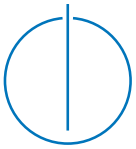
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Development of an Augmented Reality Based eLearning Tool for Teaching the Basics of Harmonic Theory

Leon Brooks



DEPARTMENT OF INFORMATICS

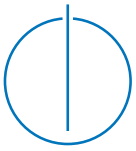
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Development of an Augmented Reality Based eLearning Tool for Teaching the Basics of Harmonic Theory

Entwicklung eines Augmented Reality basierten eLearning Tools zur Vermittlung der Grundlagen der Harmonielehre

Author:	Leon Brooks
Supervisor:	Prof. Dr. Gudrun Klinker
Advisor:	M.Sc. Sven Liedtke
Submission Date:	15.07.2022



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

Munich, 15.07.2022 Leon Brooks

Abstract

Augmented Reality is a recent emerging technology which has impacted many fields including e-learning. Head-mounted display devices (HDM) can provide significant learning benefits within the music education subdomain. While many applications exist using HDM's to teach beginners an instrument, next to no research has been done on their use for music/harmonic theory learning. Therefore, this thesis aims to present the complete implementation process for such a prototype. A literature review is conducted, the findings are used to design and implement the application, a small (8 person, 7 response) qualitative user study is carried out and finally key challenges and guidelines for future work are presented.

Contents

1 Introduction

Augmented Reality (AR), the technology which allows superimposition or composition of virtual object with the real world [2], has become more and more popular in recent years. As AR is a fairly general concept the different use cases and opportunities it provides are still being explored. One of the areas where AR can benefit users is in education and e-learning[3]. When looking at the subdomain of music education this trend has inspired some developers to integrate AR head-mounted display devices (HMD) into their applications. In comparison to hand held AR devices, like smartphones overlaying information onto pictures/video recorded through their cameras, HMD's allow for a more intuitive experience and for users to keep their hands free. This of course has lead to the development of many applications which help beginners to learn an instrument. Visual instructions are overlaid directly onto the instrument leading to experiences reminiscent of the video game Guitar Hero[1] [6, 10, 4, 19].

While these applications focus on teaching an instrument in a very concrete way, there are only very few applications aiming to teach music theory concepts as whole. Playing an instrument does not necessarily teach the underlying concepts of how music is structured and what systems are behind the interaction of different individual tones. However, this knowledge can broaden a students horizon allowing them to think about music more creatively and make them better musicians overall [5, Section 6.16]. While lowering the barrier of entry, especially the decision to completely replace sheet music by AR overlaid instructions, might be a detriment in the long run.

The goal of this thesis is to walk the reader through the full development process of the AR harmonic theory e-learning prototype application Music Vision. The application aims to teach users the basics of harmonic theory, a subdomain of music theory which focuses on the interaction between different notes and tones. In doing so it should help the user better understand the connection between sheet music and keys on the keyboard as well. The development will be targeting the HMD-device Hololens 2[2] and the implementation will utilize the Unity game engine[3] as well as the Microsoft Mixed Reality Toolkit[4] (MRTK). The complete source code will be available on Github[5].

A brief review of related applications and literature will be conducted in chapter 2. Following this, chapters 3 and 4 will explain how the gathered information was used

[1] A game where the player has to press buttons on a custom guitar like controller to "replicate" a song.
[2] `https://www.microsoft.com/en-us/hololens`
[3] `https://unity.com/`
[4] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2`
[5] `https://github.com/LeonBrooks/AR-Music-Theory-eLearing`

to design and implement the prototype. In chapter 5 the results of an eight person user study evaluating the application will be presented. Chapter 6 will analyze problems encountered during the development and formulate helpful guidelines for future work in this area. Finally, chapters 7 and 8 will summarize the previous chapters and show conceptual possibilities of improvement.

2 Related Work

When looking at the body work surrounding the topic of AR based teaching of harmonic theory, it is important to note that it is fairly unexplored. While there are many articles on AR tutoring applications for playing an instrument (e.g. [6, 10, 4, 19]), only very few exist focusing on the specific subdomain of music or harmonic theory. Additionally some of the publications presented in this chapter, especially those related to other applications, are very recent and barely cited or reviewed. They must therefore be approached with caution and should bee seen rather as ideas for design directions rather than proven concepts. Considering this, a broader approach was chosen when researching the remaining literature. By combining more general research from music theory teaching and AR pedagogy fields, this review aims to build a foundation of knowledge on which the development of the prototype application can rely on. At the same time it is explored which benefits AR may offer over over traditional applications and explained how these relate to known pedagogy concepts.

2.1 Related Applications

In their 2020 publication [1] Avanzini et al. present their AR project aiming to introduce school children without prior knowledge to harmonic theory through the means of embodiment[1]. In a three step development process they show different approaches to harmonic theory applications. The combination of a tracking based AR experience (Harmonic Walk) combined with a browser based game (Harmonic Touch), culminates into an iOS mobile application suitable for personal as well as school use (AREmbody). While lacking any type of evaluation, this project gives an interesting insight into how the AR context can affect application design. It shows that it might be useful to visualize harmonic theory principles as physical objects to enable embodiment within the user.

Continuing the idea of physical representations for harmonic theory concepts the project ChordAR [16], presented in 2022 by Lu et al., used Lego bricks to represent musical chords. Aimed at young children between the ages of 4 – 7, it used narrative driven tutorials together with brick based exercises to teach basic notes and chords. A mobile device's camera mounted on a custom made miniature workstation (Figure 2.1) was used to recognize the brick assemblies and map them to notes and chords within the application. As with AREmbody, the creators of ChordAR also mention embodiment

[1] In cognitive science: The idea that the body plays a significant role in cognition [22]

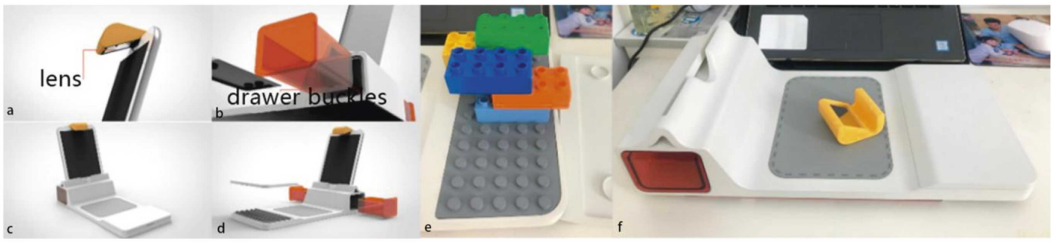


Figure 2.1: ChordAR custom workstation [16]

as one of the driving factors for their design [1, p. 3]. Contrary to AREmbody, a small user study with 12 participants was conducted. While there were some minor troubles related to the difficulty of the game, overall the children appeared to react positively to the application. Another important takeaway mentioned was that from observation it appeared as if the color of the bricks improved the participants memorization of different chords.

The third application found, which sought to combine AR and music theory, was presented in 2017 by Ho et al. [11]. While using a more traditional setup of a combination of tutorials and exercises it incorporated AR through marker based cards used for navigation and answer input. As apparent from the provided description of the application, the usage of AR was rather minimal. It was limited to a replacement of simple buttons within the app by the marker cards outside the application (Figure 2.2). Although these AR interactions are not particularly helpful for the construction of

Figure 2.2: A screenshot of the exercise mode and the markers used by Ho et al. [11]

a more modern application, it is noteworthy that in a 34 user study these markers were generally rated around 4 out of 5 on a Likert scale on questions regarding the

enjoyment and interest in this interaction type[2]. While not the focus of the paper, another interesting observation potentially more relevant for other AR applications occurred within the results of the user study. As the application used voice overs during the tutorial mode, the participants were surveyed on how much they enjoyed this. This form of instruction scored an average 4.03 out of 5.

2.2 Value of AR in an E-Learning Context

As only three applications could be found which directly combine AR with music/harmonic theory and none of them featured a state of the art approach using HMDs, this wasn't sufficient to derive strong design guidelines and concepts for developing the prototype for this work. Therefore other more general literature was reviewed. This section will be focusing on literature featuring different ways of building AR based e-learning applications and proof of their effectiveness.

2.2.1 General Advantages of AR

When comparing different meta reviews analyzing the advantages of AR over other (especially paper based) methods, some common points can be found:

- *Content Understanding:* In his 2014 review of 26 scientific publications [20], Radu notes that a large number found AR more effective for teaching students than comparable methods such as books, videos, or PC experiences. A similar result is found by Santos et al. [21]. They analyzed a total of 87 research articles, seven of which allowed the calculation of an effect size[3]. The mean of these effect sizes was 0.56 which is higher than the mean for the effect size of technology in general (0.35). Some of the effect sizes of the seven AR experiences reviewed were negative however, which suggest that the concrete implementation and usage of other didactic/pedagogy methods could be a deciding factor.

- *Knowledge Retention:* Four related papers reviewed by Radu, indicated a improved long-term memory retention of AR compared to non-AR methods. Santos et al. only hypothesized that certain implementations of AR should improve memory retention based on know cognition theories. In a brief literature review prefacing his project of teaching students a sound setup using AR annotation [7], Cook cites the publications mentioned by Radu in addition to another study, to establish the hypothesis that AR improves memory retention. He then confirms this hypothesis

[2]The two exact questions were:
1: "I enjoy learning with real picture cards." (mean score 4.0)
2: "I think it is interesting to answer assessing questions with picture cards." (mean score 4.35)

[3]"a quantitative reflection of the magnitude of some phenomenon that is used for the purpose of addressing a question of interest" [15]

in his own user study with a small number of students. Similarly Keebler et al. [14] conducted an experiment where two groups were taught a scale on the guitar and had to recite it two weeks later. One of the groups was taught using an AR device which displayed the correct grips directly onto the guitars fretboard while the other group was taught using a traditional diagram. In the recital two weeks later, the group taught using the AR method performed significantly better, suggesting a better long-term retention rate than the traditional method. Although not fully conclusive, overall it appears as if correct usage of AR can increase knowledge retention.

- *Motivation/Enjoyment:* On this topic Radu mentions three of the 26 reviewed papers explicitly stating the AR systems helped improve motivation with users, in some cases even despite them running into usability issues. Additionally both of the applications reviewed in section 2.1 that conducted user studies showed similar results. When evaluating their AR piano tutoring application [6], Chow et al. noted that all of their seven participants enjoyed using the application. In 2007 Kaufmann and Dünser published a collection of three evaluations of different versions of their AR modeling software Construct3D [13]. One of the studies asked user to rate their satisfaction with the application on a four point Likert scale. The mean satisfaction score was around 3.6 compared to about 2.8 for a similar traditional computer-aided design software. Although the other evaluations didn't score motivation/enjoyment, users of the first were described as showing signs of enjoyment by moving around, laying under and walking in and out of the models even after completing the actual task. Although some of the findings by all these different publications may be attributed to novelty, they do indicate that the usage of AR can produce high engagement and potentially further motivate learners.

2.2.2 Affordances of AR and Design Guidelines

In his 1988 book "The Psychology of Everyday Things"[4] [18], Donald Norman coins the term affordance in an interaction design context. He describes an affordance as a type of interaction a certain object offers to a certain user. For example Norman states that a chair affords sitting. In a human-computer interaction context, an affordance usually refers to interaction capabilities offered by the application to the user.

As AR affords interactions not possible by traditional applications, it is worth analyzing how this might affect design guidelines. In their literature review Santos et al. [21] define three affordances as inherent to AR by definition:

1. *Real world annotation:* The ability to annotate objects with overlay text/symbols.
2. *Context visualization:* The ability to display content in a real world context.

[4] later renamed to "The Design of Everyday Things"

3. *Vison-haptic visualization:* The ability to provide embodied interaction with virtual content

Although not labeled as affordances, Radu [20] describes similar factors that influence AR learning, most of which can be categorized under the affordances mentioned above.

Furthermore Santos et al. use known and proven learning theories to explain how and why these affordances, if correctly implemented, can produce learning benefits. Although not directly guidelines, the correct usage and implementation can be derived from this knowledge. To explain the advantages of real world annotation Santos et al. refer to multimedia learning theory [17]. From it they derive that people will learn better using combined media elements (object + annotation) which are presented close to each other [21, pp. 48 sq.]. For context visualization they use the contextual learning philosophy to justify the statement that more effective learning can be achieved by linking the information to a familiar object or environment using contextual visualization. Furthermore an AR adaptation of animate vision theory [23] is used as a basis to explain that more direct interactions (i.e. not mediated by an input device) like grabbing are superior for learning compared to the classical mouse and keyboard input. This is also in line with the concept of embodiment explained in section section 2.1.

Overall the following guidelines can be extracted from these affordances and explanations:

1. Important information should be displayed through annotations closely placed by the objects they relate to.
2. Information should be presented in a related real world context or a context familiar to the user.
3. Direct physical interaction with the main objects of interest should be included.

While not explicitly defining the unique affordances of AR, Matt Dunleavy [8] presents three design guidelines developed to take advantage of them:

1. *Enable and then challenge:* This concept was designed to deal with cognitive overload. Dunleavy found this to be a much reported problem within AR development [8, p. 29]. Therefore it is proposed that e-learning applications should *first* enable the understanding of the basics of a concept and only *then* challenge the user to solve higher level problems. Besides other concrete actions, it is proposed that replacing text with audio might help to reduce cognitive overload as well. This is also in line with the high scores users gave for the voice over in the previously motioned marker based music theory application (section 2.1).
2. *Drive by gamified story:* Dunleavy recommends driving the learning experience though a gamified story. This could include point scoring or other progression

elements as well as location based item collection and other features leveraging AR affordances. Together with narrative storytelling elements this may "cross-fertilize entertainment and education".

3. *See the unseen:* As AR affords the possibility to visualize objects/concepts which would normally be invisible, it is recommended to include this type of interaction whenever possible. Typical example use cases include medical or biological science e-learning applications where the inner workings of the human body or an animal may be displayed in a visually intuitive way using AR.

2.3 Considerations from the Field of Music Pedagogy

In his 2015 article [5] Michael R. Callahan describes a new teaching approach he took with first and second-year undergraduate music theory students. To make the lessons more interactive for students he switched from written homework to exercises preformed and recorded on a digital keyboard which were then submitted for review. A user study with 37 of the second-year students was conducted to evaluate this new approach. Although a lot of the higher level music theory concepts and exercises discussed in the article are beyond the scope of this thesis, there are still some takeaways which might be useful in the development of a less complex application:

- One of the question types used by Callahan were fill-in-the-blanks type questions. Although he uses them on more advanced concepts like simple composition, they could be adapted to fit simpler topic like scales or chords.
- During the use study some participants who primarily played other instruments noted that a piano keyboard is more helpful when learning music theory, as it provides visual aid and can play multiple tones at once [5, Section 6.4]
- At many points it is emphasized that students who weren't used to doing practical exercises pointed out that hearing their results was very helpful for them. It allowed them to get another form of immediate feedback and even changed how they analyzed music later in the course [5, Sections 6.9, 6.10, 6.12].
- The concept used also had significant positive effects on the students motivation and overall attitude towards music theory. It also help them better relate music theory to other abilities such as performing music [5, Section 6.15].

3 Application Design

This chapter will be focusing on combining the key takeaways of the last chapter in order to sketch out a software design concept on which the application prototype can be built on. It will be discussed which design guidelines may or may not be useful and applicable for this application. From this, the desired interaction models and software requirements will be developed. Furthermore the different software components and their individual functionalities will be introduced. That said, it is important to remember that the overall goal is to create a prototype to evaluate different types of user interactions and present an exemplary software design for a harmonic theory AR learning application. The focus will therefore lay more on technical aspects. Whenever applicable, the optimal pedagogic concepts will be included. But when beyond the scope of this project, simple implementations may be preferred over the theoretically best ones.

3.1 Requirements Analysis

Up until this point the exact requirements the prototype should fulfil are only defined very broadly. As defined by the title, the overall goal should be an AR application that teaches the basics of harmonic theory. Besides that, in chapter 1 it was briefly discussed that sheet music will be included. When thinking about other visible components the application must include, some form of musical instrument to create sounds and visually represent the note being played comes to mind. This is especially important as hearing the sounds of notes can significantly improve learning abilities (see section 2.3). As suggest by [5, Section 6.4] the piano keyboard is a good choice for a concrete instrument. As a Keyboard and sheet music together are sufficient to teach all simple concepts of harmonic theory, those will be the only visual components of the application besides UI-elements. Although there surely are other visual components which could further the learning process as well, this trade off was chosen to have more time to focus on the implementation of learning/instruction methods.

As mentioned previously this application will be using a HMD as AR device. This is in part because no other works have explored this option for music theory education, but also because of other benefits such as increased immersion and motivation through novelty as well as other more technical benefits. As hardware device the Hololens 2[1]

[1] `https://www.microsoft.com/en-us/hololens`

together with the Unity game engine[2] and the Microsoft Mixed Reality Toolkit[3] (MRTK) as software framework was chosen. Besides displaying virtual holograms over the HMD, this setup offers a broad range of exploitable functionalities such as hand tracking, eye tracking, speech recognition and spatial awareness[4]. This can allow the creation of said highly immersive experiences while also providing the basis for integrating embodied user interaction.

3.1.1 Integrating AR-Affordances

As shown in section 2.2 the qualified inclusion of AR and implementation of its unique affordances can yield great results for the learning process. Therefore, it is essential to discuss if and how they could best fit within this application.

When looking at the guidelines extracted from Santos et al. [21] (see subsection 2.2.2) the following conclusions can be made:

1. *Real world annotation:* Annotation is a concept which can easily be included within the application. Names of notes and keys can be marked on the sheet music or the keyboard respectively. Additionally more complex concepts and key takeaways can be marked on the sheet music as well. Although not a traditional form of annotation, coloring on both keyboard and sheet music can be used to emphasize the connection between the representations of an individual concept appearing in both simultaneously. Especially since this had shown a positive effect on memorization in the ChordAR project [16] (see section 2.1).

2. *Context visualization:* As harmonic theory contains many abstract concepts with little direct physical representation, achieving this is a more challenging task. Besides the user being able to run the application wherever he pleases (i.e. a familiar context), the main forms of context visualization are the sheet music and the keyboard themselves. They provide a concrete context to the abstract concepts presented.

3. *Vison-haptic visualization:* An obvious opportunity to include direct physical interaction is the overlaying of coloring and other instructions over a real world keyboard. While this is another form of adding context visualization as well, the technical effort required to do so is beyond the scope of this project. Instead a virtual keyboard which the user can interact with over the hand tracking functionality is more appropriate. Although this offers no tactile feedback, it does represent a form of direct physical interaction as the user can press the keys and is offered visual as well as audio feedback.

[2] `https://unity.com/`

[3] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2`

[4] The device scans the surrounding area and provides this data to the programmer as a mesh (a model represented by connected vertices).

Additionally the opportunity of combining hand tracking and the sheet music offers a unique possibility of interacting with the notes draw within the sheet music. For example it could be possible to grab and drag a note within the sheet music to change its pitch. At the same time the change could be displayed on the keyboard.

Additionally the guidelines presented by Dunleavy [8] (see subsection 2.2.2) can be integrated as follows:

1. *Enable and then challenge:* As the Hololens 2 provides only a very limited field of view, preventing cognitive overload is especially important. Large amounts of text in particular might lead to overcrowding of the field of view and could be more of a detriment than a benefit. The recommendation made by both Dunleavy and Ho et al. [11] (see section 2.1), to replace text through audio, can be very helpful here. Although fully narrated instructions are beyond the scope of a simple prototype, the MRTK offer a fairly sophisticated text-to-speech (TTS) facility which can be used to fulfil the same role.

2. *Drive by gamified story:* Most of the literature, including that analyzed by Dunleavy, was particularly aimed at children. Partly because this prototype is not specifically targeting children, but mainly due to time constraints, this recommendation could not be included within the application. This is not to say that adults cannot benefit from a gamified story, however it was not seen as quite as high of a priority compared to other aspects of development.

3. *See the unseen:* Without going through the effort of developing a custom visual representation of harmonic theory concepts there is little to visualize which would otherwise be unseen. Instead, the focus was put on visualizing the connection between sheet music and keyboard. This is accomplished in part through connected coloring, but also by changes in notation (such as an accidental[5] being added) being reflected on the keyboard and vice versa.

3.2 Software Structure

After defining some of the wanted interaction types and functionalities, a closer look can be taken at more concrete software design aspects. When contemplating which components the application should be made of, the two previously mentioned ones, the keyboard and the sheet music come to mind first. Besides those the prototype must include some type of tutorials, to be qualified as an e-learning tool. Another important decision is which component should be responsible for the playing of the

[5] A sign that can be added to a note in sheet music. Depending on the type, it can increase or decrease the pitch of the note by one semitone.

piano sounds. At first it might seem best to let the keyboard handle this as it is the receiver of the key presses as well. However, a problem arises when considering that the tutorials might need to access the sound system too and it should be possible for changes in the sheet music to be represented on the keyboard and with sound as well. As all these components would have to access the keyboard with potentially conflicting interests, this would cause the keyboard component to be significantly overloaded compared to the other components. Therefore, it appears to be more appropriate to introduce a new component to handle the sound system. This component will be called the music controller. Overall the application will be setup in a way where the music controller tracks what keys and therefore which individual sounds are being played (see Figure 3.1).

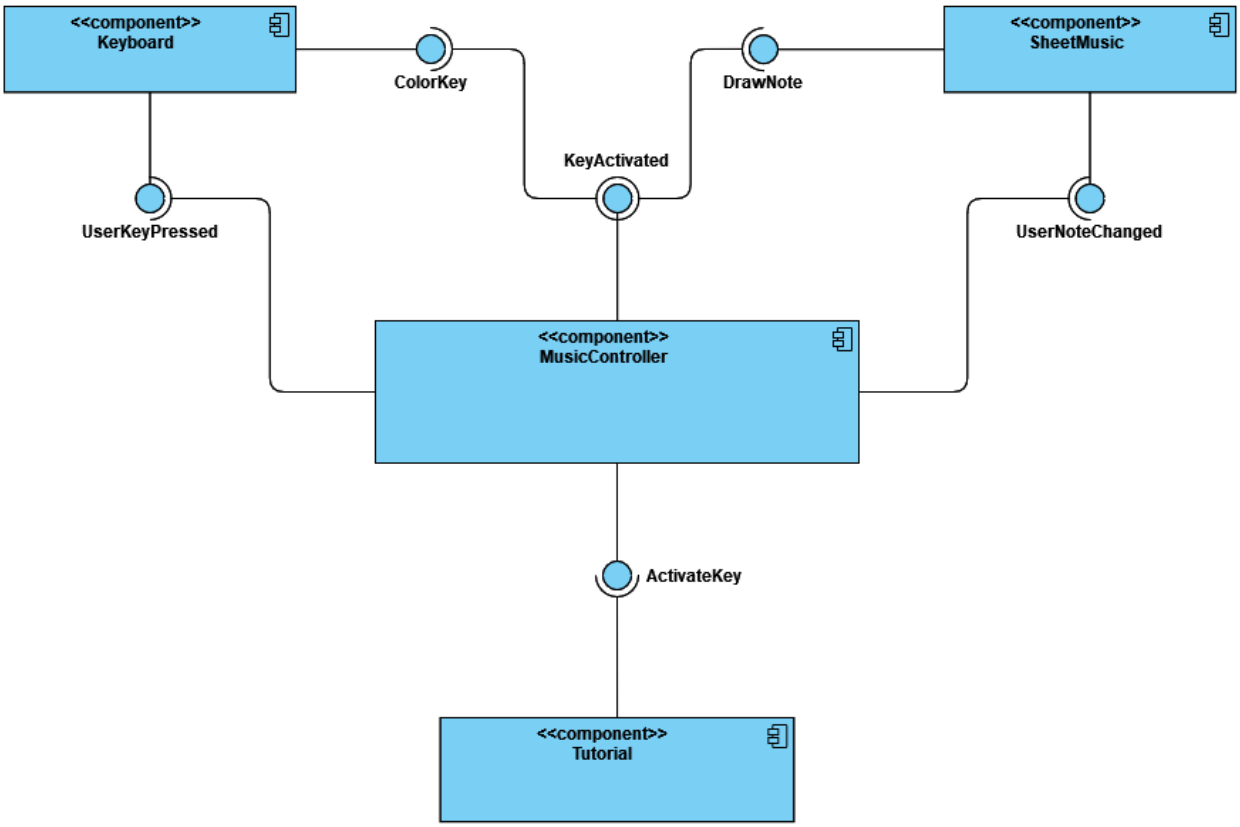


Figure 3.1: Component diagram showing the basic setup of the application.

3.2.1 Keyboard

When specifying the keyboard requirements, this first point that comes to mind is the key range. A full piano keyboard has 88 keys over seven octaves[6]. However, only notes out of the middle four octaves can be written into a piano music sheet without using special annotations. Since these four octaves are more than sufficient for teaching harmonic theory and the Hololens field of view is already limited, only said octaves will be used (c2-c5 octaves). Furthermore the keyboard must provide the following functionalities:

1. *Surface placement:* The user must be able to place the keyboard freely on any flat surface (like a table) within his vicinity.
2. *Keypress physics:* When the user presses the keys, they must respond in a way similar to real world physics.
3. *Coloring:* The keys must be able to change color.
4. *Key nameplates:* The white keys must be able to display their individual names.
5. *Key retrieval interface:* The keyboard must provide a way to access the individual keys (for proposes like color change, etc.).

3.2.2 Sheet Music

As the piano keyboard was chosen as input instrument, this choice must be reflect within the sheet music as well. Therefore, the standard piano notation of two staves[7], with the top staff written in the treble clef and the bottom staff written in the base clef, will be used. Additionally the sheet music must provide the following functionalities:

1. *Free placement:* The user must be able to freely place the sheet music anywhere in his vicinity.
2. *Drawing Notes:* The sheet music must provide an interface which allows the specification of notes and chords to be drawn within it.
3. *Coloring:* Similarly to the keyboard, notes must be able to change colors.
4. *Note annotation:* It must be possible to attach tooltips with custom text to one or multiple notes.
5. *Note dragging:* The user must be able to grab notes and drag them to change their position and add or remove accidentals.

[6] In this context: A repeating pattern of keys on the keyboard. As the name suggests it contains eight white keys, but it also contains five black keys as well.

[7] Plural of staff: A set of five lines in which note are drawn.

6. *Change propagation:* When notes are changed the sheet music must be able to map the note change to a specific key/tone and provided that data to the music controller.

3.2.3 Music Controller

The music controller manages the connection between the keyboard, the sheet music and the tutorials. It must track which notes are active, play the corresponding sounds and ensure that there is no conflict between the different other components. Furthermore, it controls whether user input is allowed and what colors are used by default.

3.2.4 Tutorials

The tutorials are the core feature of the application. They should present the basic concepts of harmonic theory to the user and provide simple exercises including different interaction methods. It is important to note that since this is a prototype the focus will not be on creating continuous exercises facilities, but rather developing exercises types that fit the AR and music theory context. Therefore the exercises will be hard-coded within the tutorials and not a system for random exercise generation. Considering that, the tutorials should fulfill the following requirements:

1. *No prerequisites:* The tutorials should assume no prior knowledge of music theory.
2. *Audio narration:* The tutorials should use TTS based narration instead of text.
3. *Include all interaction types:* The exercises should include key, note and speech input.

To ensure the first requirement is met the topics selected for the tutorials must start with the most basic concepts of harmonic theory. Overall five tutorials were created for this prototype. The first tutorial covers semitones and the basic keyboard setup. The second tutorial covers note names and how they are written within sheet music. The next tutorial explains the usage of accidentals and how that involves the black keys. The last two tutorials introduce scales and chords respectively and explain the difference between major and minor versions of each.

4 Implementation

In this chapter all the technical implementation details will be discussed and the chosen solutions presented. The different components introduced in the previous chapter will be modeled as concrete classes and scripts within the Unity game engine and the Mircrosoft Mixed Reality Toolkit (MRTK). As mentioned previously, the complete Unity project including all the source code can be found on Github[1]. All of the code presented will be in C# as this is the language Unity uses for its scripts. All mentioned scripts can be found within the script-, tutorial- or audio-directory within the Unity project.

To completely understand all explanations it is important to be familiar with the basic setup of Unity. Unity allows the developer to place basic "building blocks" known game objects[2] into a scene. Game objects are containers to which different types of components can be added. The most important component a developer can add are C# scripts. In order to be added to a game object, a script must derive from the `MonoBehaviour`[3] class. This allows it to access other components attached to a game object and receive callbacks through methods like `Update()`, which is run every frame. The developer can then overwrite these methods and add new ones to achieve the desired functionality. If a class should not be attached to a game object, and does not require the engine provided functionalities, the developer can choose not to inherit from `MonoBehaviour`. This allows for more traditional class setups which do not have an object representation within the scene. An example of this is the setup used for the tutorials (see section 4.4).

4.1 Keyboard

The `Keyboard` is represented as a game object combining four octaves which each contain twelve keys. Each key has a `KeyPressedHandler` script attached to it which manages the key physics and the user interaction. The `Keyboard` has a `KeyManager` script attached which acts as an interface for other classes. It provides methods such as `changeColor()`, `showNameplate()` and `resetKey()`. The private helper method `getKey(string key)` can retrieve any key when passed its string representation. A key's unique identifier string contains the keys signature letter and the number of the octave it is a part of. All black keys are represented as sharps, therefore a S is added

[1] `https://github.com/LeonBrooks/AR-Music-Theory-eLearing`
[2] `https://docs.unity3d.com/Manual/class-GameObject.html`
[3] `https://docs.unity3d.com/Manual/class-MonoBehaviour.html`

after the signature letter[4]. To better handle this notation, a global enum `Key` was created. It contains an entry for each of the 48 keys. It can easily be transformed into the string representation using the `toString()` method. Color changes are executed by changing the key's material to another one in the specified color. Nameplates are simple text objects overlaid onto the white keys with their signature letter (C-B).

The `KeyPressedHandler` script ensures a natural feeling movement of the keys and uses Unity's collision system to implement user interaction. As part of its hand tracking feature the MRTK provides a Unity physics compatible representation of the hands[5]. This allows Unity style collision detection[6] and therefore significantly simplifies the task at hand. Once a collision is detected the key is moved a certain distance downwards, determined by a factor which is based on how far the hand has entered the key. This ensures that the downward movement of the key is related to how fast the key is pressed. If the key is pressed past a certain point it is activated, relaying that information to the `MusicController`. When no collision is detected, the key will move upward at a fixed speed until it has reached its base position. At this point, it is important to note that the MRTK hand tracking precision is far from perfect. Overall the keyboard interaction can be described more as pressing individual keys, rather than being able to play a piece of music. Most of the time the user can press at most two keys with each hand, without hitting other keys by mistake. This low precision can partially be explained by the Hololens relying on head-mounted cameras for the tracking algorithm.

The final requirement specified for the keyboard in subsection 3.2.1 is the ability to place it on flat surfaces within the vicinity of the user. The method chosen to implement the placement process is an adapted version of the solution presented by Joost van Schaik in his blog DotNetByExample[7]. It uses the MRTK `SurfaceMagnetism`[8] solver together with the logic provided in the `SurfaceFinder` script and other required components attached to the `KeyboardPlacer` game object. The `SurfaceMagnetism` solver ensures that the keyboard will attach to any surface the user looks at. Once a satisfactory position is found, the user can press a button on the attached `Placement_Look_Promt` which deactivates the solver and locks the keyboard in place. He can then adjust the scale and rotation using the sliders of the `Placement_Confirmation_Promt` (see Figure 4.1).

[4] E.g.: The complete C2 octave would be named: C2, CS2, D2, DS2, E2, F2, FS2, G2, GS2, A2, AS2, B2

[5] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/extensions/hand-physics-service`

[6] https://docs.unity3d.com/Manual/CollidersOverview.html

[7] `https://localjoost.github.io/Placing-holograms-on-any-surface-using-the-MRTK-surface-magnetism/`

[8] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/solvers/solver#surfacemagnetism`

(a) Keyboard placement part 1: Look at surface.

(b) Keyboard placement part 2: Adjust.

(c) Pressing a key

(d) Keyboard Nameplates

Figure 4.1: Keyboard features

4.2 Sheet Music

The sheet music is divided into two scripts: the `SheetMusic` and the `Note`. The `SheetMusic` serves as the background and manages all the notes drawn within it. As real sheet music is typically a piece of paper, a Sprite[9] representation was chosen to reflect the 2D nature.

The `SheetMusic` script is attached to a game object which renders a Sprite of an empty sheet music. The script adjusts the Sprite to always be facing the user. It also offers a few public methods which expose important functionalities to other classes. Most notably the `drawNote()` and `drawChord()` allow other classes to draw into the sheet music. Notes are specified by passing a enum entry of the base key (C-B) and an integer to define whether the note is flat (-1) neutral (0) or sharp (1). It is important to realize that notes cannot be directly mapped to keys on the keyboard. There are more notes than keys and some notes share the same key and sound while having

[9] `https://docs.unity3d.com/Manual/Sprites.html`

different representations within sheet music. Therefore, only enum inputs of base keys (white keys, C-B) will be accepted, as a base key together with the accidental (flat,sharp,neutral) fully define a note. All notes will be drawn according to all basic rules of notation. Notes above or below a staff will have ledger lines added, notes that are in the upper half of the staff will be flipped and chords will have uniform alignment. The draw methods will return the drawn `Note` objects to the caller.

Notes are represented as game objects which can be instantiated at runtime by other classes, also known as prefabs[10]. All notes have a `Note` script attached to them. As explained above, the instantiation is performed by the `SheetMusic` script and the `Note` instance is returned to the caller. A note prefab has a type assigned to it, and when instantiated will automatically load the corresponding Sprite and render it. However, only the quarter type is used for this prototype as the focus is on harmonic theory and no rhythmical patterns or time signatures are explored. To change the color of a Note, a Sprite of that color must be included by the developer. The Sprite is then loaded by the `changeColor()` helper method.

To provide the note dragging functionality specified in subsection 3.2.2, the `Note` script implements the MRTK `IMixedRealityPointerHandler`[11] interface. It allows the customization of MRTK pointer interactions. By default the MRTK pointers are emitted from the hand and function similarly to a mouse. When a certain hand gesture is performed, the `OnPointerDown()` method is called. While the gesture is held, the `OnPointerDragged()` method is called every frame. Once the gesture is released, the `OnPointerUp()` method is called. Through overriding these methods, functionality may be implemented. In this case, when `OnPointerDown()` is called, the position of the hand is noted. During `OnPointerDragged()`, if the hand moves up or down relative to the base position, the note is incremented or decremented respectively. If the hand is moved left or right, a flat (left) or a sharp (right) is added to the note. Every change in the note is relayed to the `MusicController`.

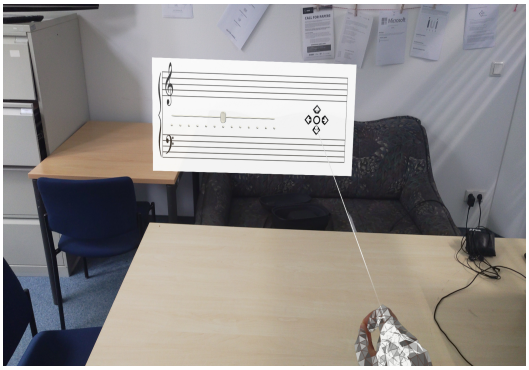
In order to allow the annotation of notes within the sheet music, a simple prefab was created. It consists of a game object containing a text box and has the `Tooltip` script attached to it. When instantiated the script will automatically draw a line towards up to two other game objects. The lines are continuously adjusted depending on the positioning of the tooltip and the attached objects. This allows for easy annotation of content within the sheet music at runtime.

Finally, to allow the user to position the sheet music freely within his vicinity, it is added to a parent game object together with a menu which includes a slider and an anchor. The slider can be used to adjust the scale of the sheet music while the anchor can be grabbed to move the sheet music around. To enable this grab-and-move behaviour

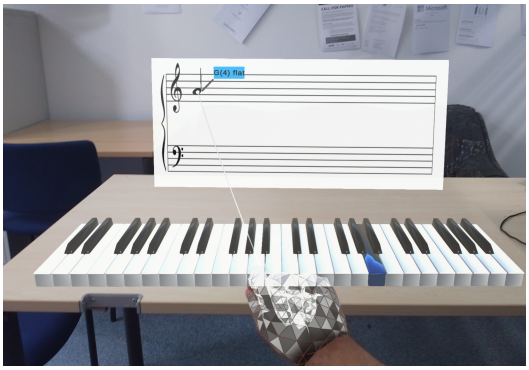
[10] `https://docs.unity3d.com/Manual/Prefabs.html`

[11] `https://docs.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.input.imixedrealitypointerhandler`

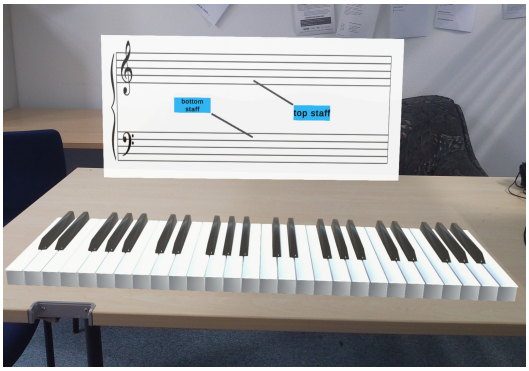
an MRTK `ObjectManipulator`[12] is added to the anchors game object. The position of sheet music is then connected to the anchor by using a Unity `ParentConstraint`[13].



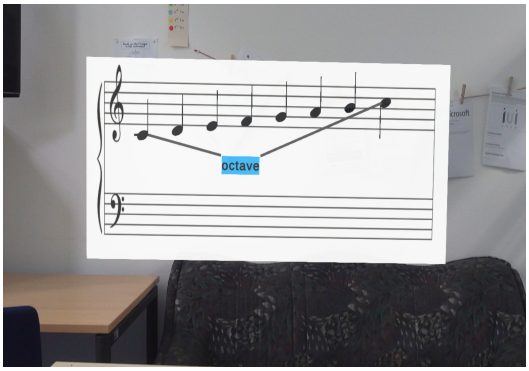
(a) Positioning the sheet music



(b) The note dragging interaction



(c) Tooltips pointing to a single anchor



(d) Tooltip pointing to two anchors

Figure 4.2: Sheet music features

4.3 Music Controller

As described in subsection 3.2.3 the `MusicController` is the architectural centerpiece of the application. Since all other classes have to access it and there will always ever be only one instance, the Singleton pattern was used to implement it. As it is also the only component that actually plays sound, it must manage all 48 `AudioClip`[14] objects which represent sound files in unity. The files used are samples of a Steinway Model B

[12] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/object-manipulator`
[13] `https://docs.unity3d.com/Manual/class-ParentConstraint.html`
[14] `https://docs.unity3d.com/Manual/class-AudioClip.html`

grand piano provided free of charge by the Ivy Audio project[15]. To make swapping between different sample sets easy, they are provided to the `MusicController` as a Unity `ScriptableObject`[16] named `PianoVariant`.

The `MusicController` keeps track of all active keys. If a `KeyPressedHandler` informs it that a note has been pressed by calling the `keyPressed()` method, it will play the corresponding sound, color the key and instructs the `SheetMusic` to draw that note as long as the user input is activated. By default all of the black keys are drawn as their sharp representation. The activation process is handled by the `noteActivated()` method. It allows specification of the color which the key and note should have, whether the note can be dragged by the user and if it should be drawn within the `SheetMusic`. For the standers user interaction default values are assigned to each of those parameters. However the tutorial can access the `noteActivated()` method directly, to have more options when trying to achieved very specific instructional goals. The deactivation process is handled by the corresponding `keyReleased()` and `noteDeactivated()` methods.

4.4 Tutorials

With all other components setting the baseline, the tutorials form the educational core of the application. They present the different harmonic theory concepts and challenge the user with exercises. As tutorials run sequentially, something similar to the software design concept of a thread is required to implement them. The standard solution Unity offers for multi-frame processing is the `Coroutine`[17] system. Within a `Coroutine` the developer may pause execution to wait for another `Coroutine`, a boolean trigger or simply a certain time frame.

A `Coroutine` must be started by a class which derives from `MonoBehaviour`. However, a `MonoBehaviour` cannot be instantiated without being attached to a game object. As tutorials are more of an abstract concept which is not directly placed within the scene, the tutorial system was split into two main classes. The tutorial `TutorialRunner` derives from `MonoBehaviour`. It provides some user interaction feature and manages and runs the tutorials. `Tutorial` is an abstract class that does not derive from `MonoBehaviour`. It contains the actual code which a `Coroutine` executes (i.e. Methods that have the return type `IEnumerator`) and provides helper methods which encapsulate commonly used code to make the tutorials themselves less verbose. All tutorials inherit from the abstract `Tutorial` class and overwrite the `tutorial()` method that the `TutorialRunner` runs. A class diagram of the tutorial system can be seen in Figure 4.3.

As defined in subsection 3.2.4 the tutorials will use TTS as well as speech input to avoid visual cognitive overload. The TTS system is placed on a separate game object

[15] `https://www.ivyaudio.com/Piano-in-162`
[16] `https://docs.unity3d.com/Manual/class-ScriptableObject.html`
[17] `https://docs.unity3d.com/Manual/Coroutines.html`

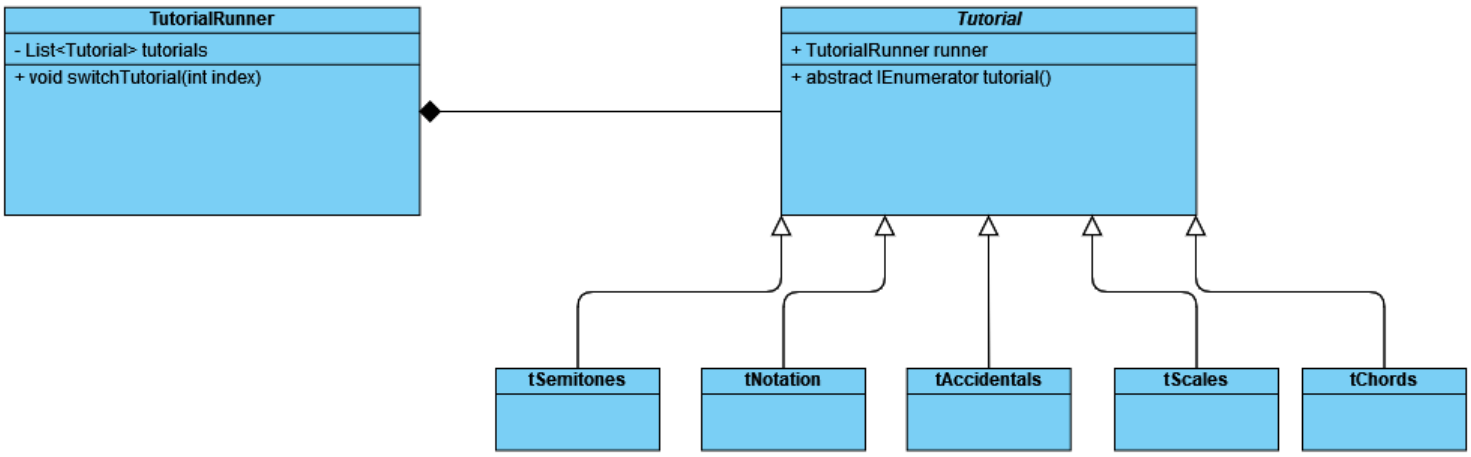


Figure 4.3: Tutorial System (other attributes and helper methods not included)

and utilizes the MRTK `TextToSpeech`[18] script. A custom script(`TTS`) is attached to offer the TTS methods in a static context as well. The MRTK `SpeechInputHandler`[19] is attached to the `TutorialRunner`'s game object as the `TutorialRunner` script handles the user speech input.

4.4.1 Tutorial Runner

As mentioned above, the `TutorialRunner` class is responsible for managing which tutorials are running. When the application starts a List with one instance of every tutorial is created. The `switchTutorial()` method can be used to start a tutorial by giving the index of its location within the tutorial list. If another tutorial is running it will end it before starting the new one. When a tutorial is started the runner will set itself as the runner reference within the concrete version of the `Tutorial` class.

Additionally the runner provides helper methods to handle the user's speech input. Whenever the user speaks one of the defined keywords a boolean is set within the runner. The running tutorial can always reset a specific boolean and then pause its execution until the boolean is set to true by the user's speech input again. Exiting a tutorial is done over the speech input as well. Once the keyword "exit" is detected the user will be asked for confirmation by utilizing the MRTK dialog feature[20]. The runner will then stop the tutorial and preform all necessary cleanup actions.

[18] `https://docs.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.audio.texttospeech`

[19] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/speech#handling-speech-input`

[20] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/dialog`

4.4.2 Tutorial Class

The `Turorial` class provides the template for the individual tutorials in the form of an abstract class. The abstract method `tutorial()`, which must be implemented by the concrete tutorials, provides the sequential code which is run by the `Coroutine`. As all the tutorials must be hard coded, this method tends to end up being a very long list of different commands. Although a certain length is unavoidable, some helper methods were introduced to somewhat alleviate this problem. For example the tutorial must pause while the TTS is speaking text. If written within the tutorial it would appear as follows:

```
if(TTS.isSpeaking()) { TTS.stopTTS(); }
TTS.speakText(text);
if(addToRepeat) { repeatText += "␣" + text; }
yield return new WaitWhile(() => TTS.isSpeaking());
```

This code snippet would have to be written every time the tutorial waits for the TTS. To avoid this, the snippet can be turned into a helper method instead:

```
protected WaitWhile speakAndWait(string text)
{
    if(TTS.isSpeaking()) { TTS.stopTTS(); }
    TTS.speakText(text);
    if(addToRepeat) { repeatText += "␣" + text; }
    return new WaitWhile(() => TTS.isSpeaking());
}
```

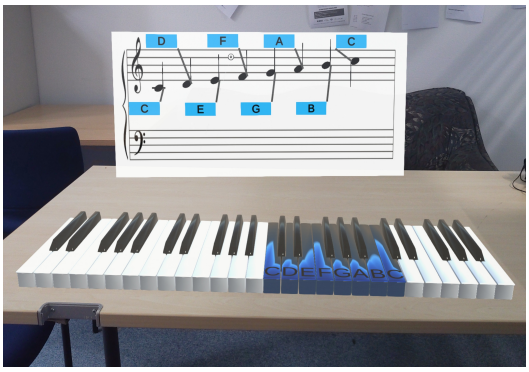
Now the only line that must be written within the tutorial itself is:

```
yield return speakAndWait(text);
```

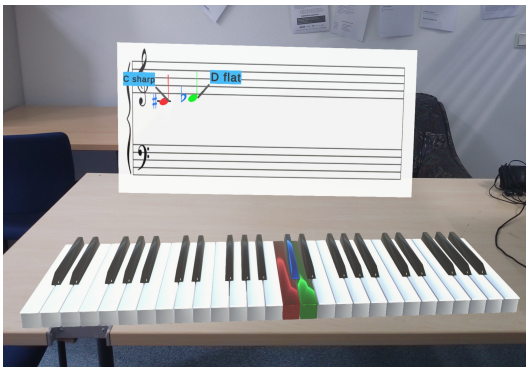
This approach was used for most helper methods within the tutorial class to significantly shorten overall tutorial length and improve readability.

As the tutorials are driven by TTS narration and visually presents information through coloring and tooltips, it is important to give the user time to take in all the information. Therefore the tutorial regularly pauses itself. The user must then say the word "continue" out loud to resume the tutorial. To ensure the user does not miss any important part of the narration, every time the tutorial is paused the user can say "repeat" instead of "continue" to hear the last text again. To achieve this all text spoken by the TTS is automatically added to a repeat queue. At each pause point, if the user moves on normally the repeat queue is reset. Important sounds are added to the queue as well and are repeated after the TTS has finished repeating the text.

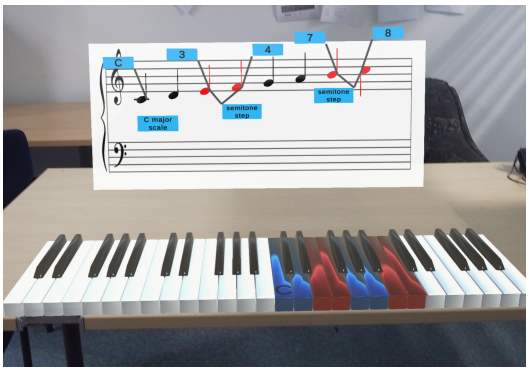
The exercises can be categorized under three basic types: Keyboard input, speech input and note input. For keyboard input exercises the user must find and press one or multiple keys to solve the task. This allows questions such as "Which key corresponds



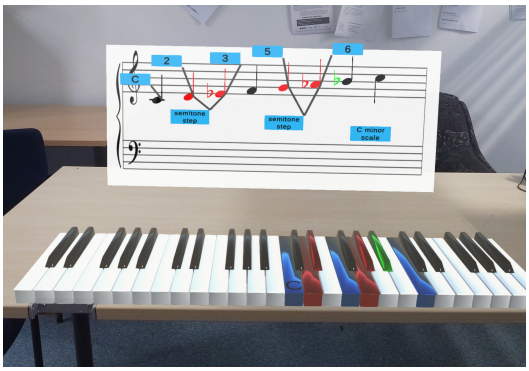
(a) Explaining note names



(b) Explaining flat and sharp key collision



(c) Explaining C major scale



(d) Explaining C minor scale

Figure 4.4: Tutorial snapshots

to this note?" or "Which key fits the blank". Speech exercises can query the user for note names or other keywords like minor, major, sharp or flat. The user must then say the correct answer out loud. This ensures the user is improving his knowledge of important terms within the learning process. Finally note input exercises require the user to drag notes to the correct positions within the sheet music. This makes open ended questions regarding harmonic concepts and notation possible. For example, the user could be asked to turn a minor chord into a major one or adjust notes to match a scale. Overall, these methods aim to provide a diverse spectrum of interaction types and the possibility to make engaging and challenging exercises.

4.5 UI Elements

As mentioned above, the navigation within tutorials is handled mostly through speech input. To save screen space when working with the limited field-of-view provided by the Hololens, all other menus are hidden by default. However, there is a startup

tutorial which explains all the controls together with how to place the keyboard and sheet music. The first menu is a MRTK hand menu[21]. It appears next to the users

(a) Hand menu (b) Main menu

Figure 4.5: UI element examples

hand when it is turned palm up and allows the readjustment of the keyboard and sheet music placement. It also can be used to access the main and the keyboard depth menus and lets the user quit the application. Those other two menus are MRTK near menus[22]. They float in mid-air a short fixed distance away and follow the users gaze. The main menu can be used to access the individual tutorials while the keyboard depth menu adjusts how deep the keyboard sits within the surface it is placed on. The latter was primarily introduced to evaluate what type of tactile feedback users prefer when interacting with objects placed on surfaces such as the piano.

Another concrete design question was how to implement short instruction text during the tutorials in order to complement the TTS instructions. As all text placed in the scene would be located at a static position by default, a head-up display style was chosen instead. Two text elements were placed at fixed positions at the top and bottom of the screen to ensure the user can always see them. This should ensure that the user always knows the task at hand, even if some part of the TTS was not understood. It is important to note, that the head-up display text for unknown reasons did not appear on screenshots. Additionally to this, the field of view in screenshots is significantly larger than when actually wearing the device. Therefore, the often mentioned limited field of view is not correctly represented by the screenshots. When running the application everything is notably more cramped than it appears on the screenshots.

[21] `https://docs.microsoft.com/en-us/windows/mixed-reality/design/hand-menu`

[22] `https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/near-menu`

5 User Study

To gather first feedback on the application, a small user study with eight participants was conducted. The users had the opportunity to test the application and note their thoughts in a qualitative survey. Due to time constraints one participant choose not to take part in the questionnaire, which lead to only seven responses being recorded. It is important to note that the intention of this evaluation in not to present any statistically significant findings or prove any hypothesis, as the small number of participants would not support any general conclusions. Instead it aims to identify potential design, usability and implementation problems with the application itself as well as the Hololens framework.

It is important to point out that all but one participant were university students or adults. As the app was not designed especially for children no special effort was made to seek out minor testers. This might present a certain age bias within the results. All questions were either scored on a five-point Likert scale (5 being the best answer) or if not applicable, a multiple choice selection was used. Some questions allowed comments to further clarify the answers or provide additional feedback. The complete summary of all questions and answers can be found within the Github repository[1].

The first few questions focused on what prior experience participants had. Four participants hadn't come in contact with music theory since high school, while the remaining three played an instrument regularly. As similar result appeared when asked for their experience with AR. Four had no previous experience, while two had used other head mounted devices and only one person had used a Hololens prior.

5.1 Input Methods

After launching the application the participants went through a tutorial explaining the controls and input methods. When things were unclear the supervisor provided further explanations. Overall the users thought this tutorial did a good job explaining the controls. It was scored a 4.29 out of 5 with no score below 3. Despite this, all users needed further explanation of the air tap gesture[2] which is used as the default click by the Hololens OS for objects not within touching distance. Many struggled with

[1] https://github.com/LeonBrooks/AR-Music-Theory-eLearing/tree/main/Thesis%20Related%20Data/Survey%20Results

[2] https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/operator-gestures-hl2#air-tap

consistently performing it during the further use of the application as well.

Questions regarding input methods themselves were split into three categories. The speech input, the keyboard input and the note input (see chapter 4). For every question the users scored each input method individually. All results can be found in Table 5.1. When asked how easy the input methods were to understand, the keyboard input scored particularly well, with an average 4.86 and no answer below a 4. The speech input method was also quite easy to understand, scoring a 4.43, while the note input was not as easy to comprehend, only scoring a 3.86.

When asked how well they thought the input method actually allowed them to input what they intended, a few users appeared to have some troubles. The keyboard input scored an overall 2.43, however the individual answers were quite mixed. The speech input was better received as all users scored it a 3, 4 or 5 with an average score of 3.86. Similarly to the keyboard input, the note input only scored a mean 2.57, although the individual answers were not as far apart as with the keyboard input. When asked what they thought of the three methods overall, the keyboard input continued to split the participants opinions. It had users give it every score from 1 to 5, averaging out at 2.71. The speech input was slightly better received at an average 3.29 but had users less split in opinion. Just like the keyboard input, the note input was scored a mean 2.71 with very split opinions. Two testers scored it a 5, two scored it a 1, two more scored it a 2 and the remaining tester scored it right in the middle at 3.

method	1	2	3	4	5	avg. score
How quickly did you understand how the input methods worked?						
keyboard	0	0	0	1	6	4.86
speech	0	0	1	2	4	4.43
notes	0	1	2	1	3	3.86
How well did you feel the methods allowed you to input what you intended?						
keyboard	2	1	3	1	0	2.43
speech	0	0	2	4	1	3.86
notes	0	4	2	1	0	2.57
How much did you like the different ways of interacting with the app overall?						
keyboard	1	2	1	2	1	2.71
speech	0	2	1	4	0	3.29
note	2	2	1	0	2	2.71

Table 5.1: Summary of input method evaluations.

Overall the results showed that the input methods leave quite a bit of room for improvement. Although some of the problems might be due to the limitations of the Hololens (in particular the hand tracking robustness), others might be due to the design of the methods as well as reliability problems within the implementation. Especially the note dragging had many users struggle and occasionally caused frustrations. This

sentiment was mirrored by the comments as well as the instructors observations. Many had trouble grabbing and releasing the notes reliably causing them to end up in completely different places than the user intended. This can be seen as a lack of familiarity with the Hololens hand tracking and the air tap gesture. However, there appeared to be problems with the grabbable surface area of the notes as well which is more likely due to problems in the concrete implementation within Unity.

As some problems with the keyboard were to be expected, the survey contained the question whether the users thought a physical keyboard would have been a better input method. The answer options were "I don't know", "Yes" and "No". Two testers weren't sure and answered I don't know. All other participants thought a physical keyboard would have been the better option.

5.2 Tutorials

The harmonic theory tutorials were the second primary evaluation target. When asked how well the users understood the content of the tutorials, no one scored their comprehension below 3 for an overall average of 4.29. The participants especially liked the coloring as well as the annotation feature. On the question of how helpful these tool were, for both of them a score of 5 was given six times while the remaining score was a 4. This is an overall mean of 4.86. One of the main goals when designing the application was to help users better understand the connection between sheet music and the keyboard. Besides the Likert scale an additional option was given for testers who though they already completely understood the connection prior to using the application. Three participants chose that option, while the remaining ones gave an overall score of 4.25 with no score under 4.

While designing the application (see chapter 3) it was decided to use audio in the form of text-to-speech as the primary narration tool. This was in part based on the recommendations of papers reviewed in chapter 2, which suggested audio could help reduce cognitive overload. Another main factor was the limited field of view provided by the Hololens. A question aiming at evaluating this decision was presented to the users during the survey. They were asked whether they though audio was best, if they preferred text or if the combination of both would have been a better option. If they thought text was the better option, they were also explicitly asked to comment on where on the screen the extra text could have been placed. Four testers though audio was better, one would have preferred text and two would have liked both. No user commented on where they would have added text, while one explicitly commented that text would have been too complicated to follow. Although no concrete conclusions can be draw due to the small sample size and mixed results, the management of field of view and cognitive capacity does seem to be a difficult question with no clear cut answer.

When asked how much they liked the tutorial overall most users gave a score of

	1	2	3	4	5	avg. score
How well did you understand the content of the music tutorials?						
	0	0	2	1	4	4.29
How much do you think tooltips and the coloring of keys/notes helped you understand the content?						
tooltips	0	0	0	1	6	4.86
coloring	0	0	0	1	6	4.86
Did the app help you better understand the connection between notes in sheet music and keys on the keyboard?						
	0	0	0	3	1	4.25
How much did you like the music tutorials overall?						
	0	1	0	5	1	3.86

Table 5.2: Summary of tutorial evaluations.

4, while one chose 5 and one chose 2. However, the user scoring a 2 made sure to inform the supervisor of his personal disdain for music theory during the test and in the survey, commented that he thought the application was a good instruction tool nevertheless.

5.3 Overall Questions

Another important consideration which arises together with the use of augmented and virtual reality head-mounted devices, is the question of physical discomfort. As both the keyboard and the note input require hand movement, the tester were asked specifically about arm fatigue. With 1 representing no fatigue and 5 representing soreness on the Likert scale, the participants gave mixed answers. Every value except for 5 was represented within the results with the average being 2.14. Although the average might make it seem like physical exhaustion is no big problem, the mixed answers indicate that a developer should always keep this factor in mind when designing an application to ensure no user is left behind. Two users also reported other physical discomfort, namely a headache and trouble with the display being too bright.

At the end of the test all users were explicitly asked by the supervisor to try out the different keyboard depth options. There were three different settings. For high depth the keyboard clearly floated above the table. With medium depth the keys were slightly above or touching the surface as if they were real keys placed on a table. Finally, with

low depth they were slightly submerged within the surface, so that the point of the keys activating the corresponding sound was about the same as the point where the finger touches the table. Four users preferred the low option and three the medium option while no one chose the high option. The testers also had the possibility to comment on why they chose a certain depth. One user said he liked the low depth because it most closely resembled a real piano, however another commented that low depth caused the activation point to be too far within the table which made the keys hard to press. Because the Hololens spatial awareness system, which detects the surfaces, is of course not 100% accurate, it can cause slight differences which can lead to users having different experiences with the same setting. Although some users might prefer an activation close to the surface, the developer should watch out to ensure it does not lead to a serious usability issue for others.

	1	2	3	4	5	avg. score
Did you enjoy using the app overall?						
	1	0	0	6	0	3.57

Table 5.3: Scores for the application overall.

When asked how much users liked the application as a whole all users, except the one who had expressed his personal dislike for music theory, gave a score of 4. Overall this means that although many had problems with the input methods and getting used the the Hololens framework, almost all enjoyed using the application. This is in line with the observation made at the end of section 2.2, that AR often produces high engagement and enjoyment ratings. In particular Radu [20, p. 1536] notes that although usability being a common issue, sometimes AR systems which were harder to use than the non-AR counterpart were able to produce higher motivation ratings.

6 Challenges and Lessons Learned

During the design, implementation and user testing many different challenges were encountered and observations made. Although some of these point might have already been explored in different literature, the combination of the music theory, AR and learning context can offer some interesting unique insights. The goal of this chapter is to analyze these challenges and observations in order to extract key takeaways and design simple guideline for future similar applications.

6.1 Design Takeaways

Although basic software architecture, sketching and class modeling was done prior to the concrete implementation, many of the finer details were decided gradually during the coding process. This approach was chosen due to a limited time frame and a rather small application size. The first component implemented was the keyboard, then the music controller, followed by the sheet music and finally the tutorials were added. This of course led to the final application having a few shortcomings in the software design context. Understanding and avoiding these might greatly benefit future applications, particularly when trying to offer forward compatibility.

6.1.1 The Difference between Notes and Keys

When developing an application combining sheet music notes with actual sounds, one must realize that there does not exist a one-to-one mapping between the two. An individual note will always describe the sound of exactly one key on the keyboard. However, the sound of an individual key on the keyboard can represented by multiple notes. This presents a unique challenge during software design. As the keyboard and music controller were implemented before the sheet music within this application, the former two were operating by defining keys and sounds through an enum which held an entry for every key on the keyboard. When later implementing the sheet music, it was realized that a mapping from the key enum to notes, and more importantly the reverse way, was required. This was then implemented with in the `SheetMusic` and `Note` class. These late adaptions led to overall "ugly" method heads, interfacing between these two systems and quite a few helper methods being required. Although the final implementation was serviceable and not too complex, there is definitely a lot of room for improvement from a software design perspective. While no concrete solution will

be presented here, for the development of future applications it is highly recommended to consider this specific question during the design phase. It might be useful for larger applications to develop a dedicated system responsible for the translation between notes and keys which unifies the data types and interfaces between all components of the program.

6.1.2 Messaging and Interfacing

Similarly to the previous point, the data flow and messaging between the different components of the applications was not optimal. As the keyboard and music controller were implemented first, the idea was that they should function like a normal instrument before moving on to the implementation of the sheet music. This meant that a keyboard key could be pressed and the corresponding sound would be played, before work on the sheet music was started. The key would also already show a color change on press. Later on, when the sheet music was being built, it was no problem to make the corresponding note appear when a key was pressed. Up until this point, all information and messaging was mainly unidirectional. The key press on the keyboard would trigger sound in the music controller and the music controller would trigger the note drawing. Although the music controller kept track of which notes were active, the activation and deactivation was closely linked to the keyboard input.

Later, when the note input and the tutorial were designed, a reverse flow of information was required. The notes/tutorial would inform the music controller of a activation/deactivation and that would trigger a change in the sound and coloring of the keyboard. As the systems were previously implemented with the initial flow of information in mind, this required high adaptation to prevent unwanted effects like circular note drawing. Many of the methods, in particular the activation/deactivation methods, had to be amended with many new parameters and the containing classes had to include new custom variables. Although this allowed the tutorials highly custom and specific access to all components, it also caused overlapping functionalities and resulted in a somewhat "messy" style of coding.

For future applications it is therefore recommended to not design and implement their systems starting from an instrument functionality perspective. Instead it might be useful to more strongly separate the different components and event triggers (instrument input, note input, tutorial calls, etc.) and model the central component only after the other components' interfaces have been fully developed. This can ensure that the central component can effectively keep track of all active notes and activation/deactivation requests by other components. It can then decide more effectively how to handle them and provide conciser methods to the remaining components. Overlapping functionalities can be prevented as well.

6.2 Usability Takeaways

During private testing and the user study a few usability issues, which only become apparent during concrete testing, were discovered. Most of these were directly connected to developing on the Hololens platform or an AR-head-mounted device in general. Three points stood out in particular:

- *Limited Field of View:* As mentioned in chapter 4, while developing and testing in emulation tools like the Unity editor, one does not realize how limited the available field of view is. When building a tutorial application this can become constricting rather quickly. In this case, the keyboard together with the sheet music almost completely filled out the field of view already. As primary solution this project chose text-to-speech to deliver a lot of text which could not have been fit on the screen otherwise. The little remaining space available at the top and bottom of the screen was used to deliver short text instruction to guide the user if they missed key audio cues. Although this was barley enough space to avoid overlaying text and other components, some users would have preferred more text instructions (see section 5.2). However, even despite being asked explicitly, no user suggested an appropriate spot to place more text, proving this to not be a trivial problem. Developers of future applications should be aware of the limited field of view and design around it. This could mean placing text outside the field of view, which would in turn require constant user head movement to navigate the application. Another option could be hiding and displaying content/text according to the moment to moment requirements. In general, new approaches and experimentation should be encouraged.

- *Barrier of Entry:* During the user tests all users had initial trouble getting used to the Hololens interaction methods. In particular the air tap gesture[1] caused many user trouble. All users were given additional instructions about this gesture by the supervisor. While some manged to adequately perform it soon after, others struggled throughout the whole test. As this fairly simple gesture caused many users severe usability problems, future applications should carefully choose which interaction methods to include. More complex interactions and gestures might easily alienate parts of the the user base and if included anyway, must be thoroughly explained through appropriate tutorials. Although the supervisor performed examples of the gesture, the application itself did not present any visualization during the interaction tutorial. This might have been a not insignificant contributing factor to the mentioned issues and should be improved by future applications.

[1] `https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/operator-gestures-hl2#air-tap`

- *Hand Tracking:* Although Hololens hand tracking is fairly state-of-the-art (it was presented in 2019), the usability is still somewhat limited. In the case of the Hololens this is in part due to it being based on head-mounted cameras. This means it only works in a limited space in front of the user and only if the vision is not obscured or covered (e.g. fingers hidden by the palm when pressing keys). Additionally, while the tracking of thumb, index finger and pinky is fairly sophisticated, the middle and ring finger are more error prone and sometimes tracked as a single unit. Within the prototype this led to the keyboard not performing like its real world counterpart. Instead, users were usually only able to slowly press up to two keys with each hand before hitting others by mistake. Overall, it can therefore not be recommended to ask the user to perform any type of precises movement with their fingers and error tolerance should be part of the design to prevent user frustration.

7 Conclusion

This thesis presented the complete development process of an augmented reality e-Learning application for basic harmonic theory. As the current literature in this very specific subdomain is especially limited, the goal was to present a comprehensive rundown, track noteworthy observations and extrapolate useful takeaways for future applications.

A brief literature review examining related applications, the value of AR in an e-learning context and aspects of music pedagogy was conducted. The resulting findings were applied to the design phase and requirement definition of this application. Furthermore, the five main components of the application were defined and the requirements divided onto them. The components were modeled and implemented as classes/scripts within the Unity game engine and the Microsoft Mixed Reality Toolkit targeting the Hololens 2 head-mounted display device. Chosen approaches were discussed and key classes and methods explained.

Following the implementation, an eight participant user study was conducted of which seven chose to take part in a qualitative survey. The three main input methods (virtual keyboard input, speech input, note dragging) were evaluated on different aspect within a five point Likert scale (5 being the best score). When asked for an overall rating the input methods scored an average 2.71, 3.29 and 2.71 points respectively. The harmonic theory tutorial achieved and overall score of 3.86. As a whole, the application received a score of 3.57. Additionally, more concrete questions concerning the input methods, tutorials and application overall were examined as well (see chapter 5).

Challenges and key takeaways which appeared during the development and testing process were summarized. On the software design side these were:

- Notes and keys/sounds cannot be mapped on a one-to-one basis and therefore represent different data types. Future applications should consider this during the design phase. A system managing and translating between these two data types might be useful.
- Designing a music application by starting all data flow at the instrument might prove to be problematic in the long run. Future Applications should separately define input events and only then design (central) handling.

Additionally the following takeaways were summarized on the usability side:

- The Hololens provides a very limited field of view. Especially in the context of e-learning this might prove troublesome, even more when considering the

placement of text. Audio is an option to alleviate this problem. Other new approaches should be sought out as well, especially as some users might prefer text (additionally) to audio.

- The Hololens/AR in general may present a significant barrier of entry to some users. Particularly gestures can present a hindrance and should be kept simple. Additionally, all gestures should be thoroughly explained, if possible with the help of visualization.
- State-of-the-art (camera based) hand tracking is not fully accurate. Tasks using it should be designed with error tolerance in mind and should be kept simple.

8 Outlook

As the presented application is only a small prototype, there are many conceptual ways it may be improved. When considering the subpar reception of the virtual keyboard input method by users in chapter 5, an obvious modification that comes to mind is including a physical keyboard instead. All users that decided to given an opinion on which was the better option would have preferred a physical keyboard instead. Besides a slightly increased implementation effort and the question of keyboard availability, the main disadvantage of a physical keyboard is the innate lack of coloring and labeling capabilities. However, works like Hackl et al. [10] and Huang et al. [12] have shown it to be viably to overlay such information onto a physical piano as well. If combining these different approaches a physical piano could provides a far better user experience and further the implementation of design goals like embodiment and context visualization.

Due to time constraints and the general scope of this project, the chosen tutorial style mirrored very classical school like education approaches. It first explained the harmonic theory concepts through narration and visualization in sheet music and on the keyboard. Then it offered the users a way to test and practice their previously acquired knowledge through simple exercises. One of the major advantages of an e-learning application not included here is the usage of gamification and narrative storytelling. While being one of Dunleavy's guidelines [8] examined in subsection 2.2.2, these principles are common in most modern day learning applications and would surely benefit the application as a whole.

Lastly, this thesis mainly explored pedagogy concepts already related or adapted to the AR context. As this project was mainly tech focused, many ideas from the fields of music and overall pedagogy were, if at all, only mentioned briefly. Future developers should be encouraged to actively seek out key takeaways from these fields of study and find creative ways to implement them within their applications.

List of Figures

List of Tables

Bibliography

[1] F. Avanzini, A. Barate, C. Mauro, L. A. Ludovico, and M. Marcella. "Developing music harmony awareness in young students through an augmented reality approach." In: *CHIRA International Conference on Computer-Human Interaction Research and Applications*. Science and Technology Publications. 2020, pp. 56–63.

[2] R. T. Azuma. "A survey of augmented reality." In: *Presence: teleoperators & virtual environments* 6.4 (1997), pp. 355–385.

[3] J. L. Bacca Acosta, S. M. Baldiris Navarro, R. Fabregat Gesa, S. Graf, et al. "Augmented reality trends in education: a systematic review of research and applications." In: *Journal of Educational Technology and Society, 2014, vol. 17, núm. 4, p. 133-149* (2014).

[4] O. Cakmakci, F. Bérard, and J. Coutaz. "An augmented reality based learning assistant for electric bass guitar." In: *Proc. of the 10th International Conference on Human-Computer Interaction, Crete, Greece*. Citeseer. 2003.

[5] M. R. Callahan. "Teaching and learning undergraduate music theory at the keyboard: Challenges, solutions, and impacts." In: *Music Theory Online* 21.3 (2015). Includes supporting video/audio clips, best viewed at: `https://mtosmt.org/issues/mto.15.21.3/mto.15.21.3.callahan.html`.

[6] 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.

[7] M. J. Cook. "Augmented reality: Examining its value in a music technology classroom. Practice and potential." In: *Waikato Journal of Education* 24.2 (2019), pp. 23–38.

[8] M. Dunleavy. "Design principles for augmented reality learning." In: *TechTrends* 58.1 (2014), pp. 28–34.

[9] A. Dünser, R. Grasset, H. Seichter, and M. Billinghurst. "Applying HCI principles to AR systems design." In: (2007).

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

[11] S.-Y. Ho, W.-W. Jiang, Y.-M. Yu, and M.-Z. Li. "Using Context Awareness and Augmented Reality to Build an Instruction System for the Fundamental Music Theories." In: *Applied Science and Management Research* 4.1 (2017), pp. 137–150.

[12] 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.

[13] H. Kaufmann and A. Dünser. "Summary of Usability Evaluations of an Educational Augmented Reality Application." In: *International Conference on Virtual Reality*. Springer, 2007, pp. 660–669.

[14] J. R. Keebler, T. J. Wiltshire, D. C. Smith, S. M. Fiore, and J. S. Bedwell. "Shifting the paradigm of music instruction: implications of embodiment stemming from an augmented reality guitar learning system." In: *Frontiers in psychology* 5 (2014), p. 471.

[15] K. Kelley and K. J. Preacher. "On effect size." In: *Psychological methods* 17.2 (2012), p. 137.

[16] Y. Lu, X. Wang, J. Gong, and Y. Liang. "ChordAR: An Educational AR Game Design for Children's Music Theory Learning." In: *Wireless Communications and Mobile Computing* 2022 (2022).

[17] R. E. Mayer. "Multimedia learning." In: *Psychology of learning and motivation*. Vol. 41. Academic Press, 2002, pp. 85–139.

[18] D. A. Norman. *The psychology of everyday things.* Title later changed to: "The design of everyday things". Basic books, 1988.

[19] B. Patzer, D. C. Smith, and J. R. Keebler. "Novelty and retention for two augmented reality learning systems." In: *Proceedings of the human factors and ergonomics society annual meeting*. Vol. 58. 1. SAGE Publications Sage CA: Los Angeles, CA. 2014, pp. 1164–1168.

[20] I. Radu. "Augmented reality in education: a meta-review and cross-media analysis." In: *Personal and ubiquitous computing* 18.6 (2014), pp. 1533–1543.

[21] M. E. C. Santos, A. Chen, T. Taketomi, G. Yamamoto, J. Miyazaki, and H. Kato. "Augmented Reality Learning Experiences: Survey of Prototype Design and Evaluation." In: *IEEE Transactions on Learning Technologies* 7.1 (2014), pp. 38–56.

[22] L. Shapiro and S. Spaulding. "Embodied Cognition." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2021. Metaphysics Research Lab, Stanford University, 2021.

[23] B. E. Shelton and N. R. Hedley. "Exploring a cognitive basis for learning spatial relationships with augmented reality." In: *Technology, Instruction, Cognition and Learning* 1.4 (2004), p. 323.