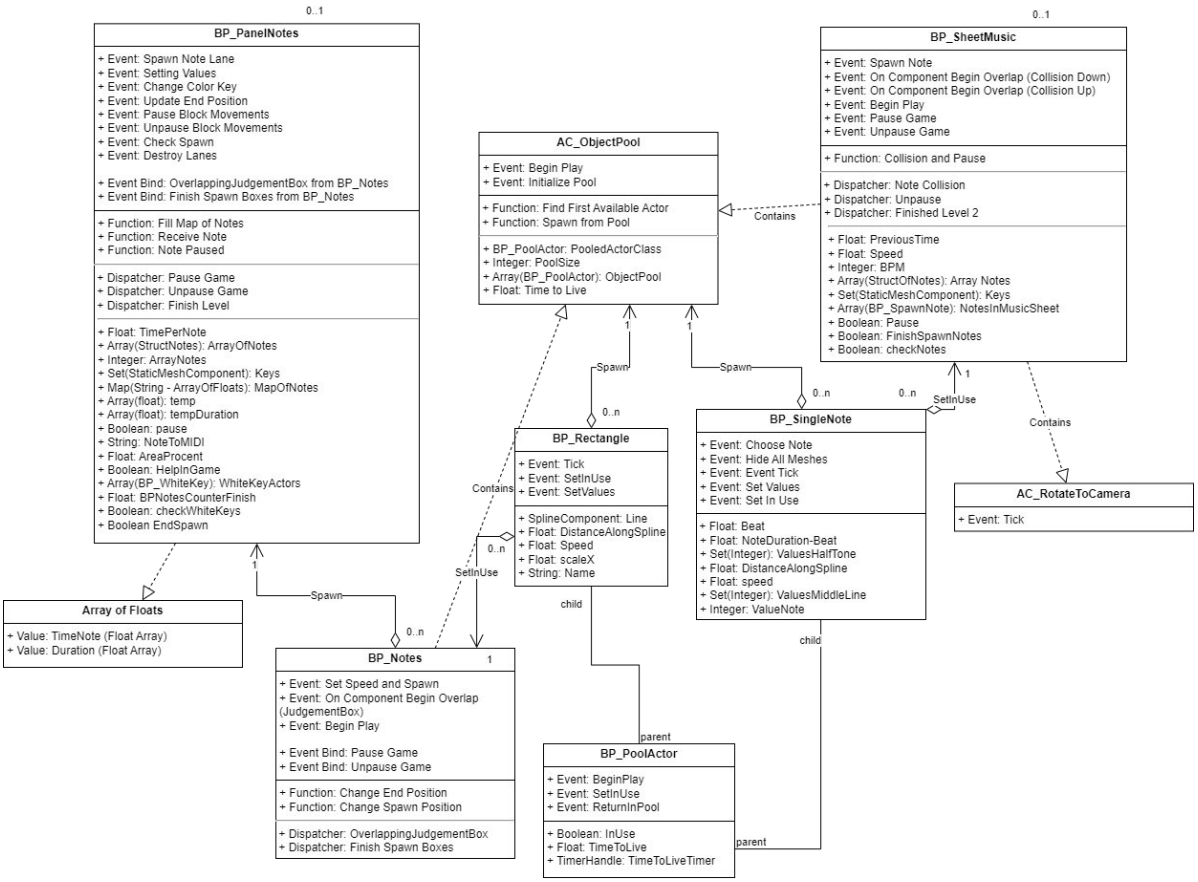

Figure 21 Creation Notes and Rectangles Part 1 Class Diagram
Source: Own creation

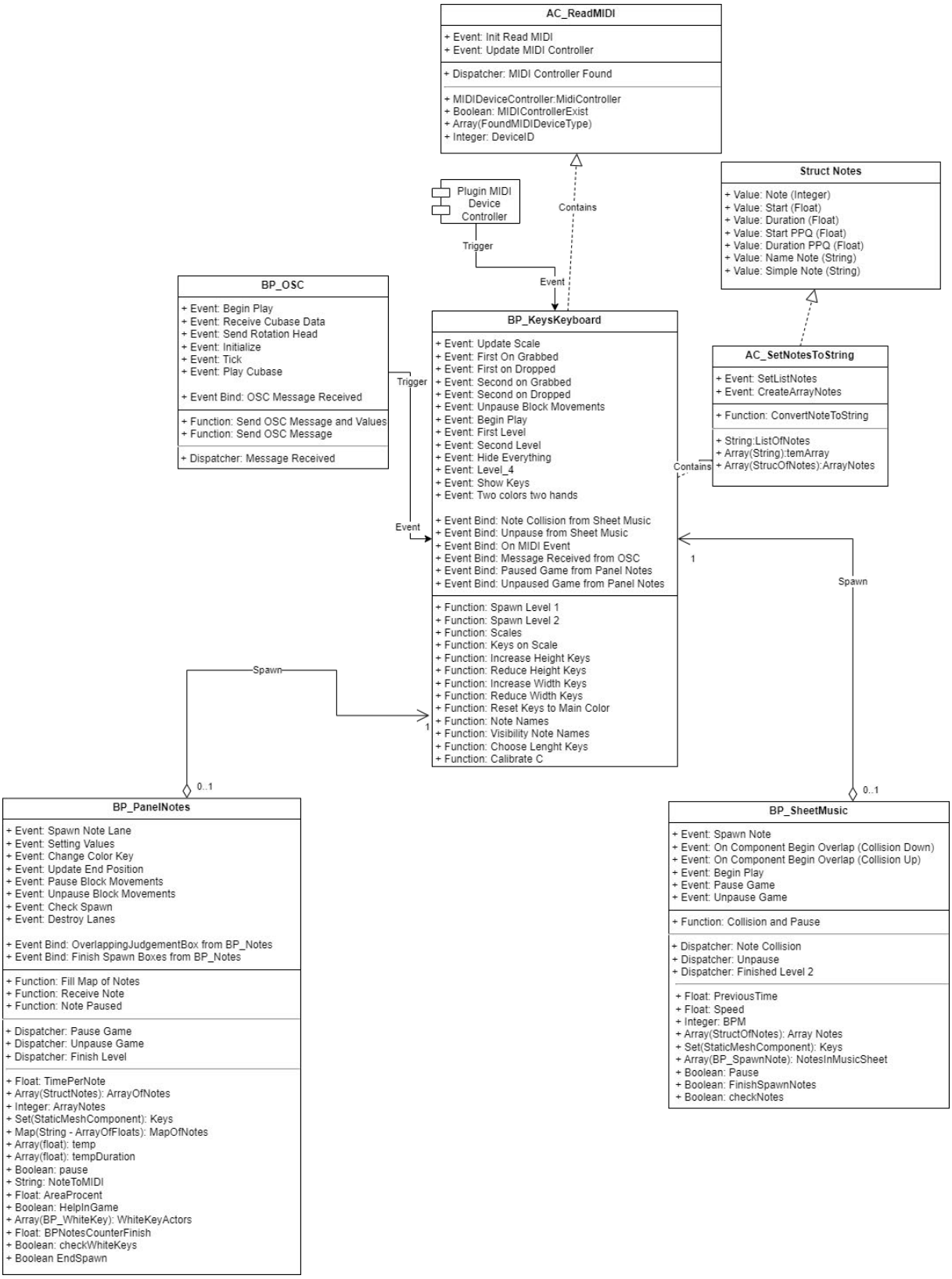

Figure 22 Creation Notes and Rectangles Part 2 Class Diagram
Source: Own creation

Notes Collision

When the notes are running and collide at the desired point, it is important to send a notification to other components. For this, the Dispatcher system by Unreal Engine will be used.

The following diagram shows the connection for the blueprint “Keys Only.” It listens to different blueprints operate at the proper moment. BP_Notes sends a notification every time a block arrives and collides with an object called Judgement Box. This notification is sent through the Dispatcher “Overlapping Judgement Box.” BP_PanelNotes listens to this dispatcher and stops the movement of the boxes and the creation of new boxes for a period of 4 seconds or until the correct note is pressed on the MIDI controller. At that moment, it sends a notification of Paused Game to other blueprints. BP_Notes receives this notification and stops the creation of new notes until it receives another notification to Unpause Game. Additionally, BP_Notes sends a notification called Finish Spawn Boxes when there are no more notes to activate, indicating that the dictionary of times is empty for that note.

BP_KeysKeyboard listens to events from the MIDI Device Controller plugin, which monitors the inputs from the MIDI controller. If the correct MIDI controller port is selected, the plugin reads the values generated by the user on the MIDI controller. For example, if the user plays key 60 on the MIDI controller, the plugin triggers an event indicating that the note was played. BP_KeysKeyboard converts this to a C4 note. If the game is paused, it compares the played note with the note waiting to be played to unpause the game.

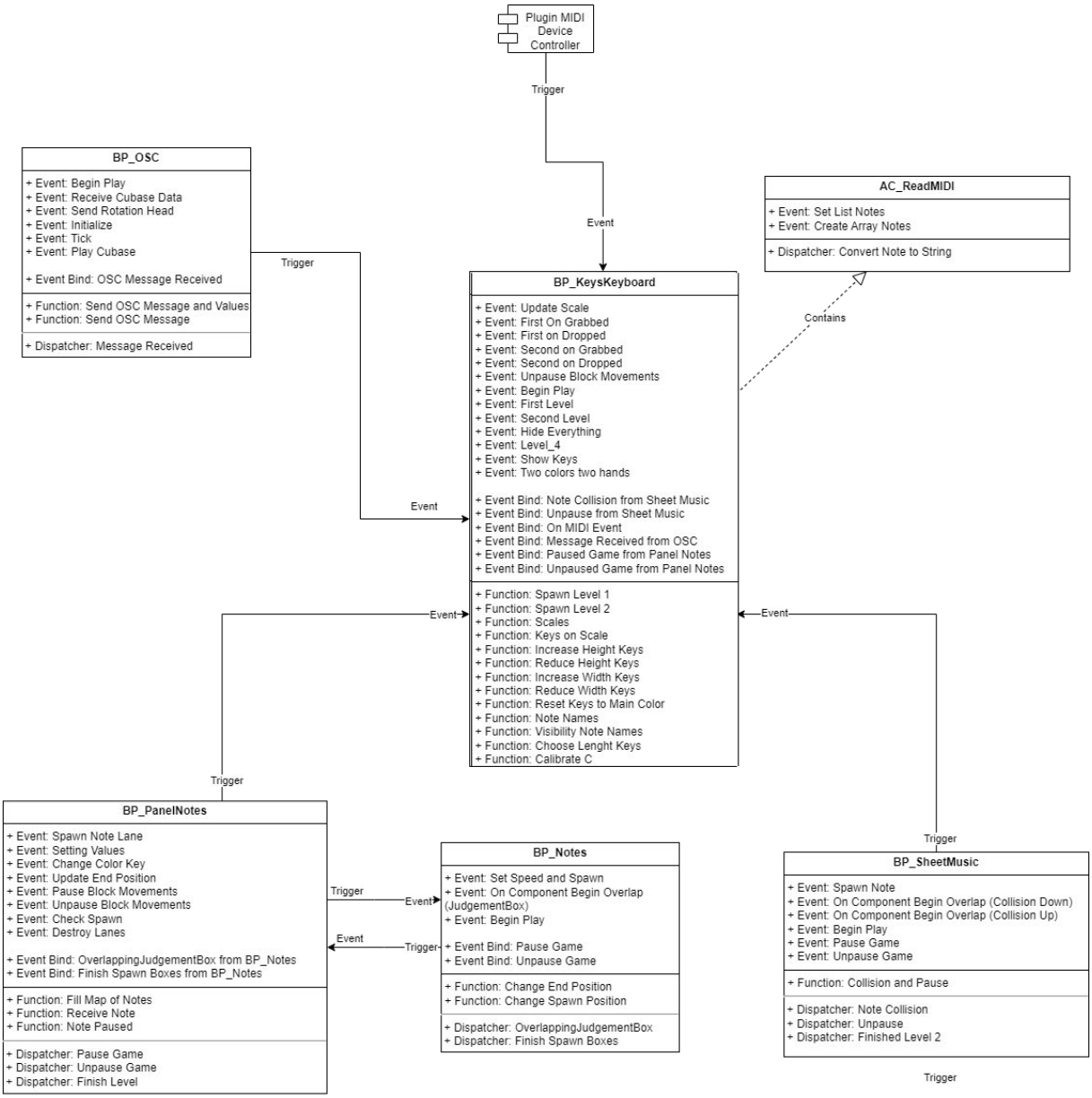

Figure 23 Notes Collision Class Diagram
Source: Own creation

BP_GameSystem is active and listens to BP_PanelNotes and BP_SheetMusic. When the game is paused, it counts down from four seconds to unpause the game. This blueprint calculates the points that the user earns and then sends notifications to WBP_Timer and WBP_First. WBP_Timer receives the points the user has accumulated and announces whether the user has won or lost the game. WBP_First receives the win notification and either activates the next level or, in the case of a loss, prompts the user to repeat the same level.

Figure 24 Notes Collision and Game System Class Diagram
Source: Own creation

Tutorial

In the initial moments of the game, BP_Robot and BP_UMG Display are created. They are spawned through AC_VRManager, which uses the room information and headset data to determine the spawn points for both objects. BP_Robot has multiple functions to demonstrate in the tutorial. It manages tutorials for major scales, minor scales, and chords, and it also controls BP_TutorialKeyboard while listening to that blueprint and reacting to triggers. The robot contains a blueprint called BP_UMG Robot Dialog, which includes a widget component called WBP_DialogRobot.

BP_TutorialKeyboard provides explanations of notes on a keyboard, creating tables, moving notes, and more. The robot and tutorial keyboard are the main actors in the tutorial level.

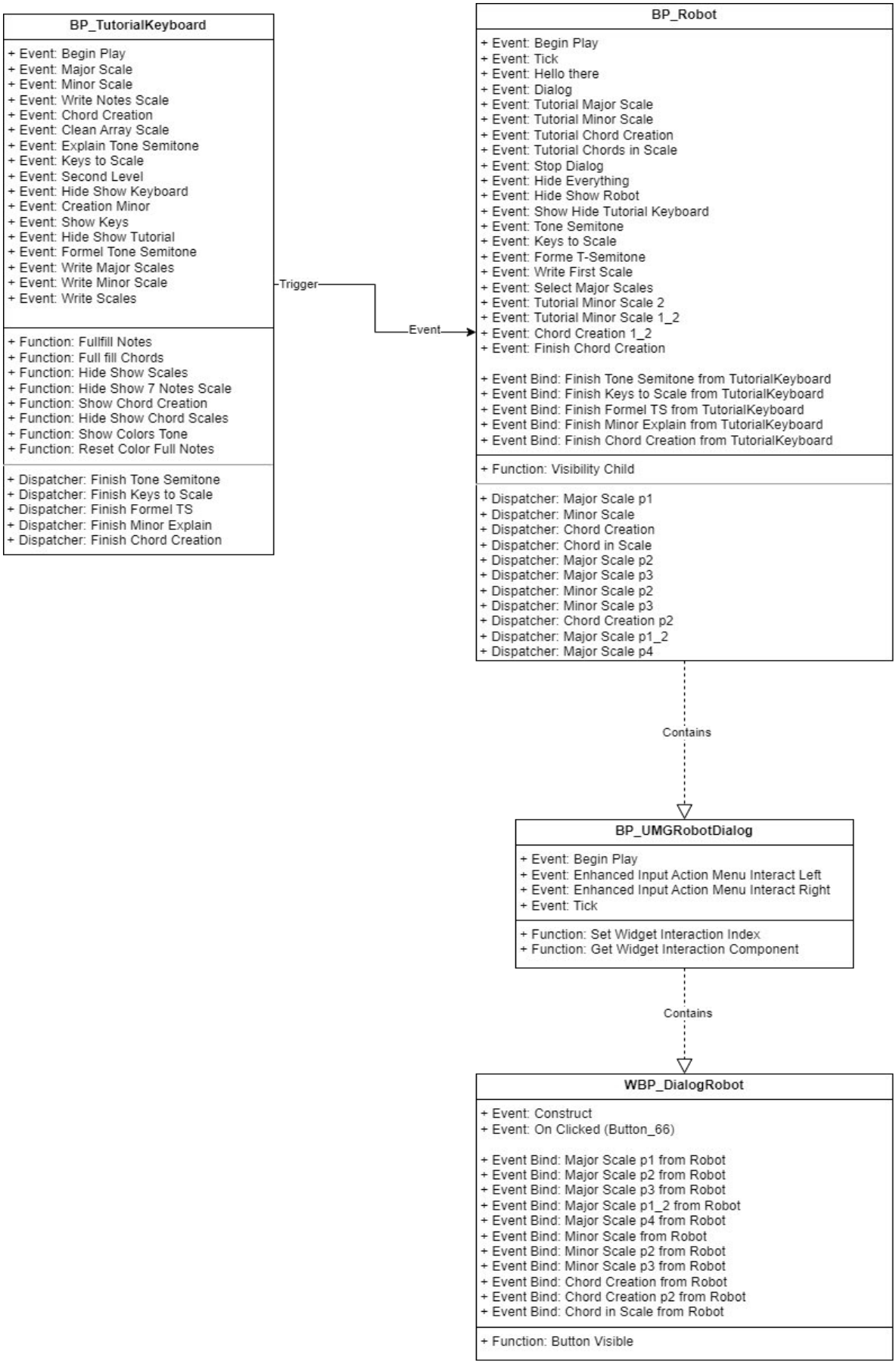

Figure 25 Tutorial Class Diagram
Source: Own creation

5 Validation

5.1 Technical Validation

The first step in technical validation is to ensure the game system operates correctly. The player should see the virtual keyboard and have their real keyboard recognized by the game. Hearing the notes from the keyboard confirms a successful connection with Cubase, allowing interaction with virtual objects and indicating that all game components have loaded properly.

Validating these elements is essential for assessing functionality. The graphics are designed to run at 90 Hz, providing a smooth experience. However, if the game occasionally shows slower movement, this may be due to hardware limitations. To investigate, we will use the profiler from the Meta Developer Hub to analyze performance metrics and identify any bottlenecks.

Ensuring that the graphics consistently reflect the intended frequency and fluidity is crucial for an engaging user experience. By validating these components thoroughly, we can enhance the game's performance and reliability, ultimately leading to a more enjoyable experience for players.

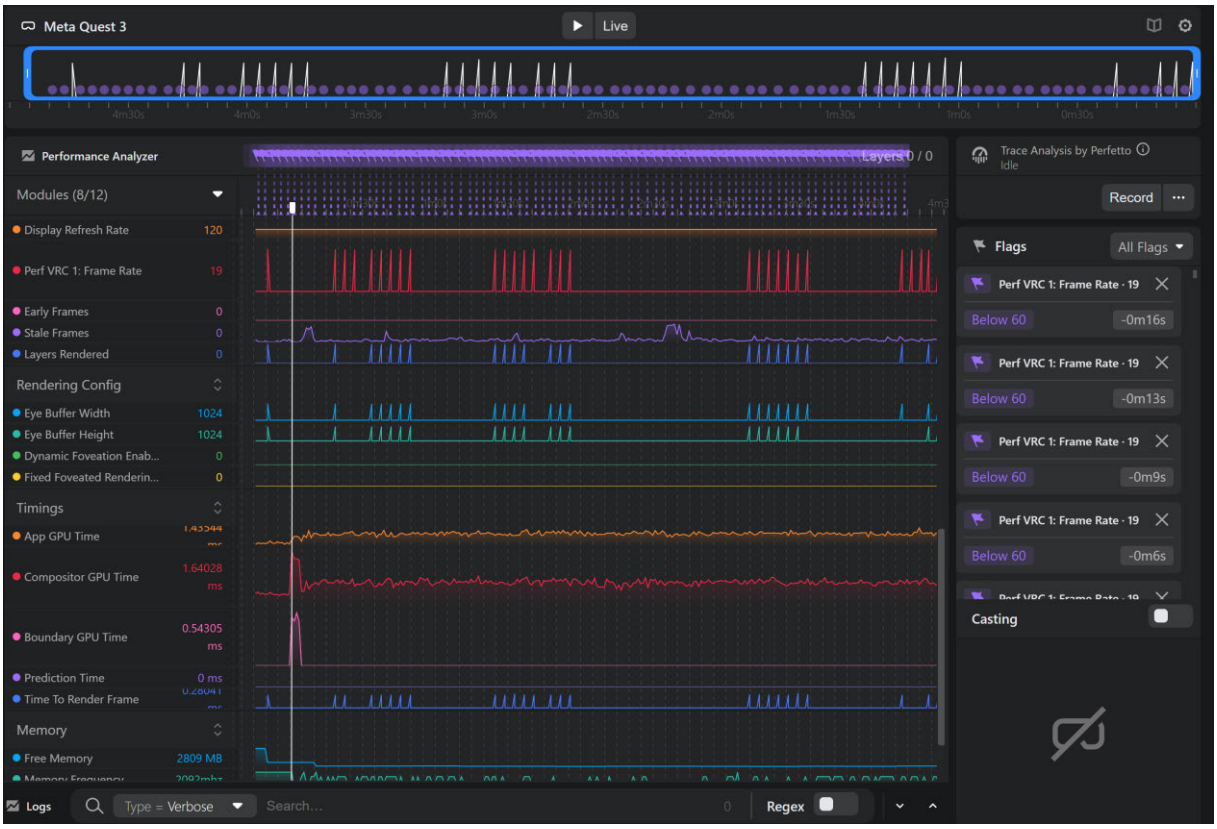

Figure 26 Graphic Profiler running Tone Tutor
Source: Meta Quest Developer Hub

Figure 27 Memory Profiler running Tone Tutor
Source: Meta Quest Developer Hub

Using the profiler, it is possible to observe some peaks in GPU time at the beginning of the game. Other peaks are seen in the frames, but these do not cause any invalidating problems. The game continues to run smoothly. Figure 23 shows that memory usage increases, which is a point to consider for future optimization. Currently, the game is running well without any issues for the player. There is an unidentified problem at the beginning of the game, which will need further investigation in future versions.

5.2 Game Concept Validation

The game was evaluated with 16 different participants: 7 without experience using a Virtual Reality headset, 5 with little experience, and 2 with more experience. One extra point to add that the participants were significantly different ages in a range approx. between 11 and 50 years old.

5.2.1 Hypotheses Validation

Willingness to play the game:

In the first section, participants evaluated their enjoyment of the game at each level. The questionnaire provided five options for participants to rate their level of enjoyment. The results are as follows:

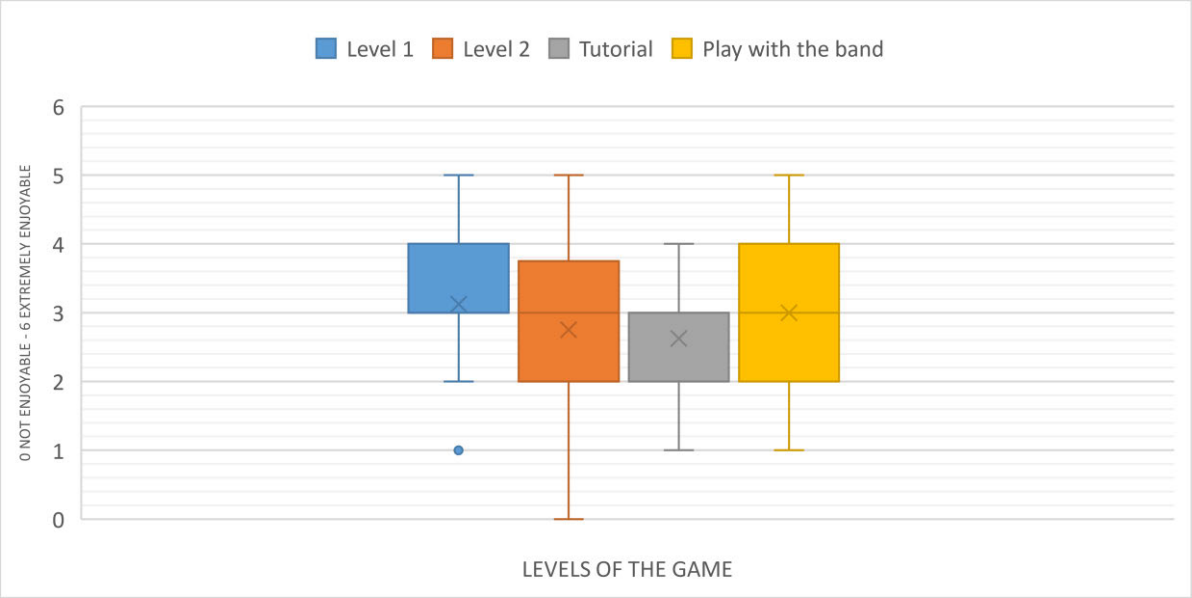

Figure 28 Enjoyment Analysis Results
Source: Own creation

The results indicated a moderate level of enjoyment, as illustrated in the graphic. The difference between each level is minimal, with almost every level staying in the range of 3, indicating moderate enjoyment. However, there is a slight tendency for more enjoyment in level 1, which was also noted by some participants.

The graphical analysis makes it possible to see the tendencies for each level. Even though the mean value is 3 (moderately enjoyable) for most levels, level 1 shows a tendency to be more enjoyable. The tendency for level 2 is harder to discern, but with a mean value of less than 3, it is classified as not enjoyable. The tutorial has the same tendency as level 2, with a mean under 3, and is also classified as not enjoyable. Level 4, "play with the band," is exactly in the middle, with moderate enjoyment, and is not considered either positively or negatively enjoyable.

To answer the hypotheses about enjoyment, with two responses classified as not enjoyable, the following hypotheses will be valid:

Ha1: The challenges presented at each level do not increase the overall enjoyment of the game.

The excitement to continue with the musical journey with this game showed:

Figure 29 Analysis Excitement for Music Learning
Source: Own creation

In order to ascertain the participants' commitment to pursuing musical education, their motivation to learn music theory was evaluated. The participants responded as follows: five indicated a slight excitement, six indicated a moderate excitement, and four indicated a high level of excitement.

A rating of "moderately excited" can be considered a positive response. The addition of the "very excited" responses brings the total number of positive responses to 10, which is an encouraging result in terms of measuring excitement following the musical journey. Accordingly, the following hypothesis can be validated:

Hb0: Users are excited to continue their musical journey with more lessons in this game.

To determine if participants were motivated to learn music after the game, the following graphic provides a good visualization of the responses.

Figure 30 Motivation to keep Playing
Source: Own creation

As shown, 11 participants were somewhat motivated, and 2 were totally motivated. This indicates that the project successfully encouraged people to learn more about music. The provided answers will help validate the hypothesis (H0) regarding motivation:

Hc0: After completing the game, users are motivated to gain further insight into musical concepts.

Considering all the hypotheses (H0 and H1) for enjoyment, excitement, and motivation, there are two H0 responses indicating positive outcomes for these factors, and one H1 response indicating a negative

outcome for enjoyment. Overall, the responses suggest a strong tendency to accept that the game increases the willingness to play.

Functionality of the game:

The Simulator Sickness Questionnaire (Bimberg et al., 2020) provided a good overview of the application's usability. Although this test was developed for virtual reality applications and may be somewhat unfair to use in a mixed reality context, it can still offer valuable insights about the game.

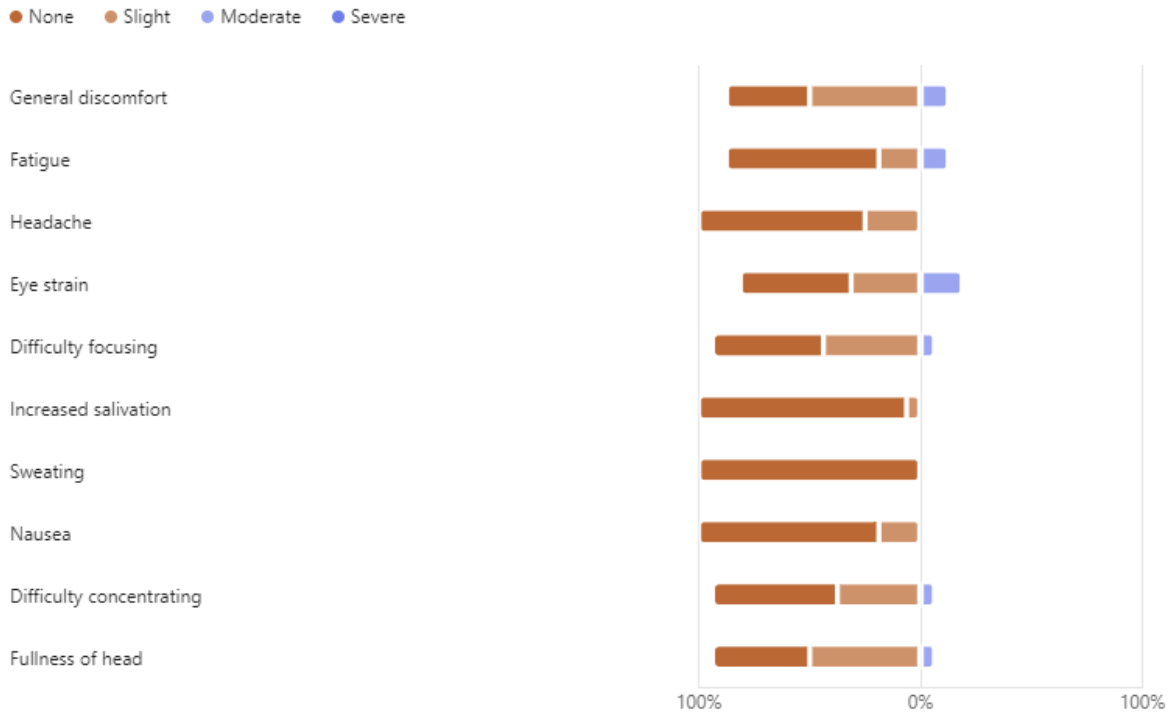

Figure 31 Sickness Questionnaire Results
Source: Own creation

During the test, participants reported experiencing **no discomfort to slight discomfort**, indicating that the mixed reality game is generally comfortable. However, to improve the experience and reduce discomfort to almost none, issues such as 'general discomfort', 'eye strain', 'difficulty concentrating', 'difficulty focusing', and a feeling of 'fullness in the head' need to be addressed. These issues seem related to problems in the test, such as the drift of virtual assets and the slow movement of the headset. Installing the game on the headset could help by utilizing the headset's GPU for faster rendering times. Additionally, a better UMG display and implementation of stereo layers could reduce discomfort. With that data, an answer will be provided to validate the following hypothesis:

H0: The use of the mixed reality game feels comfortable during use.

Another area for improvement is the spawning of the virtual keyboard in the real world. Participants found it challenging to set up the game, with 5 points indicating "somewhat not easy" and 6 points as "neutral", the graphic shows the distribution of the answers.

Figure 32 Setup Game Usability Diagram
Source: Own creation

Based on the percentage of responses, it can validate that setting up the keyboard, spawning in the real world, and selecting the MIDI controller were difficult tasks for 31% of the participants, while 38% viewed the task as neutral.

Given this distribution, the responses lean more towards the game not being easy to set up and use, with a significant portion of participants either neutral or finding it somewhat difficult. Therefore, it would be reasonable to reject the null hypothesis (H0) and accept the alternative hypothesis:

H1: The game is not easy to set up and use.

Exploring other ways to spawn the virtual keyboard will be necessary, and these ideas will be discussed in the next section.

Learning outcomes:

Another question was whether players learned something new during the game test. Most participants said they did **not learn anything new**, but **5** participants mentioned **learning something**, especially in the first level. Based on the feedback, it is clear that the tutorial part needs improvement. Participants need control over the speed of text transitions in the tutorial, and it should be more interactive, playful, and include more sounds.

H1: Participants do not learn to identify the C key or C scale after playing the game.

To finish, this app can be an engaging way to learn music theory. However, the full potential of Cubase to change the sound of the keyboard was not utilized during the test. While Cubase is a good solution for importing MIDI notes for the project, it may be too advanced for new users who are already over-stimulated by the new experiences during the game.

Emotional Response Validation:

After the analysis, an emotional validation of the game will help confirm player engagement. A qualitative analysis will interpret and summarize participant responses, classifying them to create a backlog of ideas and formulate conclusions.

First, it is important to understand the emotions experienced during the game. Participants were asked about their emotions during gameplay. Responses were categorized into three sections: **negative**, **neutral**, and **positive** emotions.

Table 1 Emotion Analysis
Source: Own creation

Participants	Answers
3	No answer
3	Negative emotions
6	Neutral emotions
4	Positive emotions
Main neutral emotion: Confusion	

Participants' comments were collected during the test. Among the various dynamics and concepts presented, the sequence of having users play easy games first, followed by the music tutorial, caused more confusion than other parts of the game. Participants suggested that it would be more effective to present the music tutorial first, accompanied by exercises based on the games, to enhance understanding. This is an important consideration for future versions. A new approach and additional suggestions will be detailed in the next section

Participants highlighted the most **liked aspects** of the game: the robot, the overlay keyboard over the real keyboard, the concept of a music sheet, the creation of chords, and the grabbing of objects. They also pointed out **dislikes**: the visual elements of the tutorial, the user interaction with the menu, blurry buttons and text, and the navigation UX, which could be improved to help navigate the lessons.

The majority found it to be an interesting and promising technology for learning. This suggests that, with some bug fixes and design adjustments based on feedback, this learning method has potential for the future.

5.2.2 Feedback and suggestions

During the test and questionnaire, participants provided feedback and suggestions about their experience using the game. These responses were classified, with some being repeated or indicating similar feedback. Here is a list of the feedback received:

1. **Performance and Visualization**:

- Make the menu static, so it does not always follow the camera, improving visualization.
- Configure the menu with stereo layer visualization to enhance clarity, while addressing collision issues with other objects.
- Allow the menu to be visible or hidden at times.

2. **Learning Interface**:
   - Reduce distractions in the tutorial by spawning relevant actors directly in front of the headset to maintain user focus.
   - Allow manual progression through the robot's tutorial texts, as reading speeds vary.
3. **UX Performance**:
   - Improve the menu over the robot and the dialog display, as they currently interfere with the main menu layout.
   - Implement feedback for objects when grabbed with one or two hands.
   - Highlight tutorial elements when active to capture user attention.
4. **Interactivity**:
   - Make interactions with the robot and tutorial keyboard more engaging. Ensure they respect the player's view by spawning close to the current point of view.
   - Consider adding voice to the robot.
   - The robot provides instructions on navigation, menu options, and object interactions.
5. **Training Session**:
   - Include a training session to familiarize users with the game interface.
6. **Level Fusion**:
   - Combine Level 1 and Level 2, where blocks fly to positions on the music sheet, allowing players to catch the notes they are playing.
7. **Tutorial Organization**:
   - Consult a music professor to organize the tutorials, especially for note reading and understanding musical concepts like whole notes and half notes. Adopt a pedagogical approach to teaching music.
8. **Robot Instructions**:
   - The robot provides instructions on navigation, menu options, and object interactions.

9. **Audio Enhancements**:
    - Add background music and other sounds, such as scale sounds.
10. **Pattern Training**:
    - In Level 1, include a pattern on the keyboard for players to follow, helping to train muscle memory, especially for beginners.
11. **Keyboard Visualization**:
    - Improve the colors of the keyboard for better visualization of the black keys.
12. **Feedback Mechanism**:
    - Provide feedback when a key is played correctly and at the right time.
13. **Difficulty Levels**:
    - Beginner: Use the same formula as in the project to indicate a win or a loss. This seems straightforward and appropriate for beginners.
    - Advanced: Require more than 80% accuracy or repeat the level. This adds a good challenge for advanced users.
14. **Level 2 Adjustments**:
    - Slow down Level 2 for better note visualization. Use a static white paper with static notes and a moving color bar to indicate when to play the note.
15. **Manual Tutorial Progression**:
    - Allow manual progression through the robot's tutorial texts, as reading speeds vary.
16. **Virtual Hands**:
    - Implement virtual hands on the virtual keyboard to help users position their hands correctly.
17. **Chord and Progression Automation**:
    - Enable automatic chord and progression playback by playing a single note on the keyboard, similar to Cubase's chord track.
18. **Increase fluency of rendering**:
    - Develop an installable version of the game for the headset, with the MIDI controller connected to the Quest 3.
    - Use meta sounds to create the sound of the notes.

19. **Virtual Keyboard Spawning**:
    - Use markers on the first and last keys of the keyboard to select the number of keys. The game should spawn the keyboard accordingly.
    - Include buttons near the virtual keyboard to manually adjust the height and size in small increments.

By addressing these specific issues, our objective is to enhance the overall user experience, making the game more comfortable and enjoyable. Furthermore, incorporating these suggestions will enable us to refine the game's design and functionality, ensuring that it meets the needs and expectations of our users.

5.3 Conclusion Game Concept

In conclusion, after analyzing and validating the main hypothesis, this application has demonstrated an increase in enjoyment, motivation, and excitement for music theory. While some participants expressed a desire to learn more music theory, others did not. However, all participants wished to continue the development of this app to further their music theory education. The questionnaire provided numerous points and ideas for improvement in a new version of the project.

Goals of the project:

1. **Functionality:** The game was not easy to set up, but it was comfortable to play, and players did not experience sickness. The connection between the components was successful, and the FPS test was good. Although drift was present, it did not invalidate the gameplay and can be resolved by following the proposed recommendations.
2. **Learning Outcomes:** Users did not learn something new with the game, but the feedback provided better insights for future consideration, ensuring this goal will be achieved. Feedback indicated that the idea needs a redesign with pedagogical supervision.
3. **User Engagement:** The project is enjoyable and exciting to play, generates motivation to learn music, and provides an immersive experience. It offers an engaging experience for learning music theory. Feedback indicated that the idea is promising but requires further work.
4. **Feedback:** The project was able to collect many ideas to consider and identify the main points that impact the player the most.

5.4 Future Work

The feedback provides a good overview for future versions and further analysis. A recommendation is to conduct a preliminary test of ideas in a simple way, without many visual enhancements. For example, using basic 3D models like a cube. Additionally, it would be beneficial to try an installable version on

the headset, without accessing it through the computer. This would likely require using a different system for audio instead of Cubase, for example Meta Sounds from Unreal Engine. The MIDI controller could be connected to the headset and should be recognized, but this needs to be tested.

For configuring the keyboard, it could be useful to try dynamically generating the keys and automatically positioning them using markers recognized in AR applications.

For the next levels, it would be better to take each option of the tutorial and begin with an introduction to the theoretical part. It might be possible to reuse the robot to explain major scales, providing examples, and allowing the user to play the scale, with penalties for hitting the wrong key. After that, the player could play an easy song using the block system game of level 1. This process can be repeated for minor scales and chord creation. Following this, it would be possible to see if using a music sheet with an explanation can be clear and enjoyable. Ideally, a preliminary test for each of these ideas would be beneficial to ensure the right balance of fun and challenge during the learning process.

6 Summary

To summarize the project, it is necessary to cite the problem that was presented: Many individuals, after long hours of work or study, lack the motivation to learn new skills, such as music theory. This subject is particularly abstract and requires a solution that maintains learner motivation.

As a solution, this project created a game using concepts of gamification to teach music theory. A virtual teacher, represented by a robot “Tone Tutor”, provided instruction in a mixed reality scenario. Gamification was implemented through a narrative and a points system to create challenges for the player. The robot delivered all the information, replacing a written tutorial, and the game offered two different modes, challenging the user with the points system. After the game's development, an analysis demonstrated promising concepts, and feedback was collected.

Objectives of the Project:

1. **Functionality**: Although the game setup was complex, it was comfortable to play, and the connection between components was successful.
2. **Learning Outcomes**: Users did not learn new content from the game, but the feedback provided valuable insights for future improvements.
3. **User Engagement**: The game provided an engaging experience, enhancing the willingness to play and promoting positive emotions towards learning music theory.
4. **Feedback**: The project collected numerous ideas for consideration.

A qualitative analysis was conducted to understand participants' emotions towards the game, and a quantitative analysis measured engagement, enjoyment, motivation to learn music, and excitement to continue using the application. Additionally, a VR sickness test was conducted to assess comfort in a mixed reality game, and a technical test evaluated functionality.

Challenges and Improvements:

- Some problems were identified, particularly with aligning the virtual keyboard over the real MIDI controller. This feedback is crucial for future versions.
- An installable version of the game on the headset could improve fluency and user experience.
- The project addressed the challenge of maintaining motivation for learning abstract subjects like music theory, especially for individuals with limited time and energy after work or studies.

This project presents a promising approach to addressing the challenges of music theory education through gamification. The feedback collected is an essential part of the project, highlighting areas for future development and testing. While gamification can be a powerful tool for learning, further testing is needed to refine each idea and identify the most motivating solutions within the game.

List of References

Alexandru Ioan Cuza University of Iasi, Simion, A., Iftene, A., Alexandru Ioan Cuza University of Iasi, Gîfu, D., & Alexandru Ioan Cuza University of Iasi. (2021). An Augmented Reality Piano Learning Tool. *RoCHI - International Conference on Human-Computer Interaction*, 134–141. https://doi.org/10.37789/rochi.2021.1.1.21

Deep Clear Eyes (Director). (2017, January 27). *beatmania THE FINAL (Arcade / 2002)—Gameplay* [Video recording]. https://www.youtube.com/watch?v=ZAVHJDB8ZUk. Retrieved October 24, 2024

Hackl, D., & Anthes, C. (2017). *HoloKeys—An Augmented Reality Application for Learning the Piano*. Forum Media Technology. https://www.semanticscholar.org/paper/HoloKeys-An-Augmented-Reality-Application-for-the-Hackl-Anthes/8760b305977d852e92eb426ab041517e05e3a766

Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert Scale: Explored and Explained. *British Journal of Applied Science & Technology*, *7*, 396–403. https://doi.org/10.9734/BJAST/2015/14975

Lim, K. Y. T., Truong, K. M., & Wu, Y. (2023). Melody Mystery: Learning Music Theory through Escape Room Puzzles. *Education Sciences*, *13*(5), Article 5. https://doi.org/10.3390/educsci13050491

Lu, Y., Wang, X., Gong, J., & Liang, Y. (2022). ChordAR: An Educational AR Game Design for Children's Music Theory Learning. *Wireless Communications and Mobile Computing*, *2022*, 1–9. https://doi.org/10.1155/2022/5268586

Maqableh, W., Zraqou, J., Alnuaimi, A., & Al-Shurman, A. (2024). Adoption of Virtual Reality Technology in Learning Elementary of Music Theory to Enhance the Learning Outcomes of Students with Disabilities. *International Journal of Instruction*, *17*(3), 37–60. https://doi.org/10.29333/iji.2024.1733a

Marrington, M. (2010). Experiencing musical composition in the DAW: the software interface as mediator of the musical idea. *Proceedings of the 6th Art of Record Production Conference*

MIDI in Unreal Engine | Unreal Engine 5.5 Documentation | Epic Developer Community. (n.d.). Epic Games Developer. Retrieved October 24, 2024, from https://dev.epicgames.com/documentation/en-us/unreal-engine/midi-in-unreal-engine

OpenXR - High-performance access to AR and VR —collectively known as XR— platforms and devices. (2016, December 6). Retrieved October 24, 2024. The Khronos Group. https://www.khronos.org/OpenXR/

Panorama T4 and T6 ▷ MIDI Controller Keyboards | DAW Control | Plugin Control. (n.d.). *Nektar*. Retrieved November 21, 2024, from https://nektartech.com/panorama-t4-t6-controller/

Robert, D., Jamri, N., Ling, S., Amin, A., & Yazid, F. (2023). Gamified Learning Intervention to Promote Music Literacy and Creativity in Elementary Music Education. *Journal of Cognitive Sciences and Human Development*, *9*, 18–41. https://doi.org/10.33736/jcshd.5481.2023

Additional Resources Section

3D Models

Electric Guitar Explorer—Download Free 3D model by Skabl (@skabl_yt). (2021, December 26). [Video recording]. https://sketchfab.com/models/a7ffc570d3fe41c89b9dde195ab0faea/embed?autostart=1, Retrieved October 24, 2024

Guitar—Download Free 3D model by Ya (@Yarik16). (2021, August 9). [Video recording]. https://sketchfab.com/models/f8ccd75e8c2648ffbbc9e6208ed919c1/embed?autostart=1, Retrieved October 24, 2024

Percussion bass guitar—Download Free 3D model by Kanade_Tatibana. (2020, February 14). [Video recording]. https://sketchfab.com/models/d1f04fc920b8425a9a78b057b532dd89/embed?autostart=1, Retrieved October 24, 2024

Textures and Materials

Acoustic Foam 001 on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Brushedgold3.jpg (3500×2625). (n.d.). Retrieved November 21, 2024, from https://www.myfreetextures.com/wp-content/uploads/2011/06/brushedgold3.jpg

Fabric 025 on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Fabric 026 on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Leather 032 on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Metal 009 on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Metal 046 B on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Plastic 012 A on ambientCG. (n.d.). Retrieved November 21, 2024, from https://ambientCG.com/

Savva, D. (n.d.-a). *Seaworn Stone Tiles Texture • Poly Haven*. Poly Haven. Retrieved November 21, 2024, from https://polyhaven.com/a/seaworn_stone_tiles

Savva, D. (n.d.-b). *Worn Planks Texture • Poly Haven*. Poly Haven. Retrieved November 21, 2024, from https://polyhaven.com/a/worn_planks

Appendix

1 Hardware Requirements

1.1 Unreal Engine 5

Operating System

- Windows 10 64-bit version 1909 revision .1350 or higher
- Windows 10 versions 2004 and 20H2 revision .789 or higher

Processor

- Quad-core Intel or AMD, 2.5 GHz or faster

Video RAM

- 8 GB RAM

Graphics Card

- DirectX 11 or 12 compatible graphics card with the latest drivers

Optional: Lumen Requirements

- **Operating System**: Windows 10 build 1909.1350 and newer with DirectX 12 support
- **Project Settings**: SM6 must be enabled
- **Graphics Cards**:
    - NVIDIA RTX-2000 series or newer
    - AMD RX-6000 series or newer
    - Intel® Arc™ A-Series Graphics Cards or newer

1.2 Meta Quest Link Requirements

- **Processor**: Intel i5-4590 / AMD Ryzen 5 1500X or greater
- **Memory**: 8 GB+ RAM
- **Operating System**: Windows 10, Windows 11
- **USB Port**: 1 USB port

Graphics Cards

- NVIDIA GPU:
    - NVIDIA Titan X

- NVIDIA GeForce GTX 970
- NVIDIA GeForce GTX 1060 Desktop, 6 GB
- NVIDIA GeForce GTX 1070 (all)
- NVIDIA GeForce GTX 1080 (all)
- NVIDIA GeForce GTX 1650 Super
- NVIDIA GeForce GTX 1660
- NVIDIA GeForce GTX 1660 TI
- NVIDIA GeForce RTX 20-series (all except 2050)
- NVIDIA GeForce RTX 30-series*

NVIDIA GeForce RTX 40-series

1.3 Mixed Reality Headsets

In this project is exclusively done with Meta XR devices; other devices will not be considered in this phase. The decision to use Meta products is based on their cost-effectiveness compared to other options.

The second option for a good headset for mass consumption is the Pico series. The latest version, the Pico 4 Ultra, was released in October 2024 with a price of €599. While it offers a slightly better resolution and pixel density than the Meta Quest 3, this difference is not significant if apps are not optimized for those resolutions. Therefore, more important factors are the weight and memory capacity of the headsets.

- Pico 4 Ultra: Approximately 580 grams, with up to 256 GB of internal capacity and 12 GB of RAM.
- Meta Quest 3: Approximately 515 grams, with up to 512 GB of internal capacity.
- Meta Quest 3S: Approximately 514 grams.

Other devices, such as the Valve Index VR Kit and HP Reverb VR Headset, are excluded as they do not offer a mixed reality experience.

The prices for the Meta headsets are as follows:

- Meta Quest 3: €549
- Meta Quest 3S: €329
- Meta Quest Pro: Around €1000

2 Installation and Configuration

Meta XR is a plugin developed by Meta for creating apps for Meta Quest 2, 3, and Pro. It can be downloaded from the website of meta developers or from the marketplace (Fab) in Unreal Engine. **Open XR** comes natively in Unreal Engine, and it will be necessary to uninstall this plugin before installing Meta XR, at least in version 5.3.2**.**

Download and install the **OSC Plugin** for Unreal Engine from the Marketplace. It opens a port (server) to listen for incoming commands and allows the creation of a client to specify the port and IP address for sending information from the main program to other devices or programs, such as Cubase.

Download and install the **MIDI Device Support Plugin** for Unreal Engine from the Marketplace. This plugin manages the connection between the computer and other devices, typically a MIDI controller.

2.1 Configuration Meta XR

Unreal Engine version 5.3.2 has an issue with the integration of OpenXR, another plugin for creating VR experiences. Therefore, the first step is to uninstall all Unreal Engine plugins named OpenXR before downloading and installing Meta XR from the marketplace. These plugins can be found in the Plugins window in Unreal Engine, located under Edit > Plugins. Once they are uninstalled, Meta XR should be searched for on the marketplace and installed. After that, Unreal Engine will ask to restart the program to activate all the changes. The following programs are also needed for developing on Meta Quest 3:

- Android Studio Flamingo version 2022.2.1 Patch 2 (May 24, 2023)
- Java SE Development Kit 17.0.10 Windows x64
- Meta Quest Link
- Meta XR plugin version 65
- Meta Quest Developer Hub

2.2 Configuration Passthrough and Hand Tracking

1. **Open Project Settings** in Unreal Engine, navigate to Plugins, and find the Meta XR Plugin.
2. **Configure the following options**:

General

- **XR API**: Oculus OVRPlugin + OpenXR backend
- **Color Space**: P3

PC

- **Support Dash**: True

- **Composite Depth**: True
- **Meta XR Simulator JSON File**: Path to /MetaXRSimulator /meta_openxr_simulator.json

Mobile

- **Supported Meta Quest Devices**: Meta Quest 3 (select) and other Quest devices if needed
- **Composite Depth**: True
- **Hand Tracking Support**: Controllers and Hands
- **Hand Tracking Frequency**: High
- **Hand Tracking Version**: V2
- **Passthrough Enabled**: True
- **Anchor Support**: True
- **Scene Support**: True

Rendering

- **Postprocessing**: Enable alpha channel support in post processing (experimental) and select AllowThroughTonemapper from the combo box.

3 Difference between Unity, Unreal Engine and Godot

Unity

Unity is a cross-platform game engine that allows developers to create games and apps for various devices and platforms. It has excellent integration with Meta and other plugins for Extended Reality (XR) development. Unity uses C# as its primary programming language, offering dynamic development capabilities. This makes it easy to learn and implement various assets, including 3D designs and 2D designs, in both low and high quality graphics as other graphical effects.

Unity's integration with the Meta XR plugin has improved significantly with recent updates, sometimes even outpacing Unreal Engine. Unity also benefits from a larger community, providing extensive support and resources compared to other game engines.

However, configuring and supporting high-quality images in Unity can be challenging compared to Unreal Engine. Despite this, the final quality of the images in Unity is almost comparable to those in Unreal Engine. But not as dynamic as the graphics created on Unreal Engine.

Licensing

Unity faced significant issues last year, leading many users to stop developing on the platform. The controversy arose from a proposed licensing runtime fee, where Unity intended to take a certain amount

of money from products created with the engine after the company reached a specific revenue threshold. This idea was canceled in the last months of 2024, reverting to the traditional licensing model.

Currently, Unity offers a personal license that is free for products generating less than $200,000 per year. However, it is important to refer to the official Unity website for the most accurate and up-to-date licensing information, as interpretations may vary:

Effective January 1, 2025, individuals, hobbyists, and small businesses using Unity to provide services to others are eligible to use Unity Personal if their respective clients, in the aggregate, have less than $200K USD of revenue or funds raised in the prior 12 months.

Effective January 1, 2025 upon plan purchase or renewal:

- *Unity Pro will be required for businesses with revenue or funding greater than $200K USD in the last 12 months, and for those who do work with them.*
- *Unity Enterprise will be required for businesses with revenue or funding greater than $25 million USD in the last 12 months.*

Godot

Godot is a game engine developed in recent years, making it newer than Unity and Unreal Engine. Godot was initially developed for the creation of 2D games. The game engine itself is developed in C++, but developers can use GDScript (similar to Python), and in the last year, support for C# and C++ has been integrated.

With this tool, it is possible to create games for many platforms, including VR headsets. However, it primarily supports Virtual Reality programs and not Augmented Reality or Mixed Reality, which limits its use for other types of projects. Godot is also an open-source program under the MIT license, which means that all products developed with this tool must include this license and be open-source.

Unreal Engine

Unreal Engine (UE5) is used for the creation of high-quality games or simulations that require detailed environments, characters, and objects. It offers the latest innovations in the market for creating these elements. While Unity also allows for the creation of high-quality graphics, it is generally more challenging to implement compared to Unreal Engine.

UE5 is developed in C++ but also implements Blueprints, its own solution for new developers. Blueprints is a visual programming language that uses a node-based interface to create gameplay elements. One challenge is the vast amount of material available, which can sometimes overwhelm learners trying to figure out where to start. Additionally, forums and communities for Unreal are not as popular as those for Unity.

The fact that C++ is harder to learn often discourages students from learning it. Blueprints are an intelligent solution from Unreal Engine, but they require a new way of thinking about programming. The transition from traditional programming to Blueprints is not as smooth as Epic Games might hope. Visual Blueprints are indeed faster to learn and prototype with, and they can be used for many tasks within Unreal Engine. New plugins and packages are often first explained with Blueprints.

Professional software needs to be optimized for proper memory usage, which typically requires programming in C++. The game engine allows for C++ code in a project developed with Blueprints, and vice versa. However, this means developers need to learn both methods: first using Blueprints and then C++. Unreal Engine offers many tools to create impressive applications, but learning everything and choosing the right solution can take more time.

In this project, Blueprints will be used. The reason for this decision is that most tutorials available online for programming in Virtual Reality or Augmented Reality are developed using Blueprints. Since both languages can coexist in the same project, a future version can be rewritten using C++, while the original project using Blueprints can still be executed.

The latest version of Unreal Engine also includes a new tool for creating character animations. The game engine itself is heavier than others and demands high hardware and software specifications. This point will be expanded in the next sections.

Some characteristics of Unreal Engine 5:

- Lumen: A system created for Unreal Engine to support full dynamic illumination of the environment. It also makes all reflections very realistic, targeting next-generation consoles.
- Nanite: A system designed for detailed environments, using a virtualized geometry system. It divides every surface into many polygons and optimizes the fluency in the game, using an intelligent new mesh format and new rendering technology, allowing for high render pixel scale detail.
- MetaHumans: A framework that allows for the creation and animation of virtual humans with high photorealistic detail. It offers a cloud-based app to convert a mesh into MetaHumans using the plugin for Unreal Engine.
- MetaSounds: A procedural, high-performance audio generation solution that allows for flexible audio design and implementation. It offers features such as customization, third-party extensibility, graph re-use, and a powerful tool for in-editor sound design. MetaSounds is a Digital Signal Processing (DSP) rendering graph, providing powerful procedural audio, sample-accurate timing, and control at the audio buffer level. This tool makes it possible to integrate with game data and player interactions, triggering gameplay events and creating immersive experiences.

Licensing

Compared to Unity, Unreal Engine allows users to access all its features for free for educational purposes. This makes it easier to learn without restrictions on the options the game engine offers.

Unreal Engine has different plans for commercial use. Individuals, businesses, or institutions earning less than $1 million USD annually do not have to pay for the use of this software. If the product generates more than $1 million USD, a 5% royalty fee is required. According to the Unreal Engine website:

Unreal Engine is free to use for students, educators, hobbyists, and most non-games companies making less than $1 million USD in annual gross revenue.

- *For game developers and other users distributing applications that incorporate Unreal Engine code at runtime (such as a game) and are licensed to third-party end users, a 5% royalty is due (discounts may apply) when the lifetime gross revenue from that product exceeds $1 million USD. No royalties are due on the first $1 million in lifetime gross product revenue.*

- *For other companies, if you earn over $1 million USD in annual gross revenue, you will need to buy Unreal Subscription seats. Unreal Subscription costs $1,850 per user per year, and includes Unreal Engine, Twinmotion, and RealityCapture.*

This project was created in Unreal Engine. Compared to other licensing models, using UE5 makes it easier to sell products in the future. The possibilities for creation in UE5 are greater, with fewer restrictions on developing solutions, without the need to purchase additional licenses as with Unity for certain features that might not be used in the project. The decision to create this project with Unreal Engine is primarily due to the higher chances of successfully releasing a product compared to using Unity.

4 Environment in Detail 3D Assets

Some of the 3D models were original creations, while others were sourced from Sketchfab. These models were available for free use, provided that the authors were credited. Unfortunately, the author of the drums model has deleted their account. Therefore, it is recommended to use an alternative drums asset in future versions.

The Music Studio

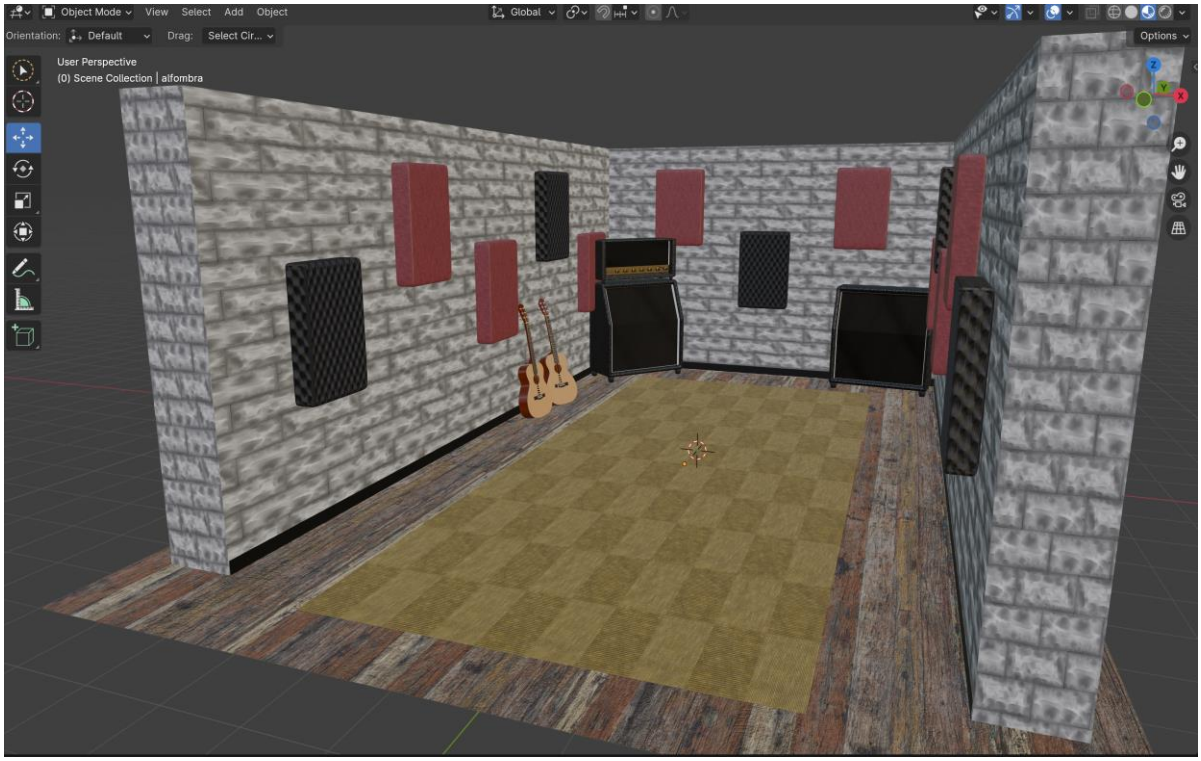

Figure 33 Scenario Music Studio
Source: Own creation

Figure 34 Speaker as Marshall
Source: Own creation

Figure 35 Model acoustic Guitar
Source: Sketchfab - Creator:(*Guitar - Download Free 3D Model by Ya (@Yarik16)*, 2021)

Tools

- Keyboard

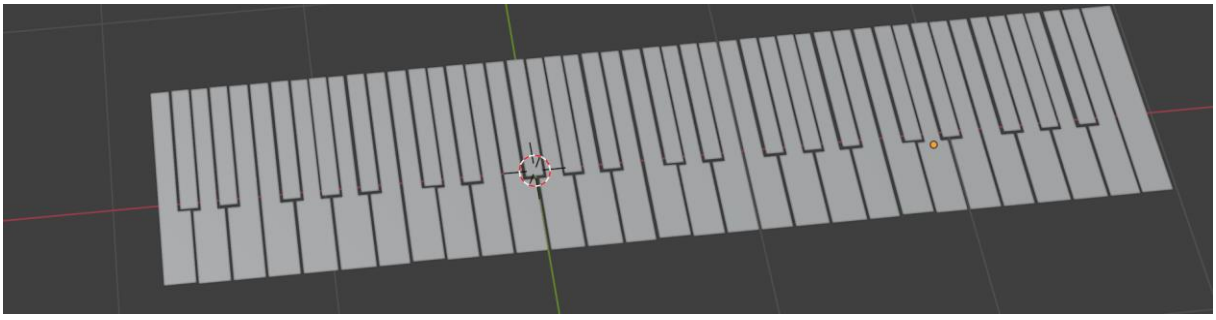

Figure 36 Model 3D Keys of Keyboard Panorama T4
Source: Own creation

- Music Sheet and Notes

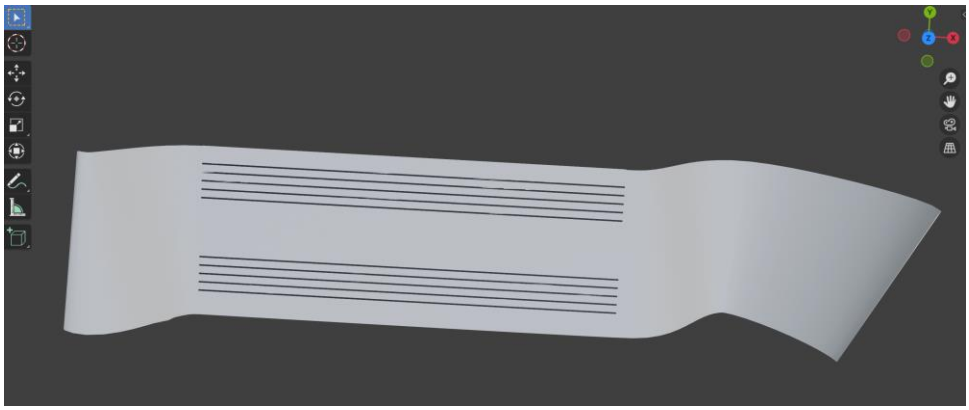

Figure 37 Model 3D of Music Sheet
Source: Own creation

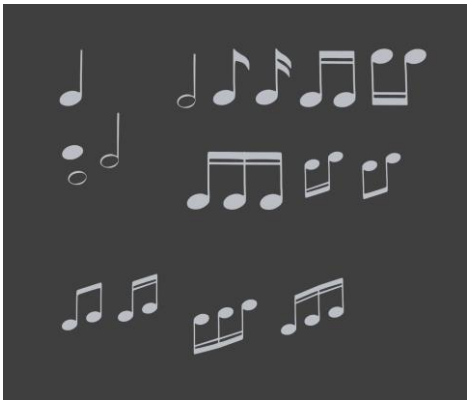

Figure 38 Model 3D of Music Notes
Source: Own creation

- A MIDI Controller

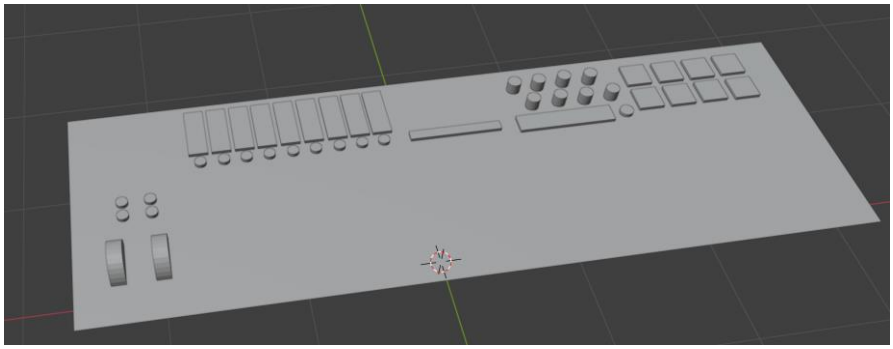

Figure 39 Model 3D MIDI Controller Buttons Paronama T4
Source: Own creation

- Instruments: Guitar, Bass, Drums

Figure 40 Model 3D Drums
Source: Sketchfab - Creator: (Model deleted by November 2024)

- Guitar and Bass

Figure 41 Model 3D - Bass and Guitar
Source: Sketchfab Creators: (*Percussion Bass Guitar - Download Free 3D Model by Kanade_Tatibana*, 2020) ; (*Electric Guitar Explorer - Download Free 3D Model by Skabl (@skabl_yt)*, 2021)

Interactive Actor

The robot Tone Tutor with characteristics that make it look friendly and fun.

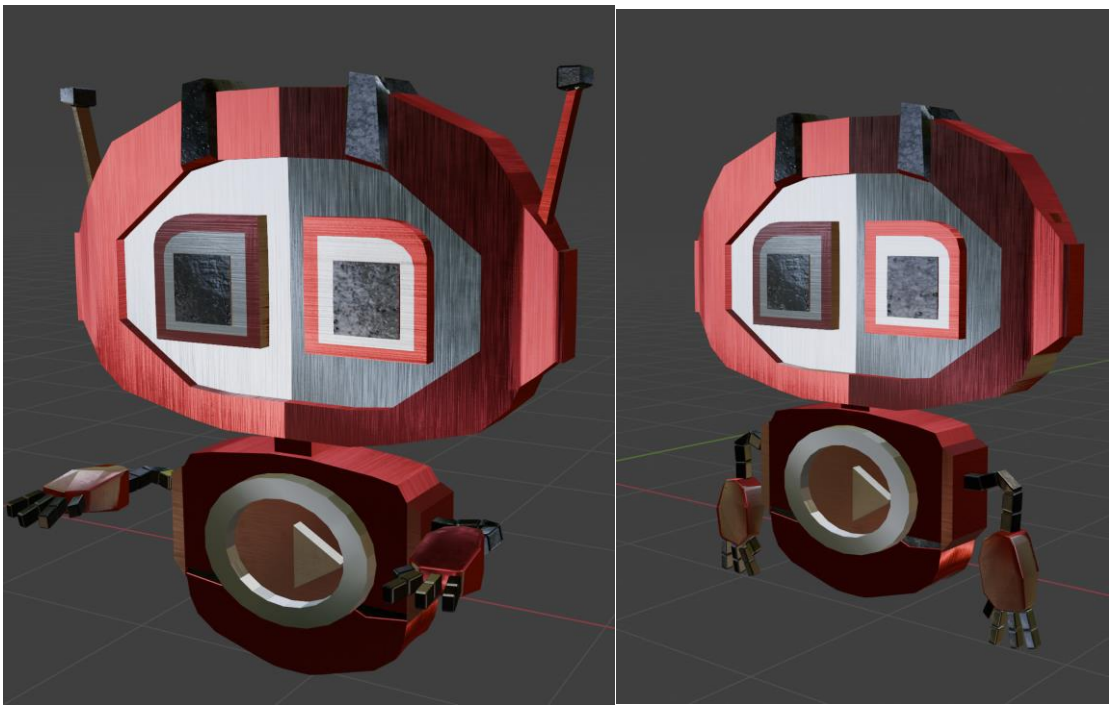

Figure 42 Model 3D Robot
Source: Own creation

5 Textures in Project

Figure 43 Metal Texture Black
Source: Ambientcg (*Metal 046 B on ambientCG*, n.d.)

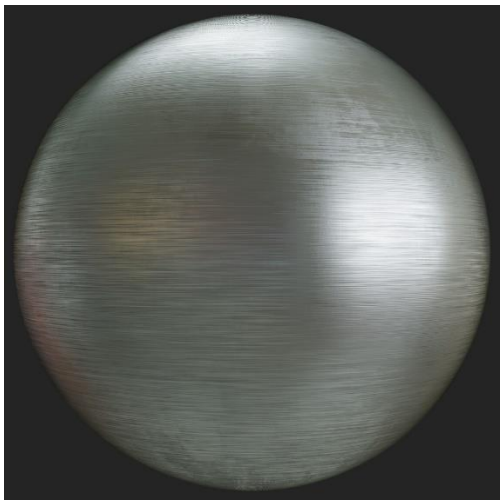

Figure 44 Metal Texture Silver
Source: Ambientcg (*Metal 009 on ambientCG*, n.d.)

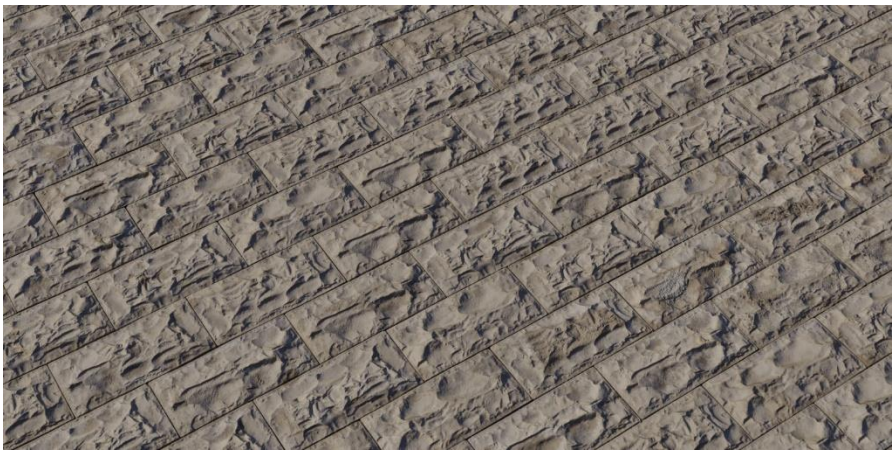

Figure 45 Wall Texture
Source: Poly Haven (Savva, n.d.-a)

Figure 46 Wood Texture
Source: Poly Haven (Savva, n.d.-b)

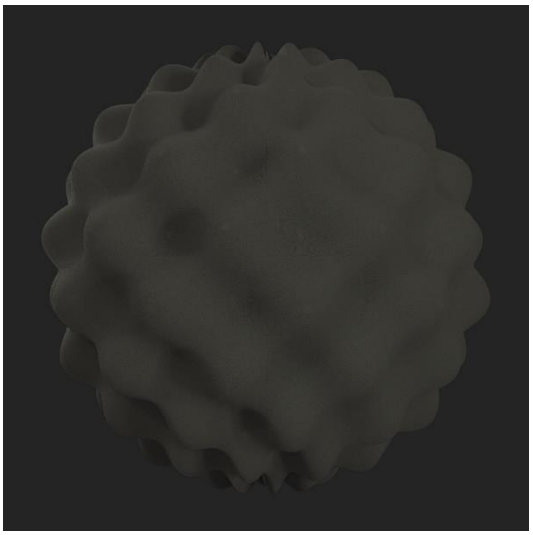

Figure 47 Foam Texture
Source: Ambientcg (*Acoustic Foam 001 on ambientCG*, n.d.)

Figure 48 Textil Red Texture
Source: Ambientcg (*Fabric 026 on ambientCG*, n.d.)

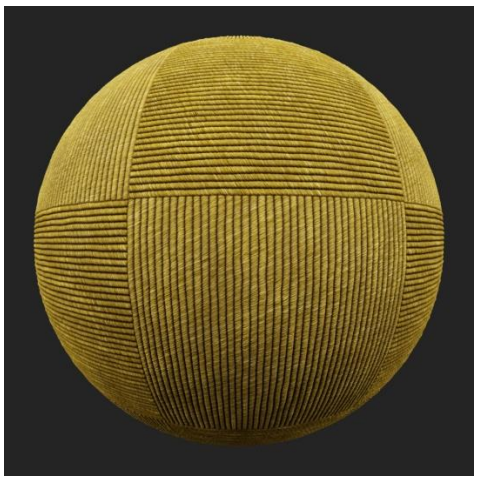

Figure 49 Textil Yellow Texture
Source: Ambientcg (*Fabric 025 on ambientCG*, n.d.)

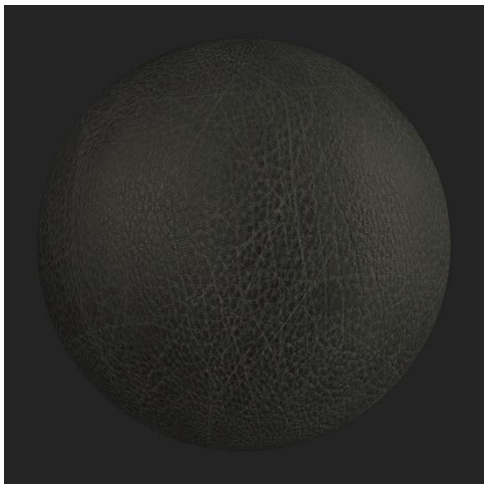

Figure 50 Leather Texture
Source: Ambientcg (*Leather 032 on ambientCG*, n.d.)

Figure 51 Plastic Texture
Source: Ambientcg (*Plastic 012 A on ambientCG*, n.d.)

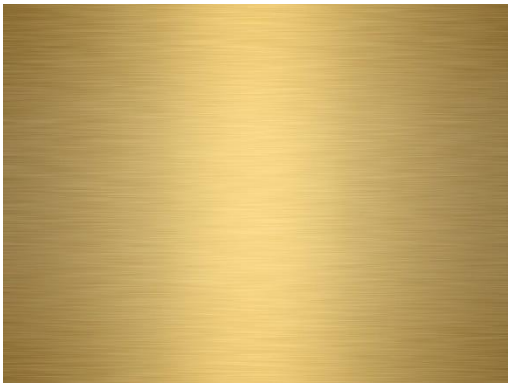

Figure 52 Gold Texture
Source: Myfreetextures (*Brushedgold3.Jpg (3500×2625)*, n.d.)

6 Questionnarie

Tone Tutor

1. Please enter your initials

2. How would you rate your overall enjoyment in every level?

	Not at all enjoyable	Slightly enjoyable	Moderately enjoyable	Very enjoyable	Extremely enjoyable
Level 1	○	○	○	○	○
Level 2	○	○	○	○	○
Tutorial	○	○	○	○	○
Play with the band	○	○	○	○	○

3. Rate each symptom based on your experience after using the simulator or VR environment

	None	Slight	Moderate	Severe
General discomfort	○	○	○	○
Fatigue	○	○	○	○
Headache	○	○	○	○
Eye strain	○	○	○	○
Difficulty focusing	○	○	○	○
Increased salivation	○	○	○	○
Sweating	○	○	○	○
Nausea	○	○	○	○
Difficulty concentrating	○	○	○	○
Fullness of head	○	○	○	○

4. What emotions did you experience while playing?

5. About the game's interface: Any specific visual elements you loved or disliked?

6. On a scale of 1 to 5, how excited are you to continue your musical journey with more lessons in this game?

- ◯ Not excited at all
- ◯ Slightly excited
- ◯ Moderately excited
- ◯ Very excited
- ◯ Extremely excited

7. On a scale of 1 to 5, how motivated are you to learn more music theory in general?

- ◯ Very motivated
- ◯ Somewhat motivated
- ◯ Neither motivated nor unmotivated
- ◯ Somewhat unmotivated
- ◯ Very unmotivated

8. What do you think about this way of learning music theory?

9. What ideas and opinions do you have that we can improve for the game?

10. How effective are the different levels of play in the game? Would you say that you learn something?

11. How could the training/workshop be improved?

12. Can you remember the notes in the C scale?

13. How easy was it to set up and start using the VR App?

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Neutral
- ◯ Somewhat not easy
- ◯ Extremely not easy

14. How much experience do you have using a VR headset?

- ◯ No experience
- ◯ Very little experience
- ◯ Moderate experience
- ◯ Quite a bit of experience
- ◯ A lot of experience

7 Music Theory for the game and dialog robot

Greetings

Hello, and welcome to Tune Tutor! We're so happy you're here, and we can't wait to help you learn to play the keyboard.

I'm Tone Tutor! I have a band, and we'd absolutely love to play together at a festival. Unfortunately, our keyboard player left the band, and we're now on the lookout for a new one! We'd love for you to take his place! All you have to do is complete all the levels, and we'll be ready to play!

Tutorial

Now, let's dive into the theory behind what you played! In music, we have 12 notes that are divided between tones and semitones. If we start with the key C and then move on to D and a black key, C#, we end up with what we see as a tone (T) and a semitone (S) in the 12-note scale, which represents the keys on the keyboard. And the notes go like...

C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
T	s	T	s	T	T	s	T	s	T	s	T	T

We're going to explore the first major scale, which is C. To do this, we'll look at the notes on the T box that has C, D, E, F, G, A, B,C. Once we've got that down, we'll move on to some more examples. For these, we'll use the same box and read all the T's on it. Are you ready to learn? Let's go!

T	s	T	s	T	T	s	T	s	T	s	T	T

major scale of D

D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D
T	s	T	s	T	T	s	T	s	T	s	T	T

Alright, now let's take those notes with the T and we'll get our D major scale all set up!

D-E-F#-G-A-B-C#

Major scale of E

E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E
T	s	T	s	T	T	s	T	s	T	s	T	T

E-F#-G#-A-B-C#-D#

The minor scale is closely related to the major scale; this means that we can derive a new scale from the relative note. This is fascinating, because if we take the C major scale pattern and start from the A note instead, our sequence of tones (T) and semitones (S) will shift, resulting in the following:

A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A
T	s	T	T	s	T	s	T	T	s	T	s	T

Now let's examine the values of T and see the results! We have our lovely A minor scale: A, B, C, D, E, F, and G, which uses the same notes as the C major scale. And here's an interesting fact: this applies to all major scales, since every major scale has a corresponding relative minor scale.

So, for now, we have a handy table with different positions of T so we can put our notes to get minor scales for any note!

T	s	T	T	s	T	s	T	T	s	T	s	T

Let's give the minor scale of D a try!

D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D
T	s	T	T	s	T	s	T	T	s	T	s	T

We're going to take the notes in T, and our scale is D, E, F, G, A, A#, and C.

Let's try the minor scale of C together!

C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
T	s	T	T	s	T	s	T	T	s	T	s	T

We take the notes in T, and our scale is C – D – D# – F – G – G# – A#.

We also play in a minor scale of E.

E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E
T	s	T	T	s	T	s	T	T	s	T	s	T

Let's take a look at the notes in T. We've got: E – F# - G – A – B – C - D

Guess what! We can now create chords as we have the scale.

I'm so excited to show you how to create chords! The formula for major chords is simple: root (1) – third (3) – fifth (5). Let's look at a few examples of scales and see how we can create the corresponding chords.

C-D-E-F-G-A-B-C

C major: C- E- G

D-E-F#-G-A-B-C#

D major: D - F# - A

E-F#-G#-A-B-C#-D#

E major: E – G#-B

It works with minor scales, which is great!

C – D – D# - F - G - G# - A#

C minor: C – D# - G

D – E – F- G- A -A# - C

D minor: D - F- A

E – F# - G – A – B – C – D

E minor: E – G – B

Let's take a moment to compare those scales for C major. Let's take a look at C, E, G, and C minor: C, D#, G, or for D major: D, F#, A, and D minor: D, F, A. Isn't it interesting how the middle note of this chord is reduced a semitone from the major to the minor?

That's how we create scales!

For this, we'll just use the six notes of the scale to create corresponding chords. Don't worry, it's really simple! And for every note, we'll use the simple formula 1-3-5 for every chord. You've got this!

As we bring this course to a close, we're so excited to keep making music together! You can play with us using the notes you already know for that scale, and you can also add some chords if you'd like! Whatever you feel comfortable with is great!

C	**D**	**E**	**F**	**G**	**A**
C	D	E	F	G	A
E	F	G	A	B	C
G	A	B	C	D	E
Major	Minor	Minor	Major	Major	Minor

Statement of independent work

I hereby declare that I have written the present Master's thesis entitled:

Tone Tutor: A virtual teacher for music theory

independently with only the stated tools. All passages that I have taken verbatim from literature or other sources such as websites have been clearly marked as quotations with the source indicated.

Hamburg, 29.11.2024

Place, Date