Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2024

Working in Progress
ThereMelo

Scott Frazier
Ron Nguyen
Farhan Sadiq
Sazzadul Islam
Caleb Myatt

Revision History

Revision	Date	Author(s)	Description
0.1	1.24.2024	RN, SF, FS, SI	document creation
0.2	1.30.2024	RN, CM	editing and adding to content
0.3	2.11.2024	FS, RN, SI, CM	editing and revising
1.0	2.11.2024	FS, RN, SI, CM, SF	Final Version 1.0

CONTENTS

List of Figures

List of Tables

1 Introduction

By taking the user's hands as input, we can convert that into an audible sound. This will allow our user to play it as an instrument, based on the theremin. The theremin is non-physical contact instrument utilizing electromagnetic fields. As the hands interact with those fields, they interfere with them, causing the instrument to produce it's sound. Our product, the ThereMelo, instead using an IR camera to take the user's hands and try to replicate the theremin.

The ThereMelo takes the user's hands relative to the IR camera, and the positioning of the user's preferred hand will determine the application's cursor. This will allow the user to interact with our Graphical User Interface. The GUI will allow the user to select either the lessons mode or the sandbox mode. In lessons, the user will learn how to play the ThereMelo as close as possible to the real life equivalent. In sandbox, the user may experiment with the instrument to however they like. The user's computer will be constantly displaying the environment on their screen, as well as outputting sound for our user.

2 System Overview

The ThereMelo is made out of four layers, each has its interactions with one another. Currently, those layers are the External Layer, Application Layer, User Interface Layer, and Data Layer. The External Layer contains all our the devices that will either take input information from the user or provide output for the user. Our Application Layer will process all information given by the user and generate an output into both External and User Interface. That means sound information will be projected to the user's speakers and the environment will be shown to the user with their monitor. With the Application Layer constantly updating the scene, the User Interface will allow the user to change any settings, such as their preferred hand of choice, lessons, audio output, etc. Finally, the Data Layer will store any user settings that they've changed and any lessons they've completed so they may review for later.

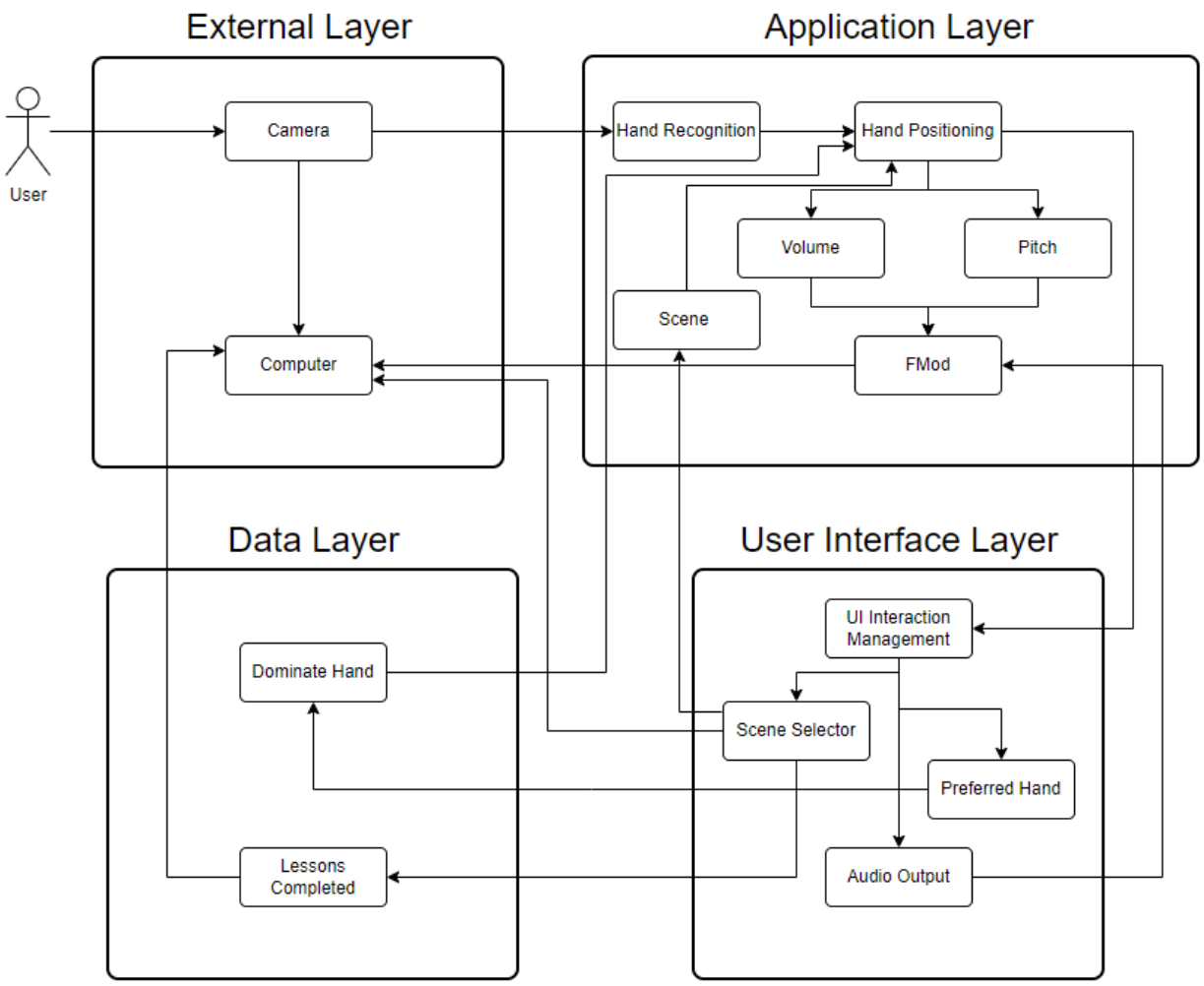

Figure 1: System architecture

3 External Layer Subsystems

3.1 External Layer Operating System

Any operating system capable of running Unity 2022.3.10f1 along with the Ultraleap Tracking application. Linux is not supported at this time.

3.2 Camera Subsystem

The Camera subsystem is a type of hardware, utilizing the Leap Motion Controller. This is the IR camera that will take all user input as data for our application. Only two hands are allowed to interact with the IR camera. Data sent are position and orientations of each hand.

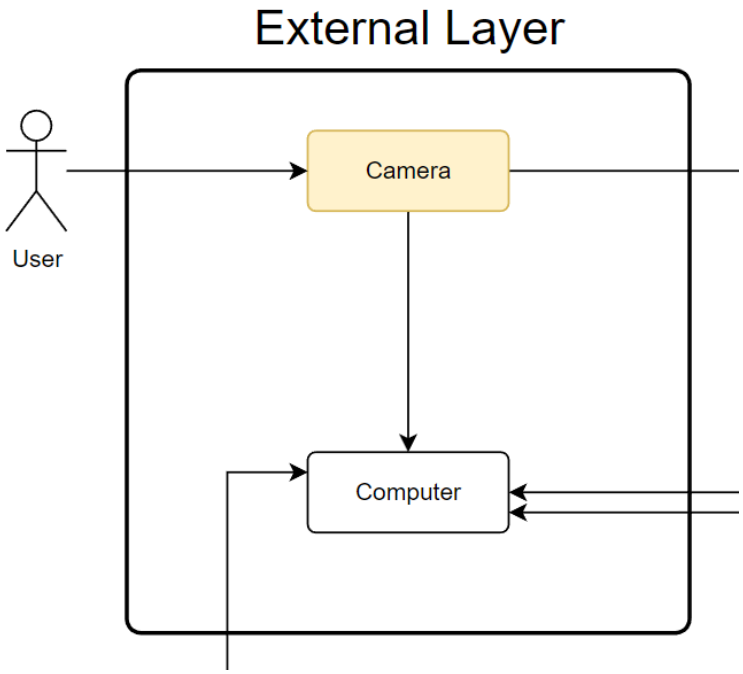

Figure 2: Camera Subsystem

3.2.1 Camera Hardware

Ultra Leap Motion Controller 2, an IR camera used to take a pair of hands as input.

3.2.2 Camera Software Dependencies

Ultraleap's Unity API Plugin, utilized to gather input data to be used for calculations. Ultraleap's Gemini hand tracker, to be able to capture hands from the IR Camera.

3.3 Computer Subsystem

This subsystem is the user's preferred machine to run our application. This will consist of a monitor to display the environment and user's virtual hands. The speakers will provide audible sound for the user to hear the instrument.

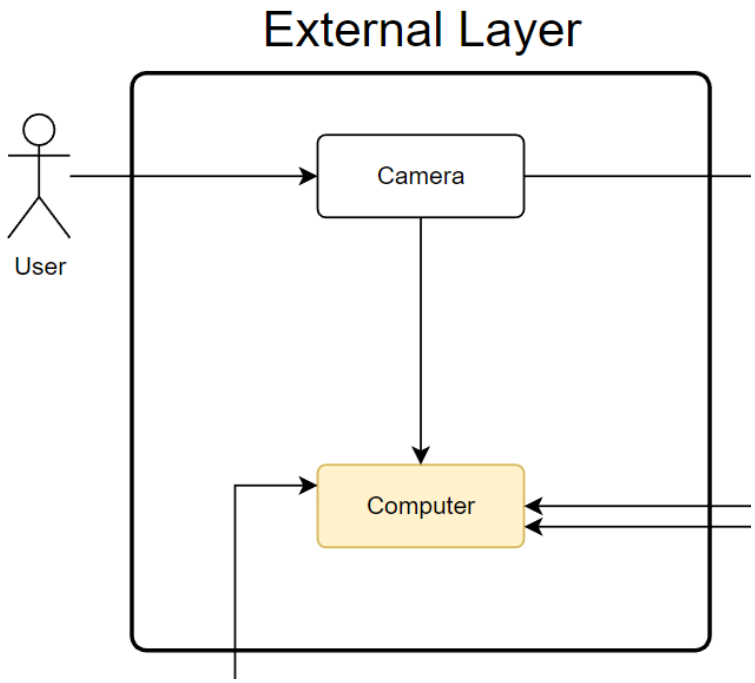

Figure 3: Computer Subsystem

3.3.1 Computer Hardware

An Intel®Core™ i3 processor 5th Gen or better. The device also requires a USB type C connector to be readily available.

3.3.2 Computer Operating System

Windows®10+, 64-bit or Mac operating systems

4 Application Layer Subsystems

4.1 Application Layer Hardware

This subsystem will run on the user's preferred computer.

4.2 Application Layer Operating System

Subsystem will run the operating system the user has.

4.3 Application Layer Software Dependencies

Subsystem will make use of Ultraleap's Unity Plugin,

4.4 Hand Recognition Subsystem

The Hand Recognition subsystem takes what data was being sent from the Camera to be utilized in our Unity project. All matrices relating to positioning and orientation from each hand, and detects if it's a valid hand. If it is, virtual hands will be displayed on screen.

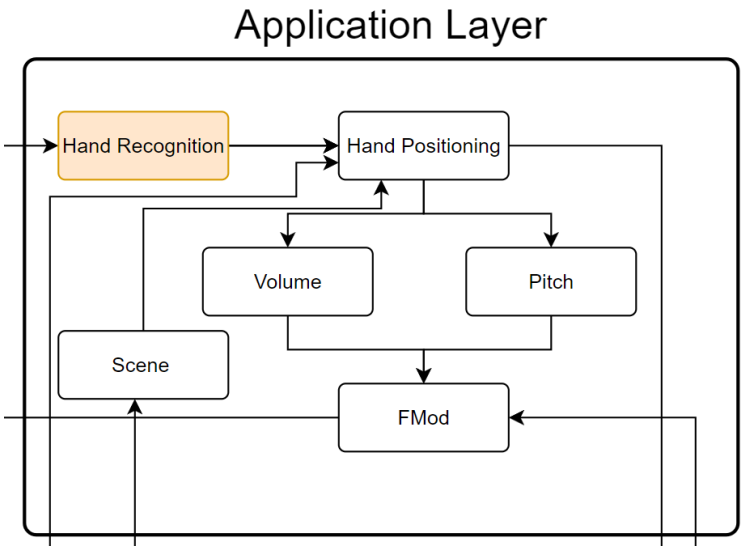

Figure 4: Hand Recognition Subsystem

4.4.1 Hand Recognition Hardware

Subsystem will require the user's preferred computer to run.

4.4.2 Hand Recognition Operating System

Subsystem will run the operating system the user currently has.

4.4.3 Hand Recognition Software Dependencies

Ultraleap Unity Plugin.

4.4.4 Hand Recognition Programming Languages

This subsystem will be utilizing the C# language, for Unity.

4.5 Hand Positioning Subsystem

This subsystem will calculate the hand's position from the corresponding pads. Each hand has their own pads to be calculated from. If the user's preferred hand is the right hand, then that hand should be calculating the magnitude from the hand to the pitch rod. The left hand will do a similar thing, but with the volume pad instead.

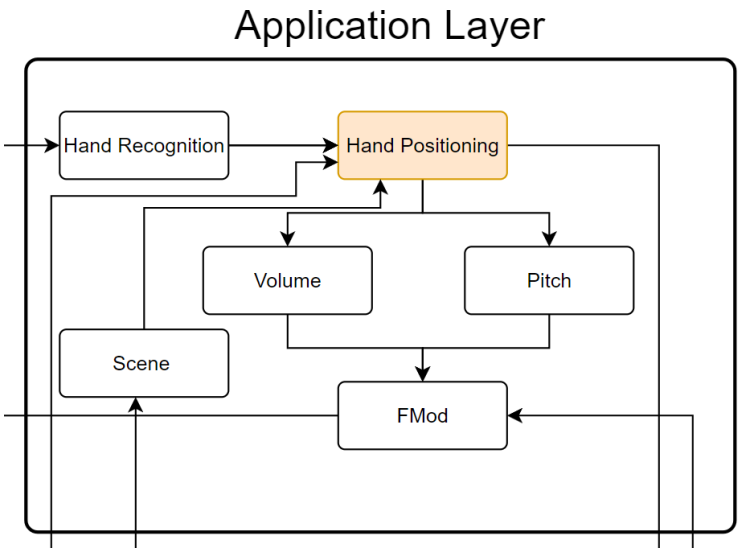

Figure 5: Hand Positioning Subsystem

4.5.1 Hand Positioning Hardware

System will be using the user's computer.

4.5.2 Hand Positioning Operating System

System will be using the user's operating system.

4.5.3 Hand Positioning Software Dependencies

Ultraleap Unity plugin

4.5.4 Hand Positioning Programming Languages

This subsystem will be utilizing the C# language, for Unity.

4.5.5 Hand Positioning Data Processing

With every frame of data from the Camera, it'll be processed within the Leap Motion's Unity Plugin, allowing a visualization of hands in the environment. The default positioning are determined by the a GameObject's base, and any movement captured by the Camera can transform the position or orientation of the virtual hands.

4.6 Volume Subsystem

The volume subsystem takes that magnitude, compare it to the user's normalized position and will change the volume depending how far the user's hand is to the object. This system will also send an event to change the FMod's volume parameter, allowing the user to control the note's velocity.

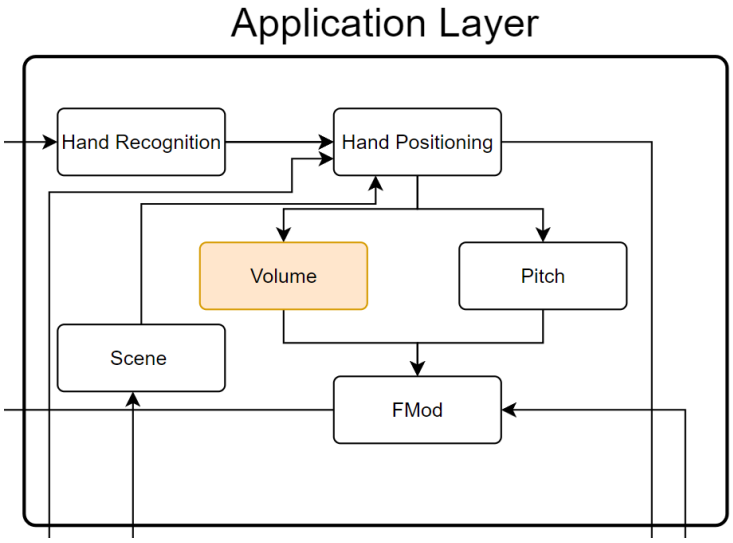

Figure 6: Volume Subsystem

4.6.1 Volume Hardware

User's computer is being utilized for calculation and firing events.

4.6.2 Volume Operating System

Should also be the user's current operating system.

4.6.3 Volume Programming Languages

C# is the language.

4.7 Pitch Subsystem

The pitch subsystem takes that magnitude, compare it to the user's normalized position and will change the volume depending how far the user's hand is to the object. This system will also send an event to change the FMod's pitch parameter, allowing the user to change the note's pitch to represent a specific musical note.

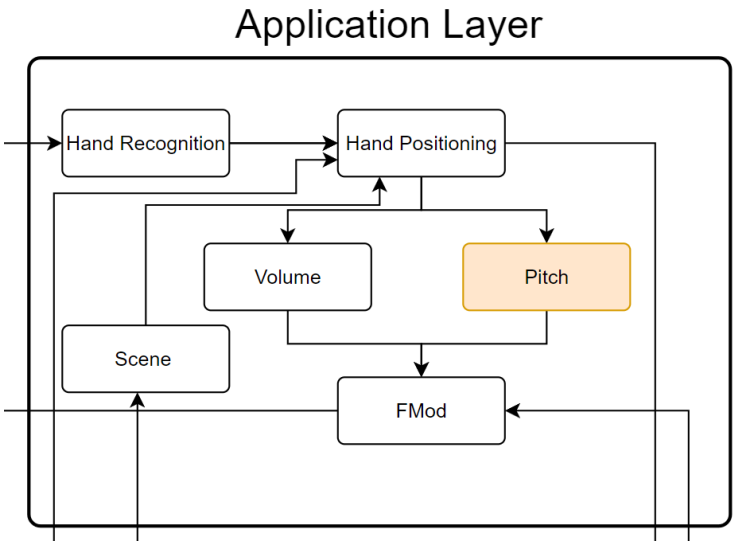

Figure 7: Pitch subsystem

4.7.1 Pitch Hardware

Similar to the Volume Subsystem, this will be utilizing the user's current computer.

4.7.2 Pitch Operating System

The current computer's operating system.

4.7.3 Pitch Programming Languages

C#, to relay parameters to FMod.

4.7.4 Pitch Data Processing

We'll be using FMod's built in digital signal processing and using a Fast Fourier Transform to be detecting the sound's pitch for a digital tuner for the user to use. This is to help the user understand what note they are playing, as well as for us to work with lessons.

4.8 FMod Subsystem

This subsystem handle all audio related instances within our environment. Any events being fired will be directed to the FMod application, which allows us to change automation parameters that will change the note's pitch frequency, and note velocity to simulate a theremin. FMod Studio, which allows us to create audio content for our environment, acts like a digital audio workspace (DAW), allows ease of usage to create the ThereMelo's sound.

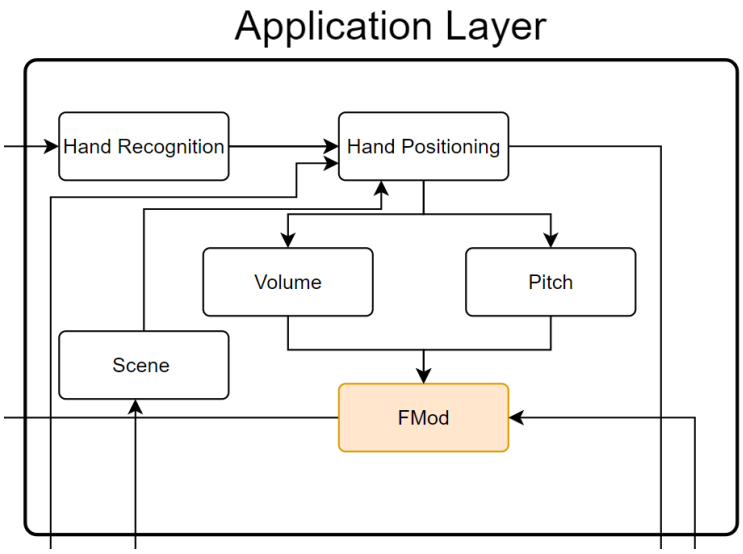

Figure 8: FMod subsystem

4.8.1 FMod Hardware

Hardware for this subsystem should run normally on the user's current computer. Requirements should meet with the Computer Subsystem.

4.8.2 FMod Operating System

This subsystem should be running the same operating system as the user's.

4.8.3 FMod Software Dependencies

FMod studio, required for both Pitch and Volume subsystems. Needed to determine both parameters to edit, and allows for easy editing of sound, sound events, etc. FMod for Unity, to integrate studio to our environment. This allows us to fire custom audio events, as well as handle sound for the project. Required for both Pitch and Volume subsystems.

4.8.4 FMod Programming Languages

C# to fire our events inside the environment.

4.9 Scene Subsystem

The scene is where the user will be able to see the hands for the theremin and see how there hands effect the sound. This also allows us the put in a tutorial for a user and allow a sandbox mode. This is from the Unity which allows us to change what is state we are in.

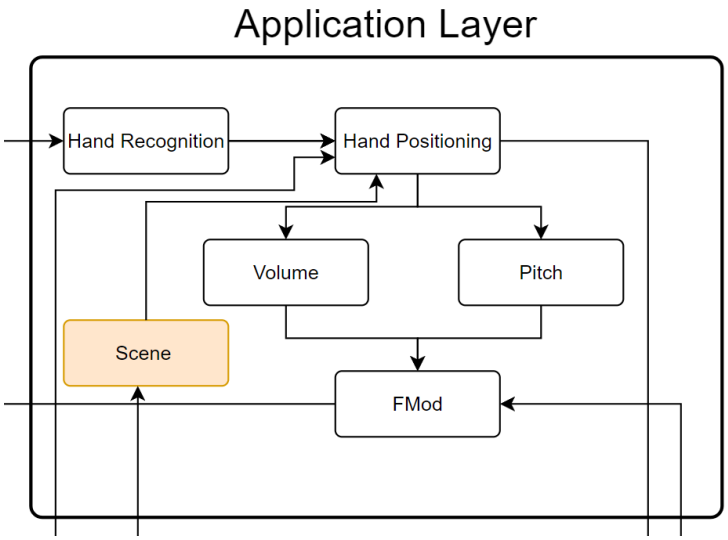

Figure 9: Scene subsystem

4.9.1 Scene Hardware

The scene will be running will be running on the user computer.

4.9.2 Scene Operating System

The current computer's operating system.

4.9.3 Scene Software Dependencies

We will be just needing Unity to allow us to use our scene created in Unity.

4.9.4 Scene Programming Languages

We will be using C#

4.9.5 Scene Data Structures

We will be taking information from the user hand position to allow them to change the scene. From this we want to change where the hand positioning to the new scene.

5 User Interface Layer Subsystems

5.1 User Interface Layer Operating System

Subsystem will run the operating system the user currently has.

5.2 User Interface Layer Software Dependencies

We will be using Ultraleap Unity Plugin to allow us to push buttons to work with a UI. A description of any software dependencies (libraries, frameworks, etc) required by the layer.

5.3 UI Management Subsystem

This subsystem will allow for the user to be able to control what scene they are on, preferred hand, and how loud the audio should be. This is the main way the user will be able to change how the program works.

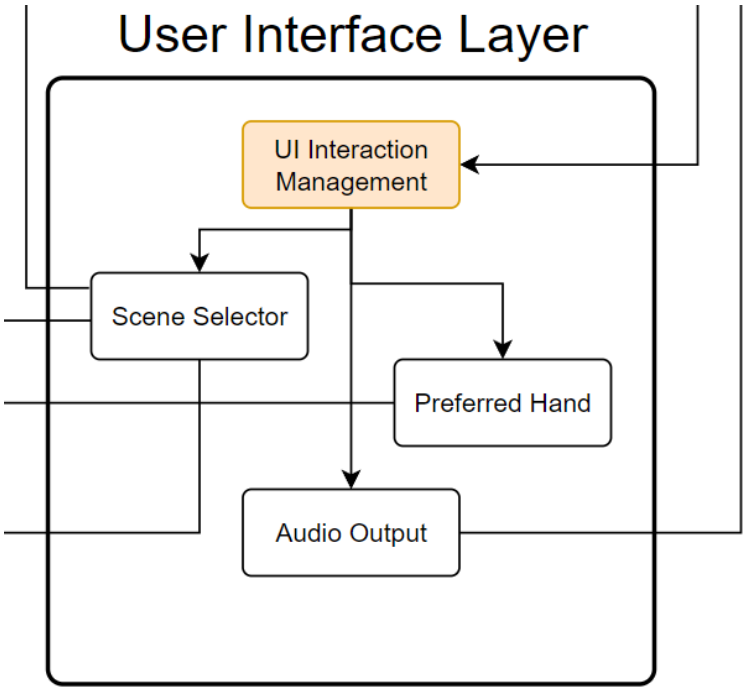

Figure 10: UI Management Subsystem

5.3.1 UI Management Hardware

This is where all the information sent by the hand positioning to allow the program to open the UI management. We have it were the user flips over there wrist and it will pull up. With the other hand we allow the user to have a few options like changing the preferred hand.

5.3.2 UI Management Operating System

Should be the user's current operating system

5.3.3 UI Management Software Dependencies

We would be using the Ultraleap Unity Plugin to allow us to see which button was pressed

5.3.4 UI Management Programming Languages

We will be using C#

5.3.5 UI Management Data Structures

We would be receive information from where the hand are which is taken care of from Ultraleap Unity Plugin. A description of any classes or other data structures that are worth discussing for the subsystem. For example, data being transmitted from a microcontroller to a PC via USB should be first be assembled into packets. What is the structure of the packets?

5.3.6 UI Management Data Processing

5.4 Scene Selector Subsystem

This would be a button which would allow the user to change over from the current scene they are in to another.

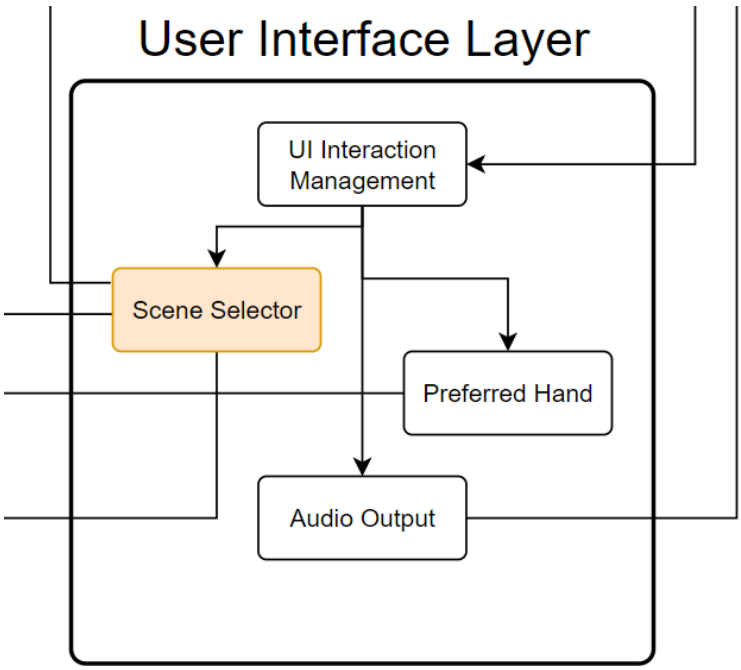

Figure 11: Scene Selector Subsystem

5.4.1 Scene Selector Hardware

This subsystem will be using the user current computer.

5.4.2 Scene Selector Operating System

This subsystem will be using the user operating system.

5.4.3 Scene Selector Software Dependencies

Unity has an inbuilt functions for this so we need to be using it.

5.4.4 Scene Selector Programming Languages

We will be using C#

5.5 Preferred Hand Subsystem

This allows the user to interaction with the UI interaction management and change which is the preferred hand. The preferred hand will relate to which hand will be producing the pitch or the volume.

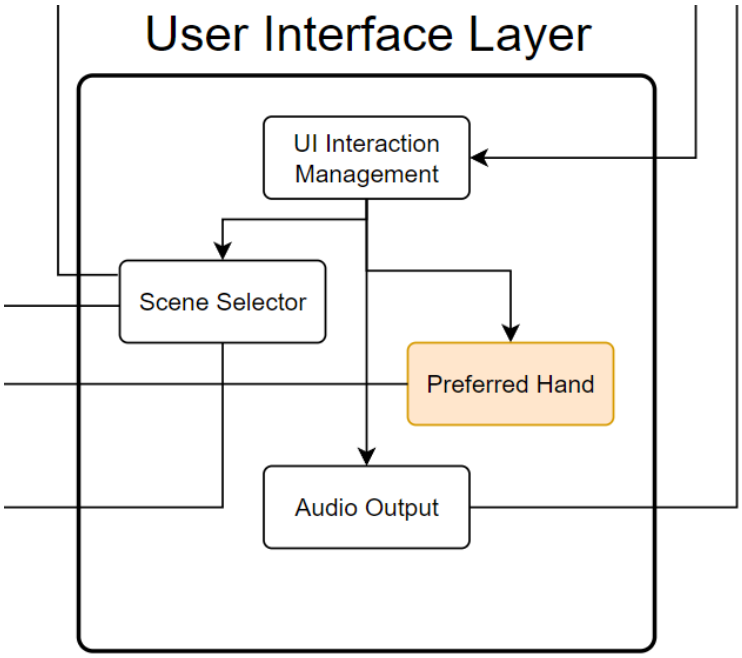

Figure 12: Preferred Hand Subsystem

5.5.1 Preferred Hand Hardware

This subsystem will be using the user's current computer.

5.5.2 Preferred Hand Programming Languages

The program will use C#

5.5.3 Preferred Hand Data Structures

We will have a check for which is the preferred hand for the end user. This will be sent over to where we keep our data and will be used constantly to check which hand is being used.

5.6 Audio Output Subsystem

The audio output allows the user to change how loud the sound from the speakers is. We wanted a way to allow the user to easily change how loud the theremin is.

5.6.1 Audio Output Hardware

This subsystem will be using the user's current computer.

5.6.2 Audio Output Operating System

This subsystem will be using the user's current operating system.

5.6.3 Audio Output Programming Languages

The program will use C#

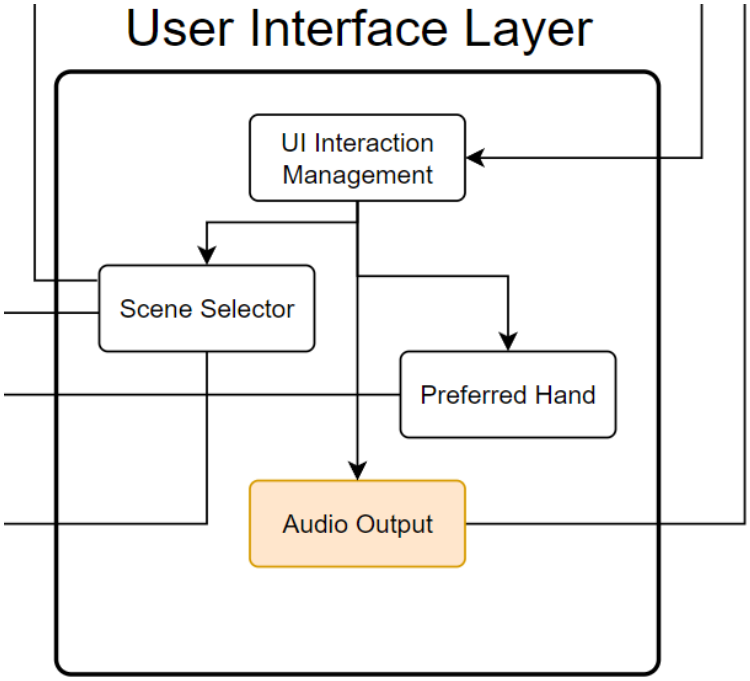

Figure 13: Audio Output subsystem

5.6.4 Audio Output Data Structures

This subsystem receives signals from UI interaction management and sends the data to FMod to generate the audio output.

5.6.5 Audio Output Data Processing

Fmod is mostly responsible for the processing of the data that it receives from the UI interaction management. The data is processed and as an output, it generates the audio.

6 Data Layer Subsystems

6.1 Data Layer Hardware

Hardware should be the user's current system.

6.2 Data Layer Operating System

This sub system will run the operating system the user is currently using.

6.3 Data Layer Software Dependencies

We will be using the data that has been processed and stored in the database as well as the data that has been collected from the application layer.

6.4 Dominate Hand Subsystem

This is part of the data layer. The UI would allow the user to choose their preferred hand, and this data would be stored as the user's dominant hand. The data for the dominant hand will also determine the hand positioning.

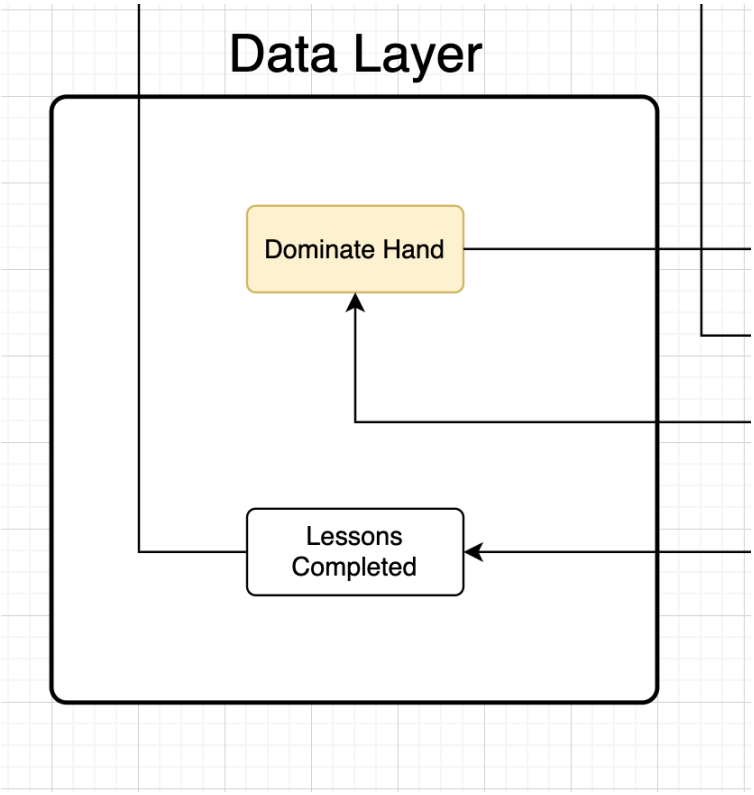

Figure 14: Preferred Hand subsystem

6.4.1 Dominate Hand Hardware

This subsystem will be using the users current computer.

6.4.2 Dominate Hand Operating System

This sub system will run the operating system that the user is currently using.

6.4.3 Dominate Hand Software Dependencies

The camera is functional and it detects the hand gestures accurately. The user has two hands (left and right) and they know which is their dominant hand.

6.4.4 Dominate Hand Programming Languages

We will be using C#.

6.4.5 Dominate Hand Data Structures

The dominant hand is the subsystem that is responsible for the data regarding the preferred hand of the user. The UI detects the dominant hand and it is related to the hand positioning/gestures, in the application layer.

6.4.6 Dominate Hand Data Processing

This is the layer where the data regarding the dominant hand of the user is processed. The hand gestures is detected in the application layer and data is processed and passed on to this sub system of the data layer.

6.5 Lessons Subsystem

This is part of the data layer. The UI would allow the users to view the lessons they have completed. This will be displayed on the screen. The data regarding the user's lesson completion will also be stored in this subsystem.

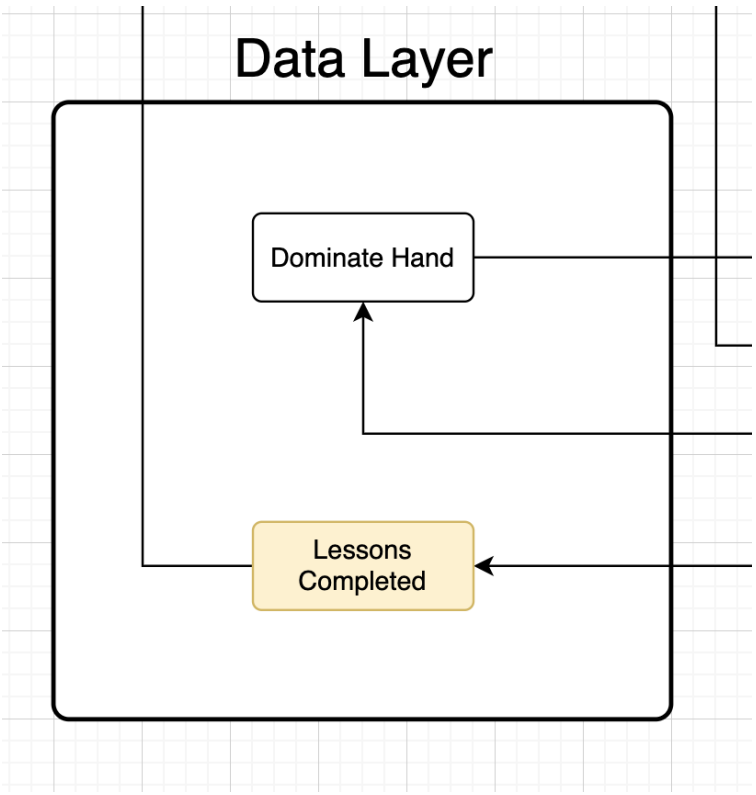

Figure 15: Lessons subsystem

6.5.1 Lessons Hardware

This subsystem will be using the users current computer.

6.5.2 Lessons Operating System

This sub system will run the operating system that the user is currently using.

6.5.3 Lessons Software Dependencies

The user acknowledges that lessons will be either completed or uncompleted. The user knows lessons are repeatable.

6.5.4 Lessons Programming Languages

We will be using C #.

6.5.5 Lessons Data Structures

Lessons completed is the subsystem that is responsible for the data regarding the lessons that the users have attempted to complete or completed. It is also connected to the external layer as the data gets displayed on the computer screen when the user wants to view it.

6.5.6 Lessons Data Processing

Any data regarding the lessons that the user have attempted to complete or completed will be processed and stored in the database and the required data can be viewed on the screen whenever the user checks for it.

7 Appendix A

REFERENCES

[1] David Abbott. Using fmod studio with unity. 2018.

[2] Daniel Bachmann, Frank Weichert, and Gerhard Rinkenauer. Evaluation of the leap motion controller as a new contact-free pointing device. *Sensors*, 15(1):214–233, 2014.

[3] Nenad Breslauer, Irena Galić, Mihael Kukec, and Ivan Samardžić. Leap motion sensor for natural user interface. *Tehnički vjesnik*, 26(2):560–565, 2019.

[4] Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, and Jaka Sodnik. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, 14(2):3702–3720, 2014.

[5] Robert McCartney, Jie Yuan, and Hans-Peter Bischof. Gesture recognition with the leap motion controller. 2015.

[6] Ciarán Robinson. *Game Audio with FMOD and Unity*. Routledge, 2019.

[7] Markus Sukoinen. Audio implementation methods in unity. 2019.

[8] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[3] [4] [2] [5] [8] [1] [7] [6]