Rhythm Render
Interim Report

TU856
BSc in Computer Science

John Hinch

C21718368

Dr Bryan Duggan

School of Computer Science

Technological University, Dublin

30/11/2024

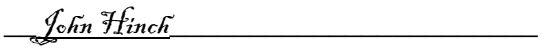
Abstract

This project presents the development of an innovative software tool that bridges the realms of digital music composition and 3D animation through the medium of Virtual Reality. The core of this tool is virtual MIDI sequencer interface that allows users to compose and manipulate musical sequences intuitively. Once a sequence is created, the software translates the data into a dynamic 3D visualization of a virtual performer or ensemble playing the sequence in real time. The VR environment is designed to enhance the immersion of this auditory experience.
The project aims to enhance the creative process by providing composers with a new medium for visualizing and experiencing their music. It also offers educational benefits, serving as a tool for learning about musical structure and performing dynamics. By merging MIDI sequencing with 3D animation, this software introduces a novel approach to interactive music visualization, with applications spanning music production, live performances, and multimedia art.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:



John Hinch

30/11/2024

Acknowledgements

I would like to thank my supervisor Dr Bryan Duggan for all of their support and guidance.

Table of Contents

1. Introduction

1.1. Project Background

Music Composition and visual art have long been intertwined, evolving parallel and influencing each other in various ways throughout history. The advent of digital technologies has expanded the boundaries of both fields, creating new opportunities for integration and real-tie interactivity. The development of multimedia software has made it possible to bridge auditory and visual experiences, offering new and exciting ways to express artistic vision and increase audience engagement.

Along with advancements in music technology, virtual reality (VR) has emerged as an incredibly powerful medium for creating immersive experiences that can engage multiple senses. While primarily used for gaming and simulation development, VR technology has found many applications in fields as diverse as education, art, therapy, etc. The use of VR to immerse users in a 3D environment opens up new ways for how people interact with and experience digital content, including music.

The core idea of this project takes inspiration from the increasing need for dynamic visualization tools in the world of music production, performance art and education. Traditional MIDI sequencers allow musicians to compose and arrange music digitally, however, they usually lack any integrated visual component that could enhance the user's experience and understanding of their composition. Nowadays, 3D animation technology has reached a level of sophistication where it can now simulate a realistic, complex visual, such as a musical performance. By combining these technologies, this project aims to change how musicians and audiences alike interact with music, offering both creative and entertaining benefits.

1.2. Project Description

This project is primarily designed for entertainment purposes, though there is the potential for educational applications as well. The system presents the user with a virtual environment and a simple MIDI sequencer. Here the user is free to create their own sequence of notes using the sequencer. Once the user is satisfied with their sequence, they simply press the perform button, this will create a 3D animation of a person playing the sequence the user created. The user will be able to select from a variety of instruments and will be able to play several animations at once.

The system is implemented using the Godot 4 games engine and Meta Quest 3 virtual reality headset. Users can interact with the system using the controllers of the Quest 3. I hope that this project will provide an entertaining and user-friendly way of creating and experience music.

1.3. Project Aims and Objectives

The overall aim of this project is to provide a new entertaining and immersive way to create and experience music. It will provide people, both new and old to music production, a brand-new way for to create and experience their music in real time. I would also like to learn and utilize a range of different production approaches and different software.

- Provide users an immersive environment to create music
- Render music sequence into 3D animation
- During operation, system should be simple and easy to follow for users with clear and concise explanations of system and controls.
- Meta Quest 3 us used to provide Virtual Reality environment and methods of control. It will also be used to allow user to interact with the environment

- The 3D environment is implemented using Godot game engine and scripts are written in GDScript programming language.
- XR plug ins will be used to create VR solutions in Godot
- Implementing a control scheme to allow the user to interact with the system
- Implementing algorithms to ensure smooth animation transitions

1.4. Project Scope

This project focuses on creating a VR environment for users to create music using a virtual MIDI sequencer and then creating an animation for said sequence. This project will not feature any online functionality, including multiplayer or song sharing.

1.5. Thesis Roadmap

This section will provide an overview of what each of the following chapters is about

Literature Review

- This chapter explores alternative existing solutions to the problem that this project attempts to overcome. It also investigates how functions of these existing solutions may be implemented into this project.

System Design

- This section explains the methodology used for this project

Prototype Development

- This chapter details the process of developing a prototype and how the design given in the previous chapter was implemented.

Issues and Future Work

- This chapter discusses the difficulties encountered during development and plans for future work will be described here

2. Literature Review

2.1. Introduction

This chapter will discuss the investigation of alternative approaches to existing solutions to the problem that this project hopes to address. It also discusses the research done on potential technologies that may be implemented to the project.

2.2. Alternative Existing Solutions to Your Problem

The principal research areas that were important for this study will be featured in this section. There are many existing Virtual Reality (VR) applications that currently exist do offer a solution to one problem that is addressed by my project, and I have included research on two of those solutions. However, I believe that most of them only address one problem, music creation. Neither solution feature any sort of performance animations, which I hope to include in my project

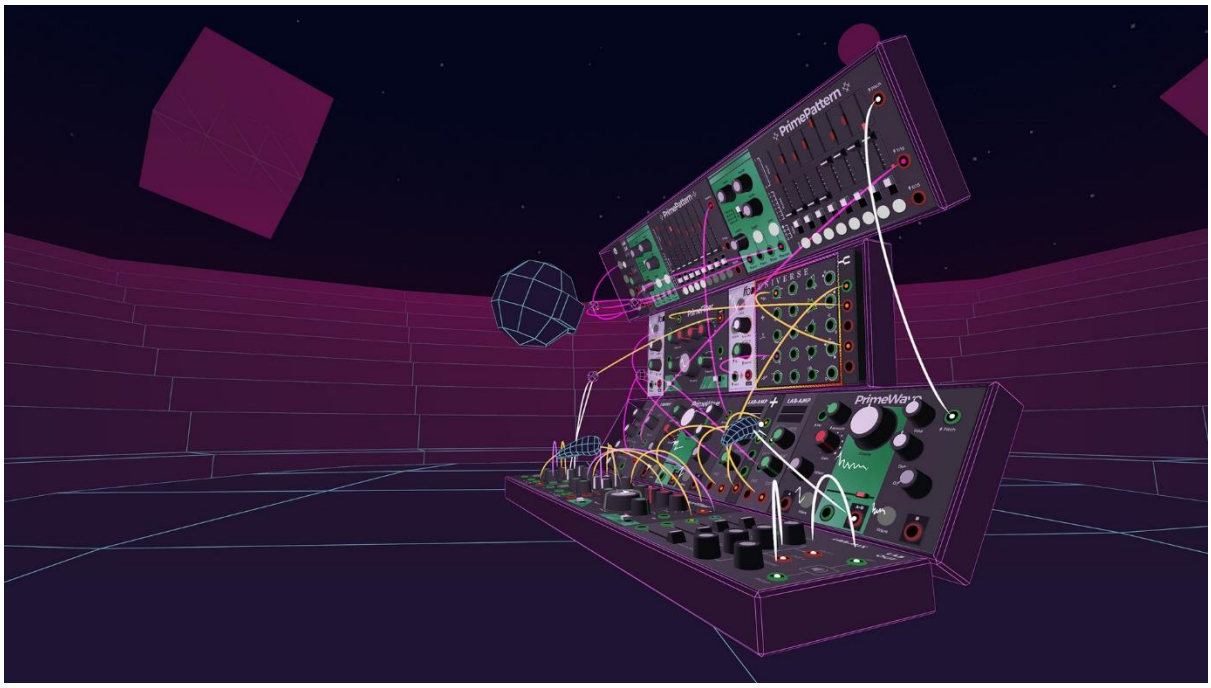
2.2.1. Virtuoso

Virtuoso is a VR application developed by Reality Interactive AB and published by Fast Travel Games AB for the Meta Quest 2 and Meta Quest 3 VR headsets along with Steam VR support. It was released on the 10th of March 2022 and continues to receive updates. The game features a robust music creation suite with various different instrument support. The suite is incredibly robust allowing the user to modify fine details for the instruments. While this application is very impressive, I believe that it is almost too complex. On top of this the game features no performance aspect for the music you create which my project aims to implement.

Screenshot of Virtuoso (Virtuoso, n.d.)

2.2.2. SynthVR

SynthVR is a VR application developed by 42tones for Steam VR, released on the 31st of January 2021. This is a modular synthesizer environment for virtual reality. The application is incredibly robust but is limited to synthesizer technology. While incredibly robust, I feel this application is both limited and too technical. There is also no animation functionality.



Screenshot of SynthVR (SynthVR, 2021)

2.3. Technologies you've researched

A Virtual Reality application can be implemented using a wide variety of technologies. This section will explore some of those technologies that were investigated and chosen for this project

2.3.1 Meta Quest 3

The Meta Quest 3 is a cutting-edge virtual reality (VR) and mixed reality (MR) headset developed by Meta (formerly Facebook). Released in October 2023, it is a significant improvement over its predecessor, the Quest 2, by offering enhanced hardware capabilities, improved user experience, and advanced features for both VR and MR applications. The headset introduces full-colour passthrough Mixed Reality, allowing users to interact with digital content overlaid on their physical environment, providing an immersive experience.

Promotional art for the Meta Quest 3 (Meta, n.d.)

2.3.2 Godot 4 Games Engine

Godot is a free and open-source game engine celebrated for its versatility and accessibility, making it a popular tool for creating both 2D and 3D games. Since its initial release in 2014, Godot has become a go-to choice for independent developers and small studios seeking a powerful yet user-friendly development platform. One of its standout features is its intuitive scripting language, GDScript, designed for rapid development and ease of use, alongside support for C# and other languages. The engine's open-source nature fosters a thriving community that actively contributes to its development and provides a wealth of resources for learning and support. (Godot, n.d.)

A major milestone in the engine's evolution came with the release of Godot 4.0 in March 2023. This update introduced a host of significant improvements, with one of the most noteworthy being a completely rewritten 3D rendering engine that leverages the Vulkan API. This upgrade brings substantial performance enhancements, enabling developers to create visually stunning and highly optimized 3D environments. Additionally, Godot 4 expanded its capabilities in immersive technology by incorporating support for XR (extended reality) frameworks, including virtual reality (VR) and augmented reality (AR). This makes the engine particularly appealing for projects exploring cutting-edge experiences in these domains, opening new doors for creators to build engaging and interactive applications.

2.3.3 Blender

Blender is a free, open-source 3D creation suite widely used for various applications, mainly 3D modelling, animation and rendering. For this application the main focus will be on the robust animation suite, featuring character rigging, keyframing and motion paths. (Blender, n.d.)

2.4 Motion Capture Software

2.4.1 FreeMoCap

FreeMoCap, or The Free Motion Capture Project is an open-source project that aims to provide research-grade markerless motion capture software to everyone for free. Built using a user-friendly framework to achieve the goal of creating a system that serves the needs of professionals while remaining intuitive to a child with no technical training or outside assistance. (FreeMoCap, 2021)

2.4.2 Rokoko Studio

Rokoko Studio is a versatile mocap software that works both using a proprietary Smartsuit as well as basic tracking with iOS devices. This software is very beginner friendly and accessible for basic motion capture, however it is limited on its free plan. The limited plan also limits length of video to just 15 seconds

2.4.3 DeepMotion Animation 3D

Animate 3D by DeepMotion is a cloud-based AI motion capture solution that turns video footage into 3D animations. No hardware needed, only an internet connection and a video clip. It is very easy to use, upload video, and get 3D motion data output. It's highly accessible, as you don't need special hardware. Results can be exported to FBX for use in VR engines. Free version limits the length of animation, and the processing is cloud-based so requires internet and has some delay. Paid plans for higher quality and longer animations.

2.4.4 Kinect-Based

Useful if you have a Kinect sensor, software like **iPi Soft** (which has a trial) or **Brekel** offers robust full-body tracking with these older devices. Requires some setup but is straightforward once you have Kinect installed and set up. Kinect offers decent full-body tracking at a low cost, making it a good choice for VR. The community provides support and advice for optimizing results. Limited accuracy compared to higher-end systems, though adequate for simple projects.

2.4.5 Comparison

Feature	**FreeMoCap**	**Rokoko Studio**	**DeepMotion Animate**	**Kinect-Based**
Cost	Free	Free - $50/m	Free - $83/m	Trial
Hardware Needed	Cameras	iPhone/Smartsuit	Video	Kinect
Ease of Setup	Moderate – High	Easy	Easy	Moderate
Accuracy	Good (camera dependent)	High	Moderate	Moderate-High
Performance	Not optimized for real-time	Real-time	Post-processing	Real-time
Customization	High (open-source)	Low	Low	Moderate

2.5. Existing Final Year Projects

2.5.1 Rhythmically Generating an Audio Virtual Reality Experience

Student: Graham Byrne

Description: This project creates a system that detects rhythmic patterns in any given music and uses those patterns to procedurally generate terrain and objects in a virtual reality video game. This is achieved using digital signal processing techniques, procedural object and mesh generation techniques. This project was developed using C# and Unity.

2.5.2 VR Music Learner

Student: Maciej Golubski

Description: This project is designed to help people learn chords and instruments in an innovative way with the help of a tracker on a mobile phone. Using Virtual Reality to provide the users an interactive experience for people to learn rhythm and a variety of chords which are found on an instrument such as guitar, piano or others. The system allows a user to select from three options. Each of which will have a different function associated with them. The first function will allow the user to learn the rhythm on the instrument of their choosing. A song will be played with different patterns appearing on the screen using artificial intelligence to determine which image to show. The

user has to play either with their own guitar to match the rhythm or can use the controllers provided with the headset to do different movements and button presses.

2.6. Conclusions

A well designed VR application, like any application, requires extensive research and accurate utilizing of technologies to be accomplished. I have chosen the technologies that will be used for this project for the reasons outlined above. This project uses Godot 4 for writing scripts and other programming features and uses FreeMoCap to record motion capture videos for animations, while Blender will be used for fine tuning animations and exporting to Godot. This project will be available on the Meta Quest 3 platform.

3. System Design

3.1. Introduction

This chapter will explain the various software methodologies implemented within this project.

3.2. Software Methodology

A software methodology is a structured framework or set of practices used to plan, develop, test and maintain software projects. It outlines the processes, techniques and tools employed by teams to ensure the efficient and effective delivery of high-quality software products. By using these approaches to software development, it can help manage complexities, reduce risks, and align the project's outcome for stakeholders.

3.2.1 Agile Methodology

Agile Methodology has become one of the most popular software development methodologies in recent years. This is because of its emphasis on flexibility, collaboration and iterative progress. It focuses on delivering small incremental updates to the project rather than completing the entire project in one large release. There is a priority on customer satisfaction, adaptability to change, and continuous improvement. Communication between the developers and the customers/users is a high priority for Agile. (Laoyan, 2024)

Pros:

- Effective and economical in terms of time and money
- Very flexible and responsive to any changes requested by the users
- Ideal for fast moving development projects

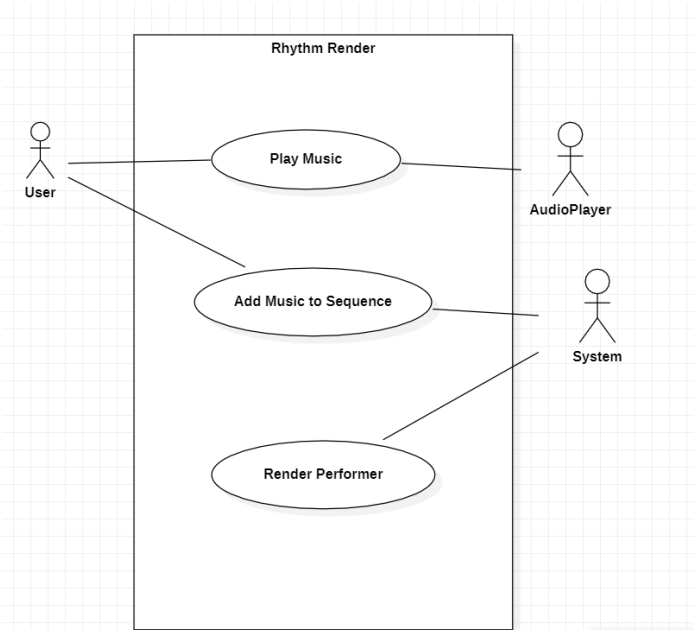
Cons:

- Due to the flexible nature, it can be harder to predict final costs or timelines
- Agile can be difficult to implement in large teams or complex projects
- Agile requires team commitment, which can be challenging for some teams

3.3. Overview of System

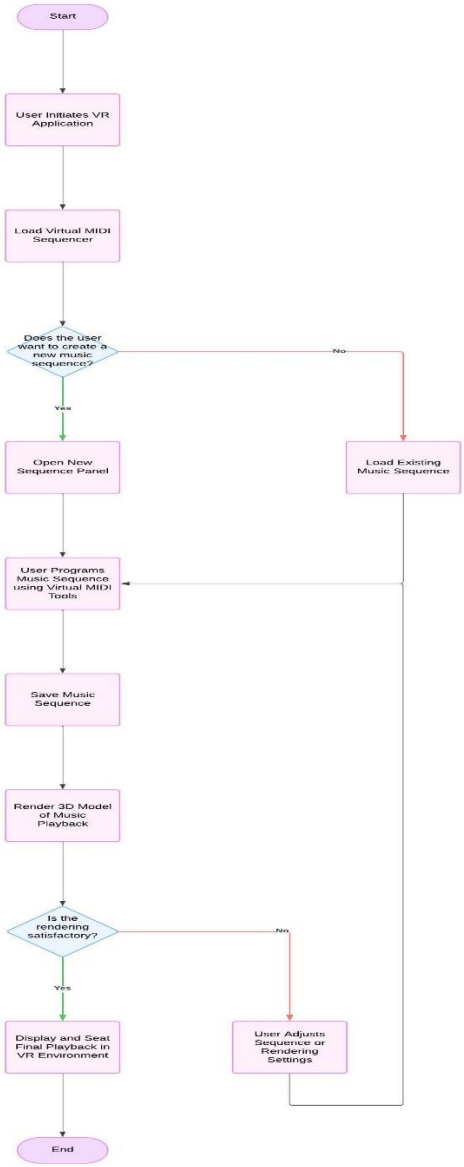
This system consists of three core components, the sequencer, the 3D rendering engine and the Virtual Reality interface. The sequencer forms the foundation of the system, allowing users to compose musical sequences. Featuring an accessible interface. The system captures this data in order to create the subsequent visualisation stage. The rendering engine processes the data generated from the sequencer to create a 3D animation of the musical performance, with each instrument being mapped to a separate model. As the sequence is played back the animation reflects real-world performances. The VR interface provides users with an interactive and immersive environment where they can experience their compositions. Users will be able to observe the animations from different angles. This component leverages VR technology to create a multisensory experience.

3.3.1 Use Case Diagram



The above Use Case Diagram outlines a simplistic overview of the project system. The user programs the music sequence, the sequence is saved and used to render a 3D model of the performance.

3.3.2 Flowchart Diagram



3.4. Conclusions

Both development team and client can be benefited from choosing the right software development methodology. A good design methodology provides you with an organized and structured set of techniques that make the design process easier and reduce the amount of time you spent to accomplish a project. The project is developed in accordance with the diagrams provided in this part in the following development stages

4. Prototype Development

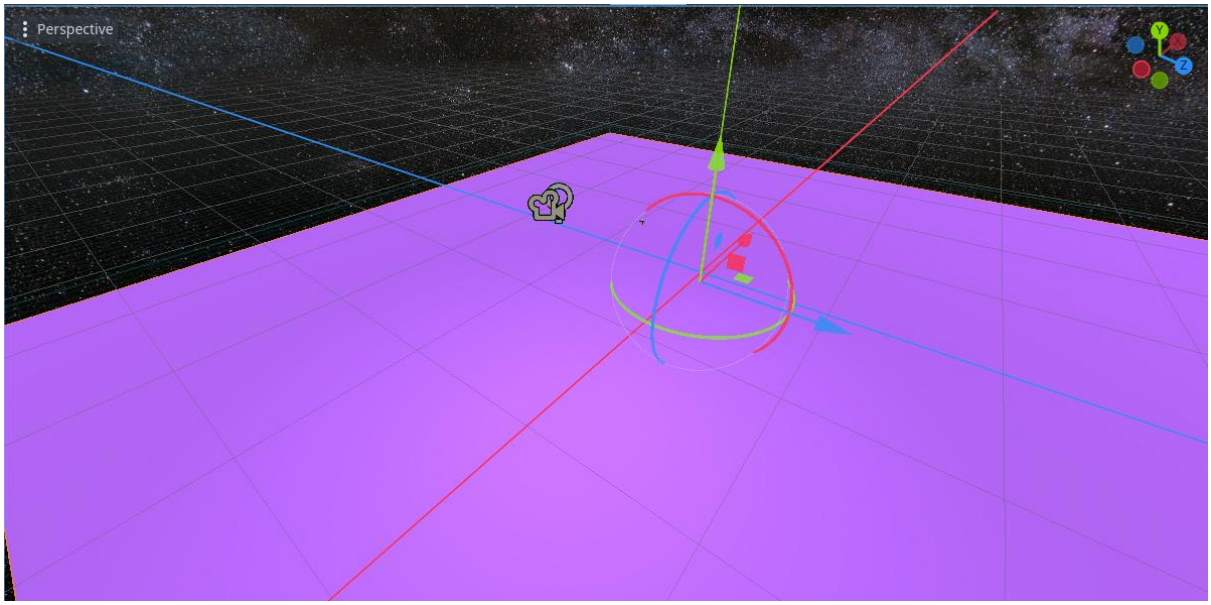
4.1. Introduction

This chapter will explore the development process involved in creating the prototype featured for this report. This prototype focuses on the transition to the performance scene with the corresponding music. Code examples will be provided.

4.2. Prototype Development

4.2.1 Sequence Scene

The Sequence Scene serves as the main scene where users will create their sequences for now. The scene comprises of several game objects as well as user interface (UI) elements, all of which will be described in detail below. In the final product users will use this scene to create their music sequence by interacting with 3D objects to choose what notes are played and when. However, for now they choose the sounds using UI elements for this prototype.



Sequence Scene in Godot 3D Viewer

Every object contained within this scene is a child of a Node3D. Within this parent there are several different types of Nodes

Nodes

- **WorldEnivronment –** Environment properties for the entire scene such as lighting or post-processing. For this scene I am using it to act as a skybox using a panoramic texture of space that I found for a separate project.
- **OmniLight3D –** Omnidirectional light source used to light the scene. I do feel that it needs further fine tuning.
- **CanvasLayer –** A node used for creating independent elements of a 2D scene. For this scene it is being used as a base class for the GUI
    - **Control –** Class for handing GUI elements, connected to the control and control scene which will be explained in detail later
- **StaticBody3D (Ground) –** A 3D physics body that doesn't move from external forces and doesn't affect other objects when it moves. Is being used as the ground for this scene

    - **CollisionShape3D –** A node that provides a shape to a collision body, in this case a flat square. All physics bodies need Collision shapes
    - **MeshInstance3D –** Creates a 3D mesh in the current scene. Used to give the floor a visible aspect
- **CharacterBody3D –** A 3D physics object designed specifically to be moved using script. Used to represent the player in the scene
    - **Camera3D –** Camera Node, used to display the scene to the player. Without it, the scene would be blank. It is a child of the CharacterBody3D as this is a first-person experience, the camera represents the players POV.
    - **CollisionShape3D –** Represents the player's body and prevent colliding with other objects. As CharacterBody3D is a physics object it needs a collision shape attached to it.
- **Marker (sequence) –** A generic 3D node used for positioning with a hint for editing. The sequence script is attached to this marker as this represents the sequencer position in the 3D scene
    - **Timer –** Countdown Timer. Currently not in use but will be used for the sequencer bpm, or beats per minute

Scripts

Sequence.gd

This script will handle the sequencer functionality of the application. Contained within this script there are several key functions.

Load_Samples()

```
func load_samples():
    var dir = DirAccess.open(path_str)
    if dir:
        dir.list_dir_begin()
        var file_name = dir.get_next()
        while file_name != "":
            if dir.current_is_dir():
                print("Found directory: " + file_name)
            if file_name.ends_with('.wav') or file_name.ends_with('.mp3'):
                file_name = file_name.left(len(file_name))
                var stream = load(path_str + "/" + file_name)
                stream.resource_name = file_name
                samples.push_back(stream)
                file_names.push_back(file_name)
                # $AudioStreamPlayer.play()
                # break
            file_name = dir.get_next()
```

This function is used to load the music files that will be played during an instance of the sequence scene. Using built-in functions, this creates a stream that will be used to list all files and directories in a given location, the location is where the sounds files are stored. The program searches through the list of files for any that end with .wav or .mp3, or sound files. These files are then added to the stream, which is an array that contains all relevant music files.

Make_Sequencer()

```
func make_sequencer():
	
	for col in range(steps):	
		
		for row in range(samples.size()):
			var pad = pad_scene.instantiate()
			
			var p = Vector3(s * col * spacer, s * row * spacer, 0)
			pad.position = p	
			pad.rotation = rotation
			
			#pad.area_entered.connect(toggle.bind(row, col))
			pad.input_event.connect(toggle.bind(row,col))
			add_child(pad)
			pad.name=str(samples[row])
```

This function is used to create the physical sequencer. Using a nested for loop the function instantiates a number of pad scenes based on the length of the array samples, which contains the relevant music files, and for the number of steps in the sequence. This function creates a grid of 3D objects which will be how the user will interact with the system in the finished product.

Toggle()

```
func toggle(e, row, col):
	print("toggle " + str(row) + " " + str(col))
	sequence[row][col] = ! sequence[row][col]
	play_sample(0, row)
	print_sequence()
	
```

Used to toggle the pads in the sequencer which will then be used to determine if a pad is in the sequence.

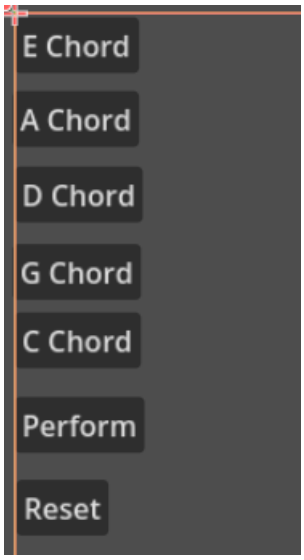
Play_sameple()

```
func play_sample(e, i):
	
	print("play sample:" + str(i))
	var p:AudioStream = samples[i]
	var asp = players[asp_index]
	asp.stream = p
	asp.play()
	asp_index = (asp_index + 1) % players.size()
```

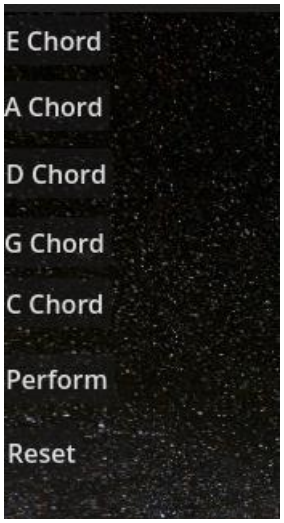
Used to play the corresponding sound file

4.2.2 Control

This is a 2D Control scene which contains a number of buttons used in this prototype to represent the sequencer, with a button corresponding to a chord on the guitar. As the user presses a button the sound file plays and is added to the sequence. There is also a perform button which brings the user to the performance scene. Finally, there is a reset button to reset the sequence.



Control node in the 2D Editor



Control as it appears running in the application

Scripts

Control.gd

The current script connected to the Control scene is a placeholder to represent the functionality of the sequencer. For the final project I hope to convert this into a menu for adjusting the sequencer itself, such as changing the instrument, the number of steps I the sequence, etc. For now the five chord buttons are connected to a function _on_button_pressed()

```
func _on_button_pressed(button_name:String):
    if chords.has(button_name):
        var sound = AudioStreamPlayer.new()
        sound.stream=chords[button_name]
        add_child(sound)
        sound.play()

        #sound.connect("finsihed",sound,"queue_free")
        GlobalSoundManager.sound_sequence.append(chords[button_name])
    print("Button Pressed:" + button_name)
    #print(GlobalSoundManager.sound_sequence)
```

_on_button`_pressed(button_name:String)

This function connects each button to the sound file with the same key as the button name. For this prototype the sound files are hardcoded in a dictionary with a key. If the dictionary contains the correct sound file an AudioStreamPlayer node is created which then plays the sound file. The sound file is added to the sound sequence array located in the global node GlobalSoundManager, which is a node that can be accessed by any file in the project.

```
var chords={
    "E Chord":preload("res://Sounds/Guitar/e_major.mp3"),
    "A Chord":preload("res://Sounds/Guitar/a_major.mp3"),
    "D Chord":preload("res://Sounds/Guitar/d_major.mp3"),
    "C Chord":preload("res://Sounds/Guitar/c_major.mp3"),
    "G Chord":preload("res://Sounds/Guitar/g_major.mp3")
}
```

Hardcoded Sound Files

```
func _on_perform_pressed():
    if GlobalSoundManager.sound_sequence.size()>0:
        #print(GlobalSoundManager.sound_sequence)
        get_tree().change_scene_to_file("res://Performance.tscn")
    pass # Replace with function body.
```

_on_perform_pressed()

This function is connected to the perform button. It simply checks if there is anything in the Global sound sequence, and if there is change to the performance scene

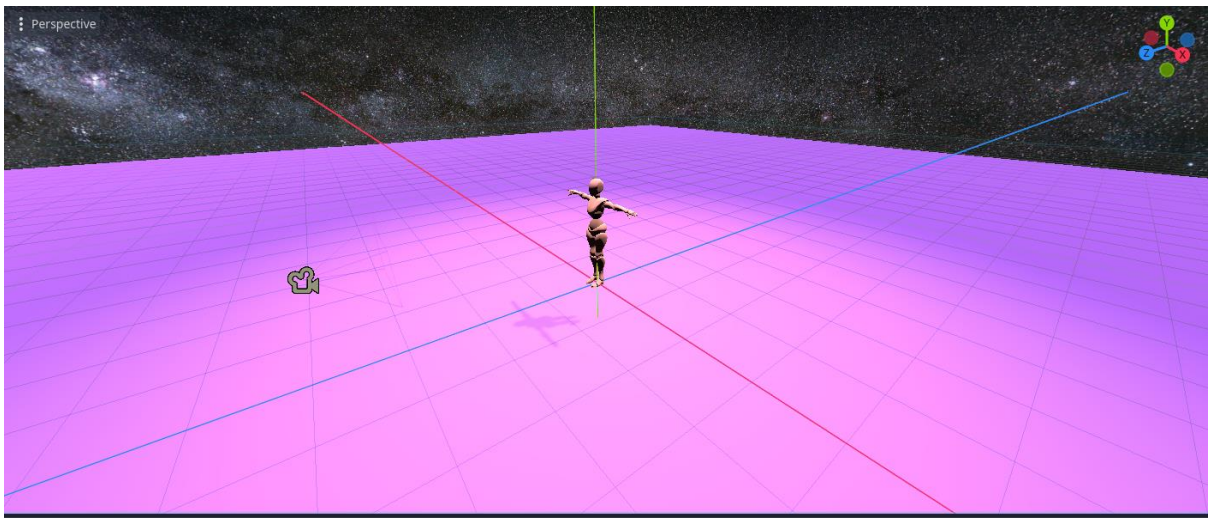
```
func _on_reset_pressed():
    GlobalSoundManager.sound_sequence.clear()
    print("Sequence Clear")
    pass # Replace with function body.
```

_on_reset_pressed()

This is connected to the reset button and simply clears the sound sequence array in the Global Sound Manager Node, for resetting the sequence.

4.2.3 Performance

This is a 3D scene where the animations of the performance related to the user's sequence will play. It is laid out in a similar way as the Sequence scene for consistency in terms of the Word Environment node, Camera Node and Omni light node. For this prototype I am using a stock animation that was downloaded from Mixamo and imported from Blender. This file contains a fully rigged model and an AnimationPlayer node for playing and manipulating animations. There is also a simple control node that contains a home button to bring the user back to the sequence scene.



Performance Scene in Godot 3D Editor

There are two scripts connected to this scene

Animation_Player.gd

```
extends AnimationPlayer


# Called when the node enters the scene tree for the first time.
func _ready():
    var animation=$"."


    if animation:
        animation.play("mixamo_com")
        print("Animation Found")
    else:
        print("Animation Not Found")
    pass # Replace with function body.


# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta):
    pass
```

This is simply to play the correct animation, in this case the mixamo_com animation.

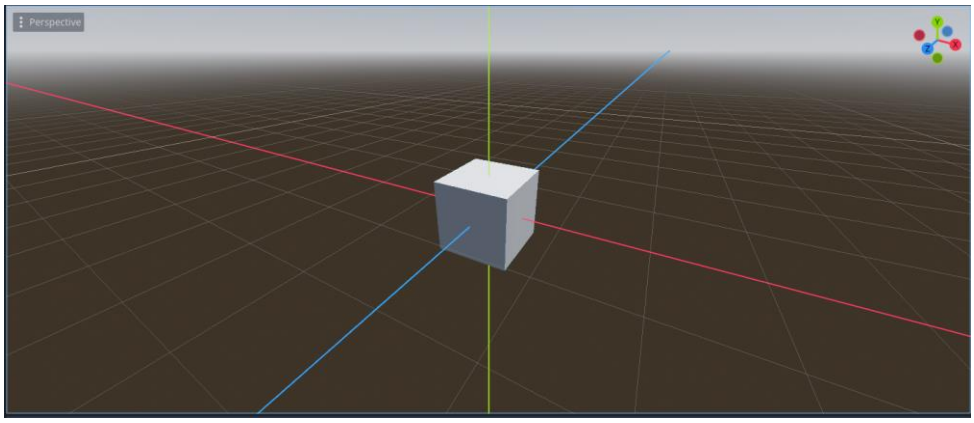
Audio_sequence.gd

```gdscript
extends Node

var current_index = 0  # To track the current sound being played.

# Called when the node enters the scene tree for the first time.
func _ready():
    if GlobalSoundManager.sound_sequence.size() > 0:
        play_next_sound()
    else:
        print("No sounds in the sequence to play.")

func play_next_sound():
    if current_index < GlobalSoundManager.sound_sequence.size():
        var audio_stream = GlobalSoundManager.sound_sequence[current_index]
        var sound_player = AudioStreamPlayer.new()
        sound_player.stream = audio_stream
        add_child(sound_player)
        sound_player.play()

        print("Playing sound:", audio_stream)

        # Connect signal using a Callable
        sound_player.connect("finished", Callable(self, "_on_sound_finished"))
    else:
        print("Finished playing the sequence.")
        current_index = 0  # Reset or stop based on your needs.

func _on_sound_finished():
    current_index += 1
    play_next_sound() # Play the next sound.
```

This script handles the Audio playback functionality of the scene. By using the Global Sound Manager sound_sequence array of a given index the script assigns the corresponding sound file to the audio_stream variable and creates a new Audio Stream Player node. It then plays the sound file. Once the sound file has finished the index increases and begins the process again for the next element in the array. If the script reaches the end of the array the index is reset to repeat the sequence.

4.2.4 Button

This is a 3D area scene that is used for the buttons of the sequencer. During the make_sequencer function of the sequence script, this is the scene that gets instantiated. It contains a MeshInstance3D, CollisionShape3D and AudioStreamPlayer3D



Button scene in Godot 3D Editor

Scripts

Button.gd

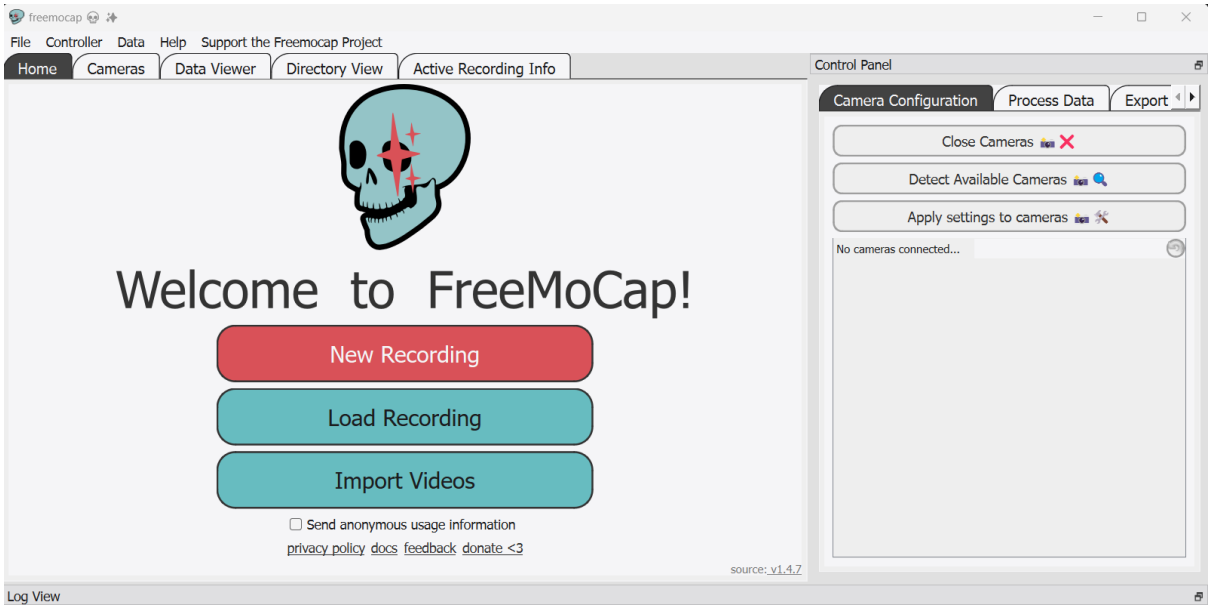
```
func _on_button_pressed(button_name: String) -> void:
    #print("Click")
    #mat.albedo_color = out_color
    _toggle()
    if play:
        $AudioStreamPlayer3D.play()
    print("Button Pressed: %s" % button_name)


# Called every frame. 'delta' is the elapsed time since the prev
func _process(delta: float) -> void:
    pass

func _toggle():
    if toggle==false:
        mat.albedo_color=out_color
    else:
        mat.albedo_color=in_color
    toggle=!toggle
```

The intended functionality of this script is that when a user clicks on an individual button the toggle function will change the colour of the button, and the corresponding audio file will play. I hope to expand this as well so that the Boolean variable toggle will be used to determine what sounds should be added to the sequence, if true add it, else don't. However, as of this prototype I have not been able to isolate the individual buttons.

4.3. Motion Capture

4.3.1 FreeMoCap

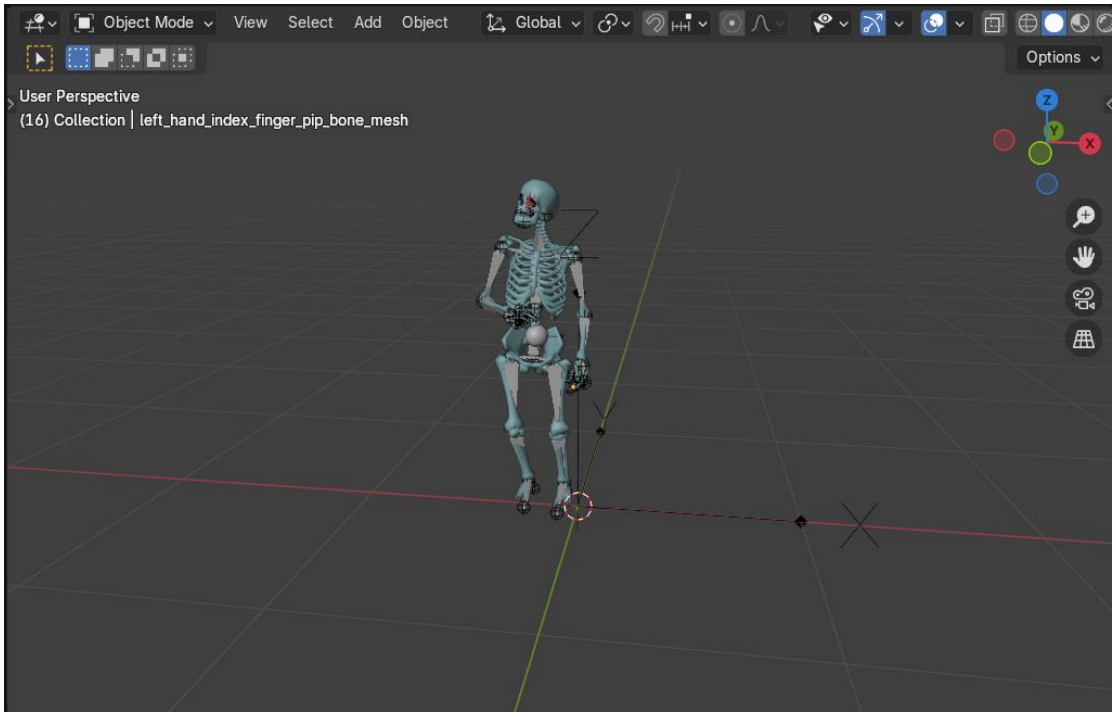
FreeMoCap is an open-source project that aims to provide research-grade marker less motion capture software to everyone for free. Built using a user-friendly framework in Python to achieve the goal of creating a system that serves the needs of professionals while remaining intuitive to a child with no technical training or outside assistance. For the purpose if this project I will be recording people playing various instruments to be used as animations in my applications. FreeMoCap uses a multicam set up and creates a blender file of the animation.



FreeMoCap Suite

4.3.2 Blender

Blender is a free, open-source 3D creation suite widely used for various applications, mainly 3D modelling, animation and rendering. For this application the main focus will be on the robust animation suite, featuring character rigging, keyframing and motion paths. I am primarily using Blender in this project to import my motion capture videos, as the software that I use automatically creates a blender file using the video



Animation in Blender Suite

4.4. Conclusions

This chapter explored, in detail, the process of developing this prototype, as well as the core functionalities of the different scenes and scripts contained in the project. Code snippets were included to offer a better understanding along with explanations.

5. Issues and Future Work

5.1. Introduction

This chapter will explore the various challenges and issues that were faced during the development of this prototype. Along with this, this chapter will detail the future work that I have planned and hope to have finished for the final product.

5.2. Issues and Risks

Several issues arose during the development of this prototype, several of which I believe stem from being new to the software that I used. One of the biggest issues that I faced, and am currently still facing, is that of import my motion capture animations into Godot.

For some reason, when I try to import the blender file of an animation created using FreeMoCap into Godot, the animation does not import correctly. I can see individual parts of the rig have animations, but they are not animating together. As well as this, the model does not seem to be animating with the rig. I hope that this is being caused by just my inexperience with both Blender and Godot, and that I can fix it soon. I do have an alternative solution that I will go into detail in the following section.

A similar issue that was faced using FreeMoCap was that using only one camera to record the motion capture caused the blender animation to act as if it was 2-Dimensional, the model was flat and there was no depth. Even if the recording had the person turn around, the blender file would slide against itself until it was facing the right way. Thankfully there is a simple solution and that is to just use the supported multi-cam functionality of FreeMoCap.

Another issue that I am facing is with the interaction of the 3D sequencer itself. For some reason the code is not registering when the user clicks on one of the 3D buttons, which in turn means that the buttons cannot currently be used for the sequencer. Similarly, when the script does recognise a mouse click, every instance of the 3D buttons is affected. For now, and for this prototype I have come up with the solution of using a 2D menu with hardcoded buttons, which do work as intended.

5.3. Plans and Future Work

There is still quite a lot of work to be done before this project is completed. First and foremost, I want to get the 3D buttons to work properly which I believe should be very doable. Once the buttons work as intended, the next step will be to get the animations imported correctly. If this cannot be done using FreeMoCap then I intend to use one of the alternatives I discussed earlier, failing that I could try to find animations that have already been made that would be close enough to the animations that I want.

While those are the main functions that need to addressed as they are the core functions of the project, there is still more functionality that I hope to include, such as a menu to allow the users to change aspects of the sequencer like instrument, bpm, and number of steps in the sequence.

Finally, once the core functions have been implemented properly, I then need to convert the application into VR and ensure that it still functions as intended. I believe, from my research, that converting the application into VR should be relatively straightforward, only needing to change certain nodes and implementing stock logic scripts for controls, user interactions and smoothness.

Bibliography

Blender, n.d. *About Us: Blender.* [Online]
Available at: https://www.blender.org/about/
[Accessed October 2024].

FreeMoCap, 2021. *About Us: FreeMoCap.* [Online]
Available at: https://freemocap.org/about-us.html
[Accessed October 2024].

Godot, n.d. *Godot Features.* [Online]
Available at: https://godotengine.org/features/
[Accessed September 2024].

Laoyan, S., 2024. *Agile:Resources:Asana.* [Online]
Available at: https://asana.com/resources/agile-methodology
[Accessed September 2024].

Meta, n.d. *Meta Quest 3.* [Online]
Available at: https://www.meta.com/ie/quest/quest-3/?srsltid=AfmBOopJ4MTFAGcFgB1-4jWyNZWv4ziU8QkXxixa3Zf7eGrzXS-FSaN0
[Accessed November 2024].

SynthVR, 2021. *Steam.com/synthvr.* [Online]
Available at: https://store.steampowered.com/app/1517890/SynthVR/

Virtuoso, n.d. *Virtuoso.com.* [Online]
Available at: https://virtuoso-vr.com/