DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Inspired by the ultimate perspective - Development of a piano learning application using augmented reality

Lukas Michael Schneidt

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Inspired by the ultimate perspective - Development of a piano learning application using augmented reality

Inspiriert von der ultimativen Perspektive - Entwicklung einer Klavier-Lernanwendung unter Verwendung von Augmented Reality

Author:	Lukas Michael Schneidt
Supervisor:	Klinker, Gudrun Johanna; Prof. Dr.
Advisor:	Sven Liedtke
Submission Date:	15.11.2019

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.11.2019 Lukas Michael Schneidt

Acknowledgments

Abstract

Having to look at sheet music while playing a new piece of music on the piano can be very tiring and slows down the overall learning process and motivation of the student. The representation of sheet music as we know it today has been invented hundreds of years ago and has not changed a lot ever since, despite the increase in technological possibilities. Augmented Reality (AR), being one of them, offers a natural way of assisting the user by scanning the environment and adding virtual supplements to their real perception. The goal of this thesis is to develop an augmented reality application for the head-mounted display Magic Leap 1, which offers a way to make it easier to learn new pieces of music or techniques. Highlighting specific properties of the keyboard concerning a specified key, or indicating which key should be played next can be a helpful asset. The application was implemented for the use with a MIDI keyboard and has been implemented using Unity3D. It was evaluated in terms of intuitiveness and usability, and it has been found that the AR piano instructor could generate interest in music theory, and it also motivated the user to learn to play the piano.

Contents

1. Introduction

"I haven't understood a bar of music in my life, but I have felt it."
Igor Stravinsky

Music has always been an essential part of humanity, regardless of the origin or culture of a human being. Nearly every society used music to express itself, therefore having the ability to play an instrument has always been covetable and brought joy to the listener and prestige to the player [Boe09].

After the invention of the piano by Bartolomeo Cristofori around the year 1700, the piano was widely spread amongst royalty and prosperous families and therefore became a symbol for wealthiness. After it became more affordable, it spread rapidly among the middle class, which not only used it as a solo instrument but also to accompany the singing in groups and became a commonly used element in people's daily life [Hil88].

The notation of music, as well as being able to read musical scores, has been a skill that requires a lot of background knowledge about musical theory, which is not necessary to make music. As this is a hurdle for most piano student beginners, an alternative approach of notating music would help the student to make it more intuitive to "read" the music.

With augmented reality becoming more and more accessible to the general public, a wide range of possibilities to improve the way we "read" the music emerged. With applications like *HoloKeys* [HA17] or *Piano AR* [Hua+11], researchers have offered solutions for an augmented reality piano teacher that is using an alternative and promising way of displaying the music, by using head-mounted displays to show the notes on the keys of the real piano.

The goal of this thesis was to (1) develop an augmented reality application for the head-mounted display "Magic Leap One" built by Magic Leap and to (2) find a way to maximize the feeling of making music by minimizing the required knowledge.

2. Related Work

2.1. Computer Based Piano Teaching Applications

Since computers became more and more accessible to the general public, developers started to create applications with the aim of teaching the user how to play an instrument. One of the first attempts of such software is called "Piano Tutor" and has been introduced to the public in 1990 [Dan+93].
"Piano Tutor" is an interactive computer system, which requires a computer with a display, an electronic piano with a computer interface, a videodisc player, a second display, and an audio mixer to output the sounds of the piano as well as the "instructor's" voice. It teaches in short lessons that address basic concepts of piano performance skills, as well as music theory concepts like notation, pitch, rhythm, and structure of music, by giving the student different tasks and giving feedback or suggestions if the student makes mistakes.

However, the idea of music teaching software already emerged years before that. In 1988, Taylor wrote in "Design For Arts in Education", that music teaching software *"... should take full advantage of the hardware's potential in order to serve the artist well ..."* [Tay88].
In fact, these systems have been drastically improved over the last 40 years. Since back then, the hardware was minimal compared to today's possibilities; the software's potential also increased drastically.

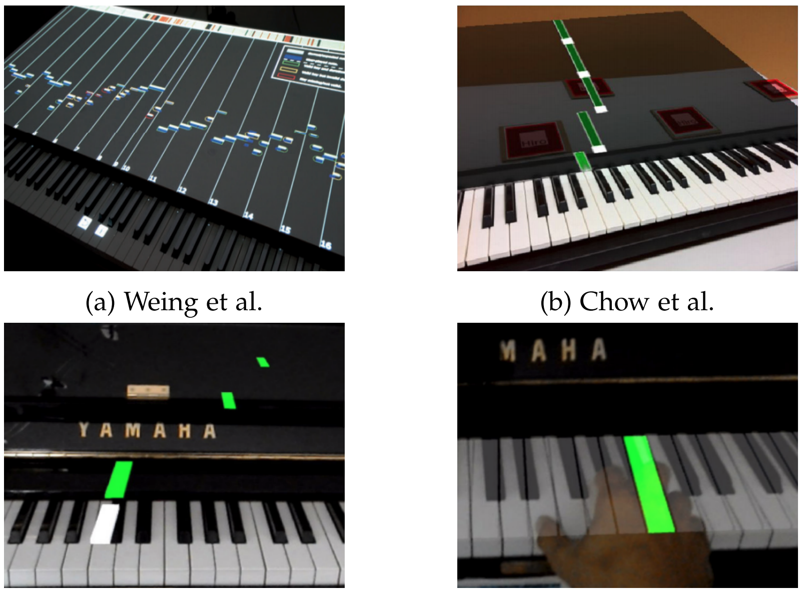
2.2. AR based piano instructors

There have been a few attempts to create an augmented reality based piano teaching application. Up-and-coming solutions have been introduced by Weing et al. in their work *PIANO: enhancing instrument learning via interactive projected augmentation* [Wei+13], by Hackl and Anthes in their work *HoloKeys - An Augmented Reality Application for Learning the Piano* [HA17] and J. Chow et al. with their work *Music Education using Augmented Reality with a Head Mounted Display* [Cho+13] which will be discussed in this section in terms of visualization of the notes, application design, and tracking capabilities.

2.2.1. Visualization

Weing et al. and Chow et al. came up with a similar approach to visualizing the notes the student has to play. They are shown as two-dimensional rectangles, which are moving towards the corresponding key of the real piano. One of the two visualization approaches presented by Hackl and Anthes, the so-called "Beatmania" visualization, also comes close to the proposed presentation. However, there are a few differences. While the displayed notes in Hackl and Anthes work are also displayed on the corresponding key, while they have to be played, Chow et al. offer a solution, where the keys are shrunk at the rear end of the key. This way, the hand of the user is not covered by the note.
Weing et al. also display the notes on the keys while they have to be played. However, their approach is fundamentally different, as all UI elements —– like the notes —– are projected from above instead of being displayed in a head-mounted display as in the other approaches. Therefore, the notes will not cover the hand, but it will be displayed on the back of the hand. A significant downside of this representation is the limitation of usable pianos. This approach is meant to be used with keyboards or pianos with a vertical plane at the same height of the keys, in order to display these notes correctly. The approach of J.Chow et al. and Hackl and Anthes are making use of HMDs, however, while Hackl and Anthes are using the *Microsoft HoloLens* [1], J.Chow et al. are using the not optical see-through HMD *Trivisio ARvision-3D HMD1* [2], which makes the whole application feel less natural.
Hackl and Anthes came up with an alternative visualization without moving notes, which they call the "Instant Approach". A key that should be played is highlighted and will be displayed as soon as the user has to release the note. The student can, therefore, observe how the notes should be pressed in real-time. However, this approach aims at advanced players, as it is nearly impossible to play along with the "instant" representation.

[1]https://docs.microsoft.com/en-us/hololens/hololens1-hardware
[2]https://est-kl.com/ru/manufacturer/trivisio/arvision-3d-hmd.html



(a) Weing et al.

(b) Chow et al.

(c) Hackl and Anthes: "Beatmania"

(d) Hackl and Anthes: "Instant"

Figure 2.1.: Visualisation approaches

2.2.2. Application design

The underlying idea of all of the presented approaches is to teach a piano student a particular piece of music. Other concepts of playing the piano (like improvisation) have not been implemented.
In order to give the user feedback about their attempts to play a track, some ideas have been realized.
J.Chow et al. compares the time the user hits a key with the time the key should be hit and gives feedback whether they pressed it at the exact correct time (perfect), if they nearly hit it at the correct time (good), if they played it way too early or too late, or if the missed the note at all. However, this does not give any feedback regarding the velocity or the fingering.
Weing et al. introduced three different play modes, which are: Listen, Practice, and Play. The "Listen" mode allows the user to listen to the piece of music, while the notes are being displayed. In the "Practice" mode, the piece stops whenever the user played a wrong note and will not continue until they played the correct one. The "Play" mode allows the user to adjust the tempo of the piece and forces them to play at a consistent speed. Feedback for correct/false notes is given in real-time, by coloring the notes in green or red.

2.2.3. Tracking

J. Chow et al. and Hackl and Anthes both use marker-based tracking algorithms to track the keyboard. While J.Chow used the *NyARToolkit*[3] to find the pose of their markers, Hackl and Anthes used the AR tracking library *Vuforia*[4] with *Unity*[5]. The tracking algorithm of J.Chow et al. used markers with big black borders, which are similar to the ones we used in this work, as described in subsection 4.2.1.
The *P.I.A.N.O* application does not need any tracking capabilities, as it is made for a specific keyboard. This makes the representation very robust; however, it significantly limits the portability of such a tool. Huang et al. have made another keyboard tracking approach [Hua+11]. This marker-less tracking algorithm can detect the exact pose of a keyboard and the position of the fingers on the keyboard. However, it requires the whole keyboard to be in the field of view of the camera, which is not guaranteed when playing with full-sized keyboards.

2.3. Augmented reality Headsets

The market for head-mounted augmented reality displays, also called HMDs, is changing very rapidly. New manufacturers with projects to be funded come and go. This section should give a rough overview of suitable HMDs for this project.

2.3.1. Microsoft Hololens 2

The **Microsoft HoloLens 2**[6] is one of the most known HMDs currently available on the free market. With a resolution of 2K per eye and a field of view of 52 degrees, it is also one of the most promising devices at this time. It offers gaze tracking, gesture input, voice support as well as spatial sound to adapt to the human as its user. With a weight of 566g it is more cumbersome as most of its competitors. While actively using the device, its battery lasts up to 3 hours. Applications for the HoloLens 2 can be developed using Unity 5.4 and Visual Studio 2015. It was released in November 2019

[3]https://nyatla.jp/nyartoolkit/wp
[4]https://developer.vuforia.com/
[5]https://unity.com/
[6]https://www.microsoft.com/en-us/p/microsoft-hololens-development-edition/8xf18pqz17ts?activetab=pivot:overviewtab

and is available for around 3500 USD.

2.3.2. Magic Leap One

Magic Leap's **Magic Leap 1**[7] offers a resolution of 1.3 million pixels per eye and a field of view of 50 degrees. Just as the HoloLens 2, it uses spatial sound, head, eye, and hand tracking to allow for a natural feeling. It only weighs 316g, which is comparable with most over-ear headphones, and its battery lasts up to 3.5 hours during active usage. In order to develop for the Magic Leap 1, one can use Unity, Unreal Engine or LuminRuntime. With a cost of 2295 USD, it is much cheaper than Microsoft's HoloLens 2. The Magic Leap 1 is available since August 2018.

2.3.3. Smartphone-powered AR Headsets

There is a broad range of smartphone-powered AR Headsets on the market that allow the users to use their smartphone and mirror its contents on a see-through surface in front of the eyes. This not only increases the possible field of view (e.g., *Tesseract Holoboard*[8] 90 degrees, Mira Reality's *Prism Pro*[9] 60 degree). As most of the technology that is needed for the augmented reality experience is built in today's smartphones, these devices are a lot cheaper than the standalone products (*Tesseract Holoboard*: 149 USD; *Prism Pro*: 99 USD).

[7]https://www.magicleap.com/magic-leap-1
[8]https://myholo.io/index.html
[9]https://mirareality.com/hardware/

3. Application Design

3.1. Keyboard Characteristics

As there are different standards of naming the keys on a keyboard, and as there are different sizes of pianos in the world, we first have to clarify which kind of keyboard is being used in this thesis and how the keys are represented.

We will use a keyboard with 88 keys, which is typical for concert pianos. The keyboard holds 7 octaves â 12 keys plus three keys added at the lower end and one key added at the higher end. An octave implements the chromatic scale, and its layout is the following: C-C♯-D-D♯-E-F-F♯-G-G♯-A-A♯-B. Each octave starts at C and ends at B, and its keys are indexed with a number starting at 0 (for the last three notes of the lowest octave A-0. A♯-0, B-0) and ends at 8 (for the first note of the highest octave C-8). These 88 keys are split into 52 white keys and 36 black keys, while the black keys represent the sharp/flat notes. Furthermore, the black keys are smaller and are raised compared to the white keys. The white keys have almost the same size and are adjacent to each other.

Because of this structure, and because of different sizes of keyboards, we will describe the size of a key relative to the size of an octave.
A white key has a width of 14.29% of one octave, while a black key has a size of 6,06% of an octave. These measures will be relevant during implementation. The five black keys of an octave are placed between the first and second, the second and third, the fourth and fifth, the fifth and sixth, and the sixth and seventh white key. This structure

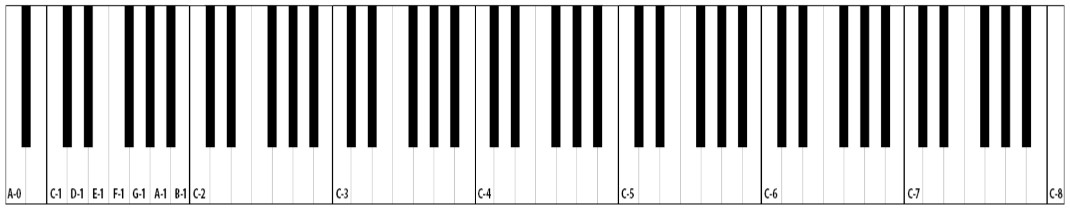


Figure 3.1.: Keyboard Representation used in this thesis

is the same for all octaves. A simple representation of a keyboard with the described naming can be found in figure 3.1)

3.2. Navigation

In order to navigate through the app or to adjust settings, there needs to be some kind of interaction with the app. These four ways of controlling the app have been examined.

- Controller
- Hand Gestures
- Voice Commands
- MIDI

The Magic Leap comes with a handheld controller, which can be connected wireless with the glasses. This controller is used for most Magic Leap applications and for navigation between apps. However, as both hands are needed to play the piano, it appears to be cumbersome to use it.
Both navigation through hand gestures and navigation through voice commands take care of this problem. However, both approaches would not feel natural. Some further feedback about these approaches can be found in chapter 5.
That is why the latter approach was elected. The is fully controllable with the MIDI keyboard. The settings can be adjusted by hitting some specific keys or chords, and the keyboard can be used as some kind of user interface e.g., hitting any key of the middle octave, which will lead the user to the main menu. The following subsection will explain the navigation method that has been used.

3.2.1. Navigation via MIDI-Keyboard

Three different special keyboard commands are necessary in order to control the app.

- **"Deny"** : This command triggers an action if the lowest key on the keyboard (A-0) is pressed. This action is used if the user wants to deny something.
- **"Accept"** : This command triggers an action if the highest key on the keyboard (C-8) is pressed. This action is used if the user want's to accept something.
- **"Go Back"** : This command triggers an action if the highest key (A-0) and the lowest key (C-8) are pressed simultaneously. This action is used to go back to the previous state and is available at all times.

3.3. Modes

There are four different play modes and one main menu.

3.3.1. Home

The "Home" scene shows the four middle octaves of the keyboard, indicated by the names of the C-keys. Pressing any of the keys with the same color will load the mode with the matching color (e.g., pressing E-3 will load the Improvisation mode).

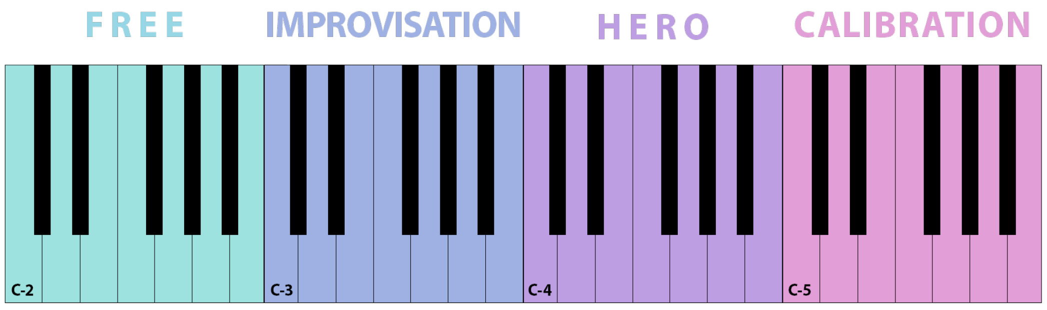


Figure 3.2.: Selectable play-modes in "Home" scene

3.3.2. Calibration

The first scene is the calibration mode. The application launches in this mode and guides the user through the calibration process. During this process, the location of the piano is determined by getting the position of two marker points on the keyboard by a tracking system.
This is necessary in order to print the virtual game elements directly onto the real keys. The game will always launch in this mode, as the keyboard has to be calibrated before the other modes can be used. Like the other modes, the calibration mode can be called at any time from the home screen to re-calibrate the keyboard.

3.3.3. Free

In the "FREE" mode, the player does not get any specific tasks. The purpose of this mode is to give the user visual feedback in addition to the auditory feedback that

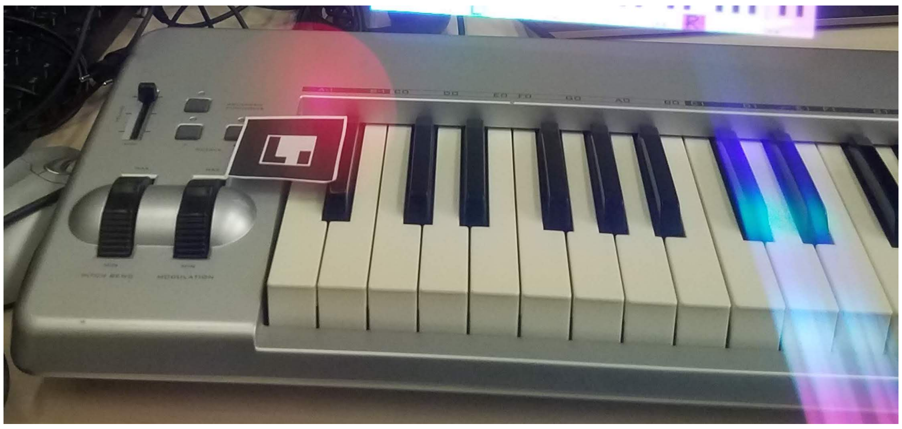


Figure 3.3.: **Calibration:** The red ball around the marker indicates that the marker has been detected

comes from the piano itself. This visual feedback should help the player to memorize the played notes better.

3.3.4. Improvisation

By making it easy for the user to play along to a piece of music, this mode is intended to arouse interest in the theory behind the music (and not the other way around, as it is taught in most cases).
To emphasize the importance of this mode, one needs some knowledge of music theory, which shall be given in the following.

Music theoretical background

The blues scale was chosen for this mode, as blues is considered the forerunner of both jazz and rock and had a significant impact on today's music. As it is so deeply involved in music as we know it today, its sound is highly relevant to musicians and should be internalized by the piano student.

The characteristic sound of blues results from the use of a minor-based scale over the top of major-based chords.
The so-called "Blues-Scale" has six notes, which makes it differ from most other scales,

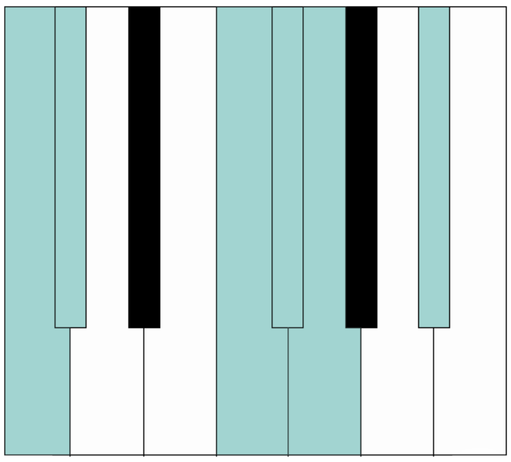


Figure 3.4.: C Blues-Scale

which have seven notes. It is the pentatonic scale with one more note, which is known as the "blue note" (See the C "Blues-Scale" in figure 3.4). This "blue-note" is the flattened fifth in the case of the minor pentatonic and the flattened third in the case of the major pentatonic. On non-keyboard instruments, like the guitar, it is played slightly lower. In blues music, one would call this flatness a "blue feeling". The origin of the blue-note is unknown.

Just like the pentatonic, the notes of the "Blues-Scale" can be played to a blues in the matching key at any time, which makes it one of the most used scales for improvisation. The player can choose between the major blues scale, or the relative minor blues scale (e.g., for a blues in C-Major one can use the C-Major, or the A-Minor blues-scale).

This mode gives a glance at what can be done in terms of improvisation. As an example, it highlights a scale that fits a previously determined key. That means, if the user wants to highlight the keys of a C-Major blues scale, he has to press any C on the keyboard. The user can then play along a blues backing track in the key he selected, by playing any of the highlighted keys, without having to think about any music theory.

Improvisation mode

The "Improvisation" mode highlights any key that is part of the selected "Blues-Scale". The user can choose which key he wants to improvise. After selecting the key, the

app will highlight all keys of the relative minor blues-scale. The keys of the relative minor pentatonic are highlighted in violet, while the blue-note is highlighted in blue to emphasize the harmonic peculiarity of this key.
If necessary, the user can go back to the key selection menu, by pressing "Go Back".

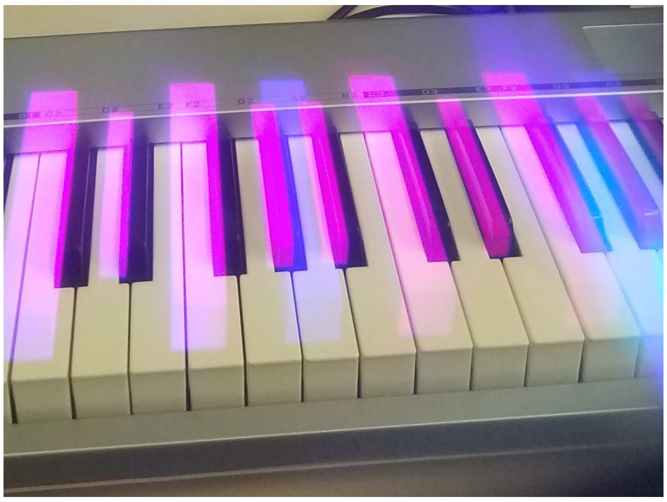


Figure 3.5.: **Improvisation mode:** Pink keys indicate that they belong to the xy pentatonic; the blue key indicates the blue-note

3.3.5. "Piano Hero"

The name of this mode is derived from a popular mixed reality game called "Guitar Hero". In this game, the player needs to hit a button on a controller at the correct time in order to play a melody. This mode uses the same concept and extends it on a larger set of buttons: the keyboard.
The task in this mode is to hit a key as soon as a white box that falls from above hits this note. The player should keep the key pressed as long as the white box touches the key. As an example, the melody of the song "Au clair de la lune" was used. The composer as well as the lyricist of this song are unknown.
This piece of music is often taught to beginners learning an instrument, that is why this simple melody has been chosen in this place.
With this mode, an alternative approach to visualizing the notes shall be given. In figure 3.6 one can find the scores of this song as compared to the "Hero" visualization in figure 3.7

The advantage of the AR approach is that one gets the information right at the spot

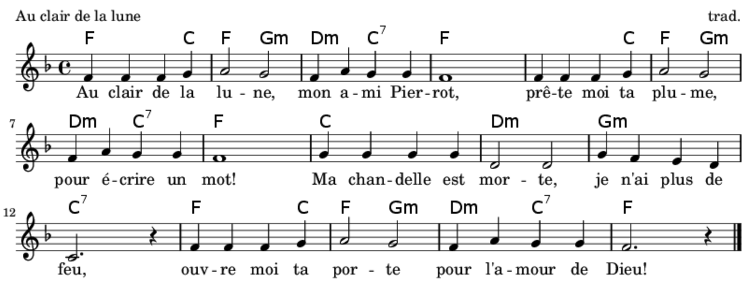

Au clair de la lune
trad.
F C F Gm Dm C7 F C F Gm
Au clair de la lu - ne, mon a - mi Pier - rot, prê - te moi ta plu - me,
Dm C7 F C Dm Gm
pour é - crire un mot! Ma chan - delle est mor - te, je n'ai plus de
C7 F C F Gm Dm C7 F
feu, ouv - re moi ta por - te pour l'a - mour de Dieu!

Figure 3.6.: Au Clair de la Lune[1]

where one needs it, while in classical scores, one still needs to translate the encoded length information, as well as the pitch information of a note.

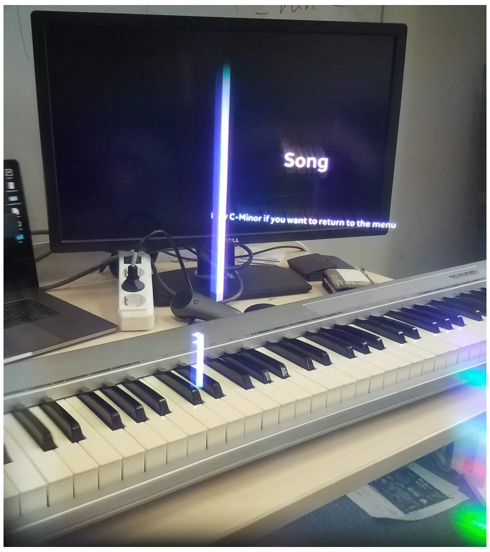


Figure 3.7.: **"Hero" mode:** The white box indicates a note that has to be played as soon as it hits the key

[1]https://bit.ly/3d45vZo

4. Implementation

The technical implementation of the presented features is discussed in this chapter. The implementation of the application can be split into the four tasks Tracking, Keyboard, MIDI, and Application Model. As this application has been created with Unity, a short description of the Unity-methods that have been used shall be given.

Unity basics

Unity works with C# scripts, which derive from the *MonoBehaviour* class of the Unity Engine. A script of this kind can be attached to a *GameObject*. After the initialization of the *GameObject*, the *Start()* method of the script is called, which can be used to initialize its properties. The *MonoBehaviour* class offers an *Update()* method, which is called each frame, to update the properties of this class or to follow some perform specific actions. Within a Unity Script that has been attached to a *GameObject*, the position, as well as the orientation of the *GameObject* can be edited through the *transform.position* and *transform.orientation* property. A distance value of 1.0 corresponds to a distance of one meter in the real world.
Unity allows to build for Magic Leaps *Lumin OS*, and it lets the developer use the Lumin SDK, which is necessary to build an app for the Magic Leap One.

4.1. Application Model

Some data needs to be available across all scenes and must not be lost when switching to another scene. This mainly applies to the data that is needed to calculate the position of the keyboard. The *ApplicationModel* class, therefore, contains the three Vector3 variables *leftMarkerPos*, *rightMarkerPos*, and *scaleVec*, which represent the position of the markers as well as the scale of one single key. The orientation of the keyboard is stored in *orientation* as a Quaternion. It also offers three bools, which tell whether the markers have been found (*foundLeftMarker* and *foundRightMarker*) or not and whether the calibration process has been finished (*calibrated*).
In order to apply the correct color scheme for the keyboard in each scene, the string

isScene contains information about the current scene.

4.2. Tracking

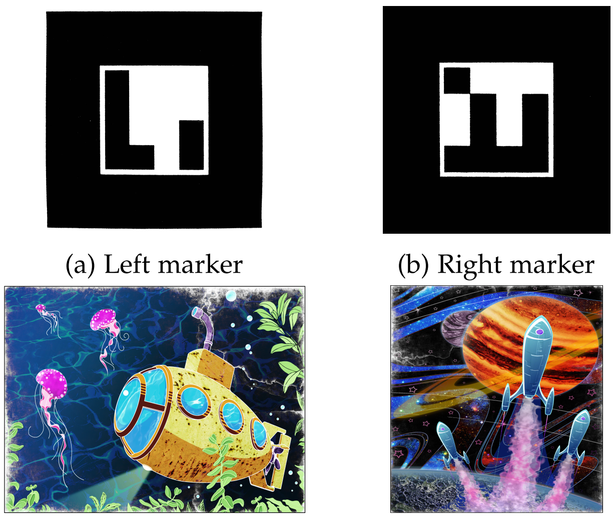
The Lumin SDK offers an image tracking system that uses the sensors of the headset to find custom-defined targets. When a target, which is a two-dimensional planar image, was found, the tracking system provides the position and orientation of this target in game-coordinates.
The tracking system allows the developer to track multiple objects at the same time; in fact, the default value of trackable objects is 25. However, this application implements a straightforward algorithm to retrieve the position, orientation, and scale of the piano, which uses only two targets. The trackers for these targets are mutually exclusive to minimize computing power.

4.2.1. Markers

Finding the perfect marker can be difficult, but the Magic Leap Documentation gives some good advice in order to improve the built-in detection.
Magic Leap recommends using images printed on the size of US Letter or DIN A4 to get good tracking results from a 1m distance. For this application, a DIN A4 sized marker is too big, as it would cover a big part of the keyboard and make the correct position of the marker hard to get. Especially when using more than one marker, it covers even more space of the keyboard.
When sitting in front of the piano, the distances between the Magic Leap and the markers are around 50cm. Therefore the size of the markers can be a lot smaller than the recommended size of DIN A4.
Following Magic Leap, these characteristics should be met by the markers:

- **many high-contrast corners**
- **little to no symmetry**
- **little to no repeating patterns**
- **many identifiable differences between themselves and other images in the image set**

However, during testing, it was found that the suggested characteristics did not yield satisfying results. While the targets that are used in the image tracking example



(a) Left marker (b) Right marker

(c) Left marker (d) Right marker

Figure 4.1.: **Tested markers:** a) and b) show the targets that yielded the best results; c) and d) show the targets from the Image Tracking Example of the Lumin SDK

project of the Lumin SDK are detected faster, the position and orientation were not as accurate as c) and d) in figure 4.1. As the accuracy of the results is more important for the application than the speed of the detection, the suggested marker set has been discarded.

4.2.2. Calibration

Due to the linear shape of the keyboard, it can be calibrated using only two markers, which have to be placed on the left edge of the lowest key (A-0) and the right edge of the highest key (C-8) to mark the start- and endpoint of the keyboard. As discussed above, the tracking algorithm allows tracking of multiple markers at the same time. This means that two trackers have been implemented, each with a separate marker as its aim. However, in order to avoid false results and to minimize computation cost, the trackers are mutually exclusive.
After retrieving and accepting the position and orientation of the first marker in game coordinates through the special keyboard command *Accept*, the first tracker is disabled, and the second tracker is enabled. The position and orientation of the marker are stored in *ApplicationModel.leftMarkerPos* and *ApplicationModel.rotation*.

After retrieving and accepting the position of the second marker, it is also stored in *ApplicationModel.rightMarkerPos*. Note that the orientation of the second marker is not used, as it does not contain any further structural information due to the parallelism of the keys.

4.3. Keyboard visualisation

The keyboard's architecture is split into two separate classes. The class *Keyboard* represents the Keyboard as a whole, while the class *Key*, which inherits from *Keyboard* represents one single key. The *Keyboard* contains all necessary variables and methods to build up the virtual pendant, while the *Key* includes all relevant information for a single key.

4.3.1. Keyboard

In each scene, this class is attached to a Keyboard Unity *GameObject*. Upon initialization, it creates 88 *GameObjects*, each representing one of the keys, and attaches the *Key* class to it. Each of these keys is initilaized with a unique *noteNumber*, which can be used as an identification number. In order to match this *noteNumber* with the MIDI standard, the *Key* elements are numbered ascending, starting from *noteNumber = 21* for the lowest key, and ending at *noteNumber = 108* for the highest key of the keyboard.
The integer array *penta* holds indices of keys per octave that belong to a particular minor blues-scale as described in subsection 3.3.4. The key of this scale can be chosen in the improvisation mode through the function *selectKey*.The first five elements of *penta* represent the minor pentatonic, while the last element represents the respective blue note.
The *Vector3* variables *blackScale* and *whiteScale* store the relative dimensions of black and white keys. These dimensions are defined as described in subsection 3.1.

4.3.2. Key

Each *Key* element is identifiable by the *noteNumber*. The *Key* with *noteNumber = 0* (if it would exist) would represent a C-0. Therefore, much information can be obtained from *noteNumber*. The relative index of a key inside an octave can be computed through *noteNumber % 12*. A simple visualization of these relative indices can be found in figure 4.2.

Furthermore, the index of the octave is given through *(noteNumber-12)/12*. The *Key*

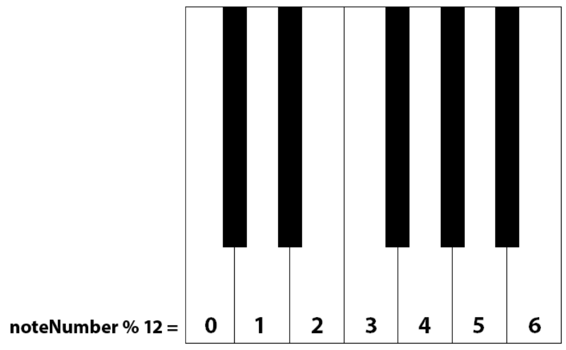


Figure 4.2.: Relative indices of keys in an octave

class includes a *pressed* and a *played* state, which indicates whether a key has been pressed by the user or whether it was played through a MIDI file. Each frame, a *Key* element's color is updated through *updateColor()*.
The position and orientation of a *Key* element are updated only if the keyboard has already been *calibrated* and if the *positionLock* is disabled. The function *isBlack* returns true if it is called on a black key and false otherwise.

updateColor()

Depending on the game mode, the keys can be displayed in different colors, e.g., to highlight certain features of a key like the belonging to a key or to give visual feedback when the key is pressed.
These color schemes are handled in the *updateColor()* method, which is called each frame through the *Update()* method. It first determines the pressed state of a key, as this state needs to be prioritized. The player should always get visual feedback when a key of the keyboard has been hit. However, this only applies if the keyboard has been calibrated. If the *Keyboard.calibrated* bool is set to false, the keys are hidden via the *Renderer* class of the Unity Engine.
The update method then checks which game mode is currently in use via the *isScene* string of the Application Model and then selects the color scheme that applies to this mode. The following paragraph shall summarize the used techniques in each mode.

Calibration As the Keyboard.calibrated bool is set to false during the whole calibration process, the keyboard is only visible as soon as the calibration process has been

finished. A pressed key is colored in green, while the rest of the keys are colored in black and white, just like the real piano, to check if the position of each virtual key matches the position of the real key.
We can make use of the repetitive structure of the keyboard and the noteNumber variable of the NoteIndicator class to determine whether a key is black or white. Every 12 keys, the structure repeats itself, so we can determine if the key is black, if the noteNumber mod 12 equals 1, 3, 6, 8, or 10.

Home, Free, Hero To make the application as natural as possible, these modes will not show any keys as long as the user does not play it. Just like in the Calibration mode, a pressed key is colored in green. As described earlier, the Hero mode implements an interface to receive a played state from an external source like a MIDI file.

Improvisation For the Improvisation mode, an integer array *bluesScale* of size six was created, to store the indices of the keys that belong to the blues scale. If a keys notenNumber module 12 is the same as one of the elements of "bluesScale", the key is colored pink. The first five elements of this array represent the pentatonic in the given key, while the last element represents the blue note, which has been discussed in 3.3.4.

updatePosition()

The position of a key is calculated depending on the octave of the key, the index of the key inside this octave, as well as the scaling vector *scaleVec* of the *Keyboard*. *octavePos(int octave* takes the index of an octave as its argument and returns the position of the start point of this octave. The keyboard is spanned along *scaleVec*, starting at *ApplicationModel.leftMarkerPos* and ending at *ApplicationModel.rightMarkerPos*. The position of each *Key*, relative to the *scaleVec* is shown in figure 4.3. As the first octave of the keyboard does not begin with a C, but with an A, the start and endpoint of the vector along the keys are spanned, need to be shifted by *-5 * scaleVec*, to compensate the offset of five white keys. The orientation of the keys is given by *ApplicationModel.rotation*. At the end of the positioning, black keys are elevated by 0.01, which corresponds to one centimeter. Furthermore, the size of the key is adapted by multiplying its relative scale with the scale, computed in the *Keyboard* class.
After updating the position of a key, its *positionLock* is set to true to prevent it from updating its position each frame.

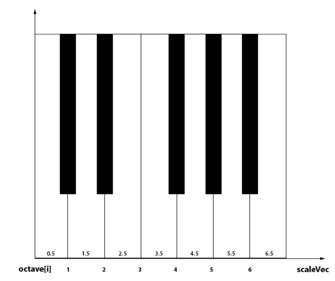


Figure 4.3.: Relative position of keys in each octave

setPlayedWithTime()

The function *setPlayedWithTime()* accepts a *float* and a *Note* as it's parameters. It checks if the *Note* already has been added to a *playedNotes* List and adds it to the list, if not. Once it has been added to *playedNotes* a *GhostNote* is created, and its position is set 0.2 distance units above the *Key* element. Its height depends on the first parameter, which indicates the duration of the note.

4.3.3. Ghost Notes

The *GhostNote* inherits from *Key* and represents the note indicator, that is sinking towards the key the user has to press during the "Hero" game mode. Once it is initialized in *Key.setPlayedWithTime()*, it descends with constant speed until it reaches its *killHeight* and is destroyed.
The position of the cube is updated each frame and slowly moves down towards the key from which it originated. As soon as the position of the note is lower than the *killHeight*, the object is destroyed.

4.4. MIDI

MIDI stands for Musical Instrument Digital Interface and describes a communication protocol, electrical connectors, and a digital interface. It is the protocol that has been used to establish the interaction between the Magic Leap and the keyboard of the piano. In the following subsection, a short introduction about the protocol shall be given.

4.4.1. MIDI Protocol

A MIDI message consists of a status byte and up to two parameter bytes.

First Byte The first half of the status byte holds information about the type of the message, while the second half determines one of 16 different channels. For this application, we only need the "Note On" as well as the "Note Off" type, which are encoded as 0x8 and 0x9. Most keyboards send MIDI messages on one channel only, so the second half of this byte can be neglected.

Second Byte This byte describes the note number, which is translated into the pitch in semitone steps, with C-4 being note number 60. As we are using keyboards with 88 keys, the relevant note numbers are between 21 for the lowest key and 108 for the highest key.

Third Byte This byte stores information about the strength with which the player has hit the key. It's translated into the volume of this note. However, this information is not used in this implementation and can, therefore, be neglected.

4.4.2. MIDI Device Input

In this application, the MIDI Input plug-in for Unity by Keijiro Takahashi has been used and adapted. The plug-in allows the developer to read incoming MIDI messages in the *MidiInput* class. The velocity of a Note On / Note Of message is represented as a float value between 0.0 for a velocity of 0 and 1.0 for a velocity of 127. Any value bigger than 0.0 is considered a pressed key, and *Key.pressed* is set to *true*.

In the same way, the pressed state of a combination of multiple keys can be inferred. This is used to implement the special keyboard commands, described in chapter 3. It also offers twelve methods to detect which kind of key has been detected (e.g., *c()* returns true if any C has been pressed), which is also used in the Improvisation mode in order to apply the color scheme of a new key to the keyboard.

4.4.3. MIDI File Input

The MIDI file reader implementation uses an external .NET library called *DryWetMidi*. It provides a full framework of reading and editing MIDI files, as well as a set of data structures, to do so. For this application, only the reading capabilities as well as some of the data structures have been used.

Once a MIDI file is read, all notes of this file are stored in an enumerator *notes* of Type *Note*, which is a class provided by the *DryWetMidi* library. Each *Note* contains information about its pitch — stored the variable *noteNumber* — , its duration — stored in the variable *length* — , as well as its time of occurrence — stored in the variable *time*. At initialization, also a timer starts, which tracks the time that has passed since the file was read. Each frame, the script moves through *notes* and checks whether *timer* lies between *Note.time* and *Note.time + Note.length*. If it does, it calls the function *Key.setPlayedWithTime()* for the affected *Key* element.

5. Evaluation

In order to evaluate the capabilities of this AR approach of a piano teaching application, interviews with several participants were conducted. The focus of the interviews was on the subjective perception of the teaching capabilities of this approach.

5.1. Methodology

5.1.1. Structure of study

The participants tested the application in a pre-calibrated state. As the aim was to evaluate the teaching capabilities and not the calibration procedure, the keyboard has been calibrated. The application was tested on the Magic Leap through the "Zero Iteration" tool and Unity. It has not been deployed to the device for this study.

5.1.2. Study procedure

Each participant started in the "Home" screen and has been given exact instructions on what they have to do. After a short introduction about the controls of the keyboard, the participants should enter the "Improvisation" mode.
The task in this mode was to choose a key of choice and to use the highlighted keys to improvise to an external backing track. After finishing this task, the participants should go back to the "Home" screen and enter the "Hero" mode. Once the "Hero" mode was started, they were instructed to play the descending virtual notes as they hit the real keyboard.
After finishing these tasks, the participants were interviewed.

5.1.3. Collected data

The interview was conducted freely and has used five critical issues as guidelines. The key issues and their intentions are listed below:

Comparison between scores and "Hero" mode

This issue aimed to get feedback for the visualization of the notes as proposed in this application compared with the classical notation of notes, and any advantages or disadvantages of the "Hero" approach should be pointed out.

Intuitivity of keyboard as game controller

The realization of the keyboard as a game controller should be evaluated with this issue. Special attention has been paid to the subjective perception of the intuitiveness of the controls. Furthermore other control concepts like speech recognition or hand gestures have been compared with the proposed approach.

Subjective feedback regarding "Improvisation" mode

As the primary purpose of the "Improvisation" mode is based on a variety of concepts of music theory, this issue should collect the subjective feedback of the participants. The degree of entertainment has been recorded on a five tiered-scale.

Further feedback

Any feedback from the participants that doesn't belong to one of the above categories should also be recorded. Improvements regarding gamification are expected to appear here.

5.2. Limitations of the study

Demography Due to the university environment, this study has been conducted in, all participants are in the age between 20 and 26. Furthermore, all participants have an academic background. This does not reflect the actual demography of piano students. However, we are focussing mainly on the degree of skills of the participants. As the application aims at beginners, the participants should have little to no prior knowledge of how to play the piano. This characteristic is met for 75% of the participants.

5.3. Results and interpretation

The results of the interviews are subdivided into the four subjects, mentioned in subsection 5.1.3

Comparison between scores and "Hero" mode

All participants were able to play the song "Au Clair de la lune" (see 3.6) without errors and without prior knowledge of which song they will have to play. This can be considered a success of one of the main features of the application, although the piece is easy to play for an advanced pianist, as naturally, a beginner starts to learn a new instrument with a simple piece.
When asking each of the participants which representation they would prefer in order to learn a piece of music in a limited amount of time, they all chose the augmented reality approach. However, 75% of them added that they would prefer the classical score notation if they would know how to read it, as it seems to offer more in-depth information about the music.

Intuitivity of keyboard as game controller

The entirety of the interviewees has accepted the keyboard as a tool to navigate through the app. The idea of using another device to control the application was found unfavorable.
However, the controls themselves were assessed as un-intuitive. This can be derived from missing or unclear instructions on how to use the keyboard for the controls.
The opinion on other interaction techniques like using voice commands or hand gestures differed widely among the respondents. While one half of them did not like the idea of using hand gestures to control the app, the other half claimed that it could significantly improve the intuitiveness of the navigation.

Subjective feedback regarding "Improvisation" mode

It could be observed that all participants were able to play along the backing track. It was even noticeable that they tried to follow the mood backing track with their improvisation.
All contributors rated the fun factor of the mode positively. As there has not been an explanation about the music theory behind the "Improvisation" mode, the respondents were *"surprised"*, that one can make music without understanding it.

We also consider this as a success of the application, as *"fueling"* the passion for making music was one of the main aims.

Further feedback During the evaluation process of the "Hero" mode, it could be observed that the participants not only tried to hit the correct key, but they also instinctively tried to press the key at the same position where the descending note would hit it. Although this resulted in the participants pressing the key at the wrong position, this can be seen positively, as it allows to quickly teach the user to hit the key at the correct position, by letting the note fall down to this position.
In general, the users have asked for more feedback throughout the app. Be it help menus that explain how to control or the app, or feedback when hitting wrong notes in the "Hero" and "Improvisation" modes.

6. Future Work

6.1. Calibration Improvement

In this work, a simple calibration algorithm has been implemented. During the course of testing and evaluation, it showed that the proposed calibration process was not easy to understand for the user, nor was the accuracy of its results satisfying. There are multiple possibilities for improvement in terms of stability, accuracy, and usability. A few of these possible approaches are described in this section.
The calibration not only should bring good results in theory, but it also has to be clear and understandable for the user. Therefore, the improvements can be split into two main areas, which are improvements in the algorithm and improvements of the intuitiveness.

Improvements of the algorithm

The implemented algorithm only uses two targets from which it extrapolates the position and the orientation of the whole keyboard. This method is prone to errors since it will give terrible results if only one of these targets has not been detected correctly. A way more stable approach would be to use at least two more targets and to use least-squares fitting, to find the line, along which the keyboard is positioned.
The least-squares algorithm takes a set of points and returns a line, of which the sum of the minimal distances to the points is minimal.
However, the more targets are used, the longer it takes the user to get their position, which will negatively affect the usability. A compromise between the number of targets and the usability would have to be found.

Improvements of the Intuitivity

As discussed in section 5, the marker detection was not easy to understand and unintuitive. One of the main reasons was the missing visual feedback if a marker has been detected. This can easily be solved by displaying the tracker's status in the canvas.

However, the user also needs to be able to match the virtual and real position of the marker visually. The magic leap does not allow us to display any virtual objects that are closer than 0.37m to avoid the possibility of eye strain. However, to retrieve the position of the marker, it is sometimes necessary to move closer to the marker. Even if the marker has been detected, the Magic Leap would not be able to display the detected position until the user has increased the distance a little.
This is most likely caused by the size of the markers. A short test of the image tracking algorithm using markers with a bigger size resulted in faster and more stable detection, even with the marker being placed one meter away from the Magic Leap.
Nonetheless, the bigger the markers, the more of the keyboard they cover, which would make it less natural for the user. A compromise between the marker size and the limitation of the usability would have to be found here as well.
Another conceivable approach would be to implement an object recognition algorithm that detects the whole keyboard, which is expected to give very accurate results. However, there is currently no such implementation in the Lumin SDK, and as there is no way to access the raw image data of the Magic Leap, such an algorithm cannot be used.

6.2. Gamification

Most people that want to play the piano starting when they are young and learning the piano is a task that needs a lot of effort of the student to be put into exercising and playing the instrument. This can be very tiring, and especially children do not have a long attention span. Adding principles of gamification is meant to make the learning process more comfortable. A few possible applications of gamification for this application shall be given in this section.

Point system

An often-used principle of gamification is adding a point system to the application, which rewards the user with points for the right actions and penalizes for wrong actions. Especially in terms of teaching an instrument, it is helpful to reward the user if he plays the right notes or to penalize wrong notes [SM83]. This principle could be implemented in the "Hero" mode, to give the user feedback for his attempt to play the selected song. Once a point system is established, a high score can be used to encourage the user further to reach a certain amount of points or to beat other students.

Difficulty levels

With the above point system, the current degree of skill of the user can be measured. It would be possible to give the user tasks whose difficulty is adapted to this degree of skill. The difficulty can be subdivided into levels, which also would encourage the user to practice in order to reach a certain level.

7. Conclusion

In this thesis, the educational opportunities of head-mounted augmented reality displays have been examined. A piano teaching application for the Magic Leap has been developed that should help the student to learn to play the piano faster. The application utilizes a simple calibration algorithm, which uses two marker points to set up a virtual environment that enables the application to highlight individual keys and indicate a particular musical characteristic.
An alternative approach of presenting sheet music has been implemented, which gives the user the information only where and when he needs it, which makes it an intuitive representation. We implemented an improvisation mode that uses theoretical music concepts to highlight specific keys which the student should use to improvise over a piece of music in a manually adjustable key.

The application was evaluated through interviews with several participants in terms of intuitiveness and usability. The respondents were all of the same opinions that the improvisation mode has sparked their interest in learning more about music theory. It shows that with advanced technology not only practical can be thought but it can also motivate the student.
Representing the sheet music as cuboids that are falling towards the keyboard has been assessed intuitive, as the test persons were able to play a song without errors and without any further knowledge.

With gamification concepts like adding a point system and high scores or technical improvements through a more robust calibration algorithm, this thesis gives an overview of what can be done in the future to to improve the usability of such an app further. It was also found that although technology has come very far in augmented reality, there are still limitations like a too-small field of view, which restricts the natural feeling when using an AR HMD. All in all, we believe that even at this stand of technology, such an app can make learning the piano much easier.

List of Figures

Bibliography

[Boe09] D. Boer. "Music makes the people come together: Social functions of music listening for young people across cultures." In: (2009).

[Cho+13] J. Chow, H. Feng, R. Amor, and B. C. Wünsche. "Music education using augmented reality with a head mounted display." In: *Proceedings of the Fourteenth Australasian User Interface Conference-Volume 139*. 2013, pp. 73–79.

[Dan+93] R. B. Dannenberg, M. Sanchez, A. Joseph, R. Joseph, R. Saul, and P. Capell. "Results from the piano tutor project." In: *Proceedings of the Fourth Biennial Arts and Technology Symposium*. 1993, pp. 143–150.

[HA17] D. Hackl and C. Anthes. "HoloKeys-An Augmented Reality Application for Learning the Piano." In: *Forum Media Technology*. 2017, pp. 140–144.

[Hil88] D. Hildebrandt. *Pianoforte: a social history of the piano*. Vintage, 1988.

[Hua+11] F. Huang, Y. Zhou, Y. Yu, Z. Wang, and S. Du. "Piano ar: A markerless augmented reality based piano teaching system." In: *2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics*. Vol. 2. IEEE. 2011, pp. 47–52.

[SM83] C. Szynal-Brown and R. R. Morgan. "The effects of reward on tutor's behaviors in a cross-age tutoring context." In: *Journal of experimental child psychology* 36.2 (1983), pp. 196–208.

[Tay88] J. A. Taylor. "Computers in music and music instruction: The joys of hardware and the woes of software." In: *Design for Arts in Education* 89.5 (1988), pp. 50–55.

[Wei+13] M. Weing, A. Röhlig, K. Rogers, J. Gugenheimer, F. Schaub, B. Könings, E. Rukzio, and M. Weber. "PIANO: enhancing instrument learning via interactive projected augmentation." In: *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. 2013, pp. 75–78.

A. Declaration of consent for interviews

Einverständniserklärung

Ich	Erdelt, Timo (Name, Vorname)
Geburtsdatum	20.05.1993

erkläre, dass ich mich freiwillig dazu bereit erklärt habe, im Rahmen der Bachelorarbeit

“Inspired by the ultimate perspective -Development of a piano learning application using augmented reality”

„Inspiriert von der ultimativen Perspektive -Entwicklung einer Klavier-Lernanwendungunter Verwendung von Augmented Reality“

Von Lukas Schneidt an der Nutzerstudie

„Interview: Evaluation of an AR piano learning application”

„Interview: Auswertung einer AR-Klavier-Lernanwendung“

teilzunehmen.

✓ Ich wurde für mich ausreichend mündlich und/oder schriftlich über die wissenschaftliche Untersuchung informiert.

✓ Ich erkläre mich bereit, dass im Rahmen der Studie Daten über mich gesammelt und anonymisiert aufgezeichnet werden. Bei der Veröffentlichung in der Abschlussarbeit wird aus den Daten nicht hervorgehen, wer an dieser Untersuchung teilgenommen hat.

Die aufgezeichneten Daten beinhalten:

- Antworten des Fragebogens, Beobachtungen der Versuchsleitung

✓ Mit der vorstehend geschilderten Vorgehensweise bin ich einverstanden und bestätige dies mit meiner Unterschrift.

München, den 08.03.2020	[illegible]
(Ort, Datum)	(Unterschrift der Testperson)

Lehrstuhl für Informatikanwendungen in der Medizin & Augmented Reality
Fakultät für Informatik
Technische Universität München

Einverständniserklärung

Ich Ottmann, Isabel
(Name, Vorname)

Geburtsdatum 26.09.1995

erkläre, dass ich mich freiwillig dazu bereit erklärt habe, im Rahmen der Bachelorarbeit

"Inspired by the ultimate perspective -Development of a piano learning application using augmented reality"

„Inspiriert von der ultimativen Perspektive -Entwicklung einer Klavier-Lernanwendungunter Verwendung von Augmented Reality"

Von Lukas Schneidt an der Nutzerstudie

„Interview: Evaluation of an AR piano learning application"

„Interview: Auswertung einer AR-Klavier-Lernanwendung"

teilzunehmen.

✓ Ich wurde für mich ausreichend mündlich und/oder schriftlich über die wissenschaftliche Untersuchung informiert.

✓ Ich erkläre mich bereit, dass im Rahmen der Studie Daten über mich gesammelt und anonymisiert aufgezeichnet werden. Bei der Veröffentlichung in der Abschlussarbeit wird aus den Daten nicht hervorgehen, wer an dieser Untersuchung teilgenommen hat.

Die aufgezeichneten Daten beinhalten:

- Antworten des Fragebogens, Beobachtungen der Versuchsleitung

✓ Mit der vorstehend geschilderten Vorgehensweise bin ich einverstanden und bestätige dies mit meiner Unterschrift.

Garching, 3.3.2020
(Ort, Datum)

[illegible]
(Unterschrift der Testperson)

Lehrstuhl für Informatikanwendungen in der Medizin & Augmented Reality
Fakultät für Informatik
Technische Universität München

Einverständniserklärung

Ich Irro Janine
(Name, Vorname)

Geburtsdatum 11.10.1999

erkläre, dass ich mich freiwillig dazu bereit erklärt habe, im Rahmen der Bachelorarbeit

"Inspired by the ultimate perspective -Development of a piano learning application using augmented reality"

„Inspiriert von der ultimativen Perspektive -Entwicklung einer Klavier-Lernanwendungunter Verwendung von Augmented Reality"

Von Lukas Schneidt an der Nutzerstudie

„Interview: Evaluation of an AR piano learning application"

„Interview: Auswertung einer AR-Klavier-Lernanwendung"

teilzunehmen.

✓ Ich wurde für mich ausreichend mündlich und/oder schriftlich über die wissenschaftliche Untersuchung informiert.

✓ Ich erkläre mich bereit, dass im Rahmen der Studie Daten über mich gesammelt und anonymisiert aufgezeichnet werden. Bei der Veröffentlichung in der Abschlussarbeit wird aus den Daten nicht hervorgehen, wer an dieser Untersuchung teilgenommen hat.

Die aufgezeichneten Daten beinhalten:

- Antworten des Fragebogens, Beobachtungen der Versuchsleitung

✓ Mit der vorstehend geschilderten Vorgehensweise bin ich einverstanden und bestätige dies mit meiner Unterschrift.

München, 03.03.2020
(Ort, Datum)

[illegible]
(Unterschrift der Testperson)

Lehrstuhl für Informatikanwendungen in der Medizin & Augmented Reality
Fakultät für Informatik
Technische Universität München

Einverständniserklärung

Ich ZENGERLE, MARISA
(Name, Vorname)

Geburtsdatum 28.04.94

erkläre, dass ich mich freiwillig dazu bereit erklärt habe, im Rahmen der Bachelorarbeit

"Inspired by the ultimate perspective -Development of a piano learning application using augmented reality"

„Inspiriert von der ultimativen Perspektive -Entwicklung einer Klavier-Lernanwendungunter Verwendung von Augmented Reality"

Von Lukas Schneidt an der Nutzerstudie

„Interview: Evaluation of an AR piano learning application"

„Interview: Auswertung einer AR-Klavier-Lernanwendung"

teilzunehmen.

✓ Ich wurde für mich ausreichend mündlich und/oder schriftlich über die wissenschaftliche Untersuchung informiert.

✓ Ich erkläre mich bereit, dass im Rahmen der Studie Daten über mich gesammelt und anonymisiert aufgezeichnet werden. Bei der Veröffentlichung in der Abschlussarbeit wird aus den Daten nicht hervorgehen, wer an dieser Untersuchung teilgenommen hat.

Die aufgezeichneten Daten beinhalten:

- Antworten des Fragebogens, Beobachtungen der Versuchsleitung

✓ Mit der vorstehend geschilderten Vorgehensweise bin ich einverstanden und bestätige dies mit meiner Unterschrift.

Garching, 3.3.20
(Ort, Datum)

[illegible]
(Unterschrift der Testperson)