FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Designing a Musical User Interface for Non-Musically Trained People

Including a Mechatronic String Instrument

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Media and Human-Centered Computing

eingereicht von

Jakob Michael Blattner, BSc
Matrikelnummer 1026117

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr.techn. Hilda Tellioğlu
Mitwirkung: Univ.Ass. Dipl.-Ing. Peter Fikar, Bakk.techn.

Wien, 13. Dezember 2018

Jakob Michael Blattner | Hilda Tellioğlu

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Designing a Musical User Interface for Non-Musically Trained People

Including a Mechatronic String Instrument

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media and Human-Centered Computing

by

Jakob Michael Blattner, BSc
Registration Number 1026117

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr.techn. Hilda Tellioğlu
Assistance: Univ.Ass. Dipl.-Ing. Peter Fikar, Bakk.techn.

Vienna, 13th December, 2018

Jakob Michael Blattner | Hilda Tellioğlu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Jakob Michael Blattner, BSc
Mittelgasse 5/3, 1060 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Dezember 2018

Jakob Michael Blattner

Acknowledgements

Author: *Jakob Blattner*

First, I would like to thank my parents, who have always supported me in my education and made me the person I am today. I also thank my girlfriend and friends who have supported me with words and deeds in the course of this thesis.

Furthermore, I would like to thank our supervisor Associate Prof. Dipl.-Ing. Dr.Hilda Tellioğlu and Assistant Dipl. Ing. Peter Fikar for the support and time they both have invested in the supervision of this thesis. I am also grateful for the help and advice Mag. Dr. Roman Ganhör gave us in the field of electrical engineering. Great thanks also goes to the Technical University of Vienna, which provided us with the financial means for this work through a scholarship

Last but not least, I would like to thank my colleague Raphael Kamper, without whose dedication and perseverance this work would not have been possible.

Thank you!

Abstract

Music has been an integral part of every culture and every human being since time immemorial, including the creation of music. It usually takes years to master an instrument and its playing technique so that well-sounding music can be produced. A circumstance that for many people is, for a variety of reasons, a major obstacle. With the increasing availability of microcontrollers and other electrical equipment over the last decade, technical solutions can be used to address a wide range of problems. So too, people who would otherwise not have the opportunity to provide an outlet for their musical creativity and to give a sense of accomplishment in creating a musical entity.

The aim of this thesis is to design, implement and evaluate a haptic, non-collaborative user interface with tangible input. The resulting system is intended to assist the user in composing music while at the same time analogously generating the sounds produced by a mechanical component on which guitar strings are mounted. The system was developed primarily for people without prior musical knowledge. For this reason, a comprehensive literature research was carried out, through which the authors acquired knowledge in various topics. These include the area of user interfaces, user interface design, feedback, music combined with technology and mechanical music production. In order to deepen this knowledge and to include the opinions of specialists, interviews with experts from the affected areas were held. The acquired knowledge was implemented in an iterative design process. The mechanical component of the system was implemented without the assistance of users, since the fulfillment of technical benchmarks was sufficient here. The system component, which has direct contact with the users via the user interface, was implemented in a selected iterative and user-centered design process. Various research methods were used in this process, such as sketches, wireframes, mockups, personas and above all user tests. In these user tests, the current state of the system was checked to find out if the interface allows the user to produce well-sounding music and to sufficiently support it. The results of the study show that the chosen approach forms a promising basis that needs further development in additional iterative design cycles.

Kurzfassung

Musik ist seit Menschengedenken ein wichtiger Bestandteil jeder Kultur und jedes Menschen, so auch die Erschaffung von Musik. Es dauert in der Regel jedoch Jahre, bis man ein Instrument und dessen Spieltechnik so beherrscht, dass wohlklingende Musik erzeugt werden kann. Ein Umstand, der für viele Menschen, aus verschiedensten Gründen, ein großes Hinderniss darstellt. Durch die immer besser werdende Verfügbarkeit von Microplatinenen und anderen Elektrogeräten in den letzten zehn Jahren können technische Lösungen für eine Vielzahl von Problemenen eingesetzt werden. Somit auch, Menschen die sonst nicht die Möglichkeit hätten, ein Ventil für ihre musikalische Kreativität zu bieten und ein Gefühl der Erfüllung bei der Schaffung eines musikalischen Gebildes zu geben.

Das Ziel dieser Diplomarbeit ist es, eine haptische, nicht kollaborative Benutzeroberfläche mit greifbaren Eingabemöglichkeiten zu entwerfen, implementieren und auszuwerten. Das resultierende System soll den/die BenutzerIn beim Komponieren von Musik unterstützen und gleichzeitig die erzeugten Töne durch eine mechanische Komponente, auf der Gitarrensaiten aufgespannt sind, analog erzeugen. Das System wurde in erster Linie für Menschen ohne musikalischer Vorkenntnis entwickelt. Aus diesem Grund wurde eine umfangreiche Literaturerecherche durchgführt, durch sich die Autoren Wissen in unterschiedlichen Themengebieten angeeignet wurden. Dazu gehören die Gebiet Benutzeroberflächen, Benutzeroberflächen Design, Feedback, Musik in Kombination mit Technik und der mechanischen Musikerzeugung. Um dieses Wissen weiter zu vertiefen und die Meinungen von Experten miteinzubeziehen wurden daraufhin Interviews mit Experten aus den betroffenen Bereichen abgehalten. Das dadurch aquirierte Wissen wurde in einem iterativen Designprozess umgesetzt. Die mechanische Komponente des Systems wurde ohne mitwirken von Benutzern umgesetzt, da hier die Erfüllung von technischen Benchmarks ausreichend war. Die Systemkomponente welche über die Benutzeroberfläche direkten Kontakt zu den Benutzern aufweißt, wurde in einem ausgewählten iterativen und benutzerzentrierten Designprozess umgesetzt. In diesem Prozess wurden verschiedene Forschungsmethoden, wie zum Beispiel Sketches, Wireframes, Mockups, Personas und vor allem Benutzertests durchgeführt. In diesen Benutzertests wurde der momentane Stand des Systems überprüft um herauszufinden, ob die Oberfläche den Benutzer das Erzeugen von wohlklingender Musik ermöglicht und dahingehend auch ausreichend unterstützt. Die Ergebnisse der Studie zeigen, dass der gewählte Ansatz eine vielversprechende Basis bildet, die in weiteren iterativen Designzyklen noch weiter ausgebaut werden müssen.

Contents

CHAPTER 1

Introduction

Preliminary Note: This thesis is written by two authors: Jakob Blattner and Raphael Kamper. Therefore, *Author* marks at the beginning of each chapter, section or subsection indicate the according authorship.

Author: Jakob Blattner

Music has been an important part of every person's personality and self-identification since human recollection, allowing them to express their own creativity.

Like music, technology has evolved over the millennia and instruments through new possibilities in production, music production of sound and expression. These new possibilities also affect people's interaction with instruments, which has already been decoupled a few hundred years ago, as the pianola (see Subsection 5.2.2) illustrates. A trend which continued to develop into the modern age. Nowadays, with the availability of cheap microcontrollers providing enough computing power to process various input and output signals, there are plenty of musical interfaces that allow new forms of interaction to compose music. Most of these interfaces are connected to a computer for further processing with audio software. There is also the other way round that a music playing robot is controlled via specific software. In nearly all of these scenarios, the user isn't able to influence the music while it's being played. In either way the user has to have knowledge about the instrument, music and/or the software being used. Interfaces where the user can influence the created music are comparatively rare, interfaces specially for non-musically trained people even rarer.

The motivation of this work is to enable people with no experience in the music creation process to awake their curiosity and fascination for music and maybe even express their emotions. To reach this goal, the researchers of this thesis want to combine the analog sound creation from an instrument with user interaction via tangible devices, which foremost enables a natural way of interacting with a system. The thus generated melody is intended to generate a deeper connection to the own creation and promote the user's

interest in music. This combination of tangible user interfaces (TUIs) and analog music creation for non-musically trained people hasn't been done before.

1.1 Aim of this Work

The aim of this thesis is to design, implement and evaluate a haptic user interface that supports the user in composing music, which is mechanically generated by the strings of a plucked instrument, through visual, auditive and haptic feedback. This is crucial and there are numerous ways of providing feedback. This is especially challenging as there is in any case acoustic feedback by the tones played and the additional feedback of the system should not interfere with it.

Another important factor is the choice of configurable parameters and the according mapping into the system. As the interface should be usable by non-musically trained people (but doesn't specifically exclude musicians), the mapping of the input and the according output has to be intuitive, without the need to know something about notes, tone height, rhythm, etc. An area where tangible user interfaces provide excellent properties.

To assist the user while composing, the interface will provide suggestions on how to set those parameters to create *good* sounding music (as by middle-european standards), based on the basic principles of music theory. If there exist possibilities to help the user with other functionalities, they will also be added in the course of this thesis.

The resulting installation will be used in a museal context. In Vienna, for example, exist museums[1] in which the interactivity of their installations is a big part of their identity. In a settings as such, the initial interaction of possible users takes easier place than e.g. in a park or other public places. In addition, physical and weather-related external influences can be nearly left out completely as a factor to be considered in the creation process.

Another predefined property of the system to be created is the non-collaborativity. Most of the time an installation is either for one or for at least two users. Because of the fact that a collaborative approach automatically needs a minimum of two users to interact with the system and that circumstance is not always given, it limits the number of possible usages and can prevent an interaction in advance. The approach of designing an installation for both purposes is also more likely to fail to do either and would exceed the scope of this thesis.

Another pre-defined property is the usage of a string instrument to generate the analog output of the system. This definition seems narrowly defined, but the fact that not only a guitar or bass but also a zither, harp, and hurdy-gurdy are defined as string instruments, a variety of ways to strike a string and different sounds are possible and offer a wide range of possibilities.

[1] Like the Technical Museum and the Soundmuseum

1.2 Structure of this Work

The fields of user interface design, human computer interaction, robotics, mechanics and mechatronics as well as building music instruments are huge scientific areas, evolved for centuries or at least decades. Each of those fields has its own academic studies. To bring all those disciplines together in the context of designing a tangible, musical user interface for a robotic music instrument, the authors decided to split this thesis into five parts. The first part contains the theoretical foundations of this work, subdivided in *User Interfaces*, *User Interface Design*, *Machine Musicianship* and *Musical Mechatronics*. The purpose of the theoretical part is to gain fundamental knowledge in the areas in question and research for related projects.

The second part of this thesis contains the methodology of the practical part of thesis. The first chapter consists of the analysis of the related projects and the theory itself obtained in the first part of this work, including identifying possible hurdles and selecting the best qualified approaches for the following practical part. The second chapter includes the description of each user research method being applied in the course of the practical part.

The third part contains the documentation of the creation process of the sound generating, mechatronical part of this thesis. The creation process itself was iterative, but had no users take part in the evaluation process, as the mechantronical part only had to pass self-imposed benchmarks by the authors. In the end, it needed three design iterations to fulfill this benchmarks, as because every chapter contains one of this design iterations.

The following fourth part of the thesis contains the documentation of the design and creation of the musical user interface (MUI) and its tangible components. In it the users are a fundamental part of the evaluation, as the applied iterative design process is user centered. It contains a documentation of every method being described in the methodology and their repercussions on the ongoing development process. The creation of the MUI took two design iterations, one of each put in one chapter.

The final part of the thesis consists of two chapters. The first one contains a summary of the design process and its results, whereas the second one highlights possible and needed work to be done in future work.

Part I

Theoretical Foundations

Practice without theory is blind. Theory without practice is sterile.

Karl Marx, Capital, Vol.I, Preface to the French Edition, 1887, p.21

Author: Jakob Blattner

This part of the thesis covers all different areas on which the project of this work is built upon. There are five areas of which everyone is described in one chapter respectively. The first chapter of this part deals with user interfaces, their subdivision, history, properties, advantages and disadvantages and related projects. The following chapter is concerned with the design of user interfaces. This includes usability and utility, feedback, design processes, design knowledge and the process of gathering data about the interface. The next two chapter focus on *robotic musicianship*. According to Mason Bretan et al.[1], robotic musicianship:

> "[...] focuses on the construction of machines capable of producing sound, analyzing music, and generating music in such a way that allows them to showcase musicality and interact with human musicians."[1, p. 100]

They further distinguish between *musical mechatronics* and *machine musicianship*. *musical mechatronics* addresses the design of the physical sound generating system and *machine musicianship*

> "[...] focuses on developing algorithms and cognitive models representative of various aspects of music perception, composition, performance, and theory."[1, p. 100]

Musical mechatronics will be discussed in Chapter 5. This includes music theory, music perception, music algorithm and software and protocols. The last chapter of this part contains the last missing area of interest: machine musicianship. This chapter details the classification of musical instruments, mechanical, mechatronic and robotic musical instruments, and finally related projects.

CHAPTER 2

User Interfaces

Author: Jakob Blattner

The exponential rise of digital technologies in the twenty-first century shaped the interaction of humans with technology in every context imaginable [2]. Those changes affect among other educational, cultural, social and economical sectors of the world, which are driven by need for more efficiency and effectiveness of the underlying procedures. According to the Oxford Dictionary[1] effectiveness defines the *degree to which something is successful in producing a desired result* whereas being efficient describes *achieving maximum productivity with minimum wasted effort or expense.*.

Said demands don't stop from the field of *Human-Computer-Interaction* (HCI). HCI is the study of understanding and creating technology that people want to use, will be able to use and will find effective when used. According to Carroll [3] the development of user interfaces in the 60s and 70s of the 20th century was one of the four main roots from which HCI was formed. Simply said, a user interface is the part of a computer and its software of an application which enables the interaction between the user and the computer [4]. To be a bit more precise: the user interface can be seen, heard, touched, talked to, or otherwise understood or directed by the people. It has essentially two components: input and output. Input is how a person communicates his or her needs or desires to the computer [5]. Some common input components are the keyboard, mouse, the human skin (for touch-sensitive screens or pads), and someones voice (for spoken instructions). Output is how the computer submits the results of its calculations and requirements to the user. Nowadays, the most common computer output mechanism is the screen.

The first section of this chapter covers the topic of user interface in the context of this work. To do this, it will look at the two types of user interfaces, *Graphical User Interfaces*

[1] The Oxford Dictionary online

(GUIs) and *Tangible User Interfaces* (TUIs). The following section than talks about the conversion of theses types of interfaces in the so called *Musical User Interfaces* (MUIs). The following and at the same time last section describes projects and implementations which can be related to the thesis of this work.

2.1 Types of User Interfaces

This section will cover different types of user interfaces (UIs). As already stated in the introduction of this chapter, the covered types of interfaces will be limited to modern interface types, which excludes *Command-Line-Interfaces*. The following will give a short historical overview of the emergence of user interfaces, respectively HCI.

Before the 1960s, the term *user interface* was not defined [3]. The focus of computing was on computations, not on intelligibly presenting the results of those computations. Computers were expensive, precious and complicated machines where only a relatively small number of people were allowed to work with them [6]. Those highly guarded, room filling devices were reserved for scientists or engineers, which knew the usage of theses computers very well. Whether is was connecting relays with patch cords in the 1940s, changing magnetic memory drums in the 1950s, working with punch cards in the 1960s or writing command line inputs in the 1970s. The usage of a computer was highly exclusive. In the early 1980s, computers became more powerful and therefore also usable in work space and private environments. This resulted in new, different possible applications and in a much broader user base. Human-Computer-Interaction suddenly became a very important topic, illustrated by the publication of the book *The Psychology of Human-Computer-Interaction* by Card, Moran and Newell [7] in 1983.

2.1.1 Graphical User Interfaces

On the following pages, the author will define GUIs, give a historical overview of the most important developments, and show advantages and disadvantages of graphical user interfaces.

To begin with, it is necessary to first define what a graphical user interface is. Catarci [8] defines GUIs as follows:

> "Graphical User Interfaces are user interfaces that make extensive use of graphical objects (icons, diagrams, forms, etc.) that the user may directly manipulate on the screen through several kinds of pointing devices (including her/his fingers) and get an almost instantaneous feedback (near real-time interactivity)."

Communicating and perceiving information in GUIs is based on two dimensional visual signs [8]. Visual signs are also contained in pictures, photographs and geographic maps. They have many variables like size, intensity, texture, shape, orientation, and color.

Those signs all contain details about the information being communicated. But the basic attributes of graphical user interfaces are the following:

- The *Direct Manipulation Interaction* [9]: Direct manipulation was introduced by Shneiderman who described user interfaces which enable the user to do the following:
  - Visibility of the objects and actions of interest
  - Physical actions on the object of interest instead of complex syntax
  - Effects of those actions that are rapid, incremental and reversible
- The *Metaphor*: Designers use this term to refer to visual conventions and genres that, although familiar, need not resemble any real-world objects [10].
- The *Visual Representation*: Visual representation captures the visual signs that stand in for something else and takes its place. The basic of visual representation is the so called mapping. Mapping defines two sets of items. One set that are being represented and one set of visual elements that are used to represent them. Mapping should be expressive (neither lose any information nor lead to additional, irrelevant implications) and effective (allow for a fast and unambiguously interpretation of the underlying items) [11].

All said characteristics of the GUI had to be invented before they could be used. The emergence of the GUI began in the early 60s of the last century [5]. Ivan Sutherland [12] developed the *Sketchpad*[2] (see Figure 2.1) as part of his PhD thesis at the Massachusetts Institute of Technology. The sketchpad enabled the user to manipulate geometrical objects with a light pen (see Figure2.1). Only in 1980, the full text of the dissertation was available in form of a book. Even seventeen years later, it was released as a outstanding dissertation in computer science, a circumstance that shows the huge importance of Sutherlands dissertation [13].

In 1968 Douglas Engelbart defined the basic blueprint of the modern graphical user interface [4]. Engelbart created several key components, each of which became a big contributors in the development of GUIs. The first was the so called *bitmapping*. It describes the circumstance, that each pixel on a computer screen is assigned to one bit of the computer's memory. If a pixel is lit up, the value of the bit is one otherwise zero. Engelbart's invention also included the principle of *direct manipulation* and *windows* as a form of data presentation. To carry out the direct manipulation of files and of the windows on screen, Engelbart also invented a special input device. The computer mouse, formally a Tangible User Interface (which will be explained in 2.1.2). The visual feedback of the pointer gave the user a new form of immediacy and direct experience.

In 1981 launched the so called Xerox STAR [6]. The research for this computer device already started in the 1970s in the Xerox Palo Alto Research Center (PARC). Xerox's

[2] See Alan Kay's video Doing with Images Makes Symbols.

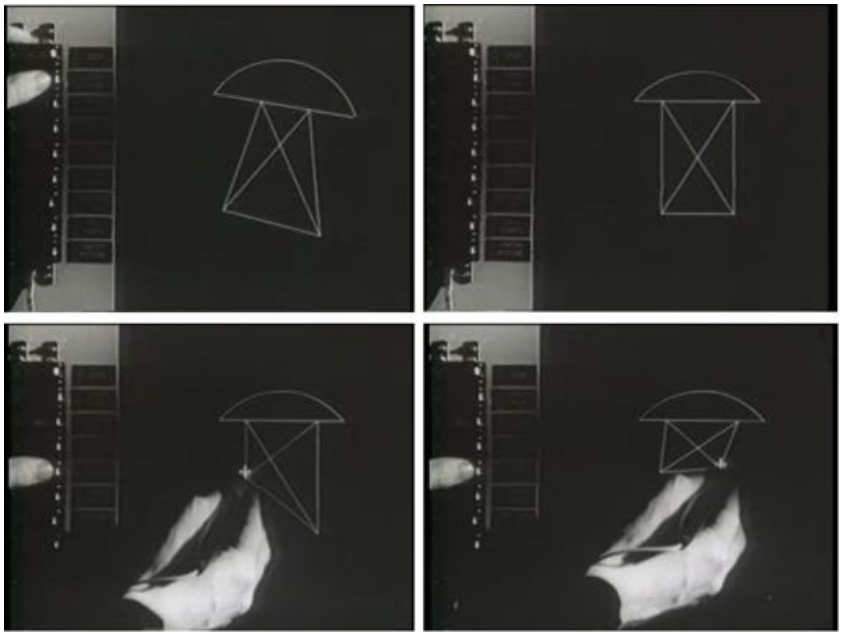


Figure 2.1: Manipulating objects with the sketchpad (from top left to bottom right)

computer system was the first commercially released computer system with a GUI. It had windows, icons, menus and a pointing device (WIMP). It implemented essential concepts of desktop computing which guided the following 20 years of GUI development [3]. The success of the STAR derived from Douglas Engelbart. A number of scientists in the Xerox PARC signed an agreement with Engelbart to use Engelbarts inventions [14]. Among them were the already explained concepts of bitmapping, windows (and therefore direct manipulation) and the computer mouse [4] . But the scientists struggled with Engelbart's implementation of windows. The solution to their problems was to regard the screen as a desk and each window as a file on said desk. The desktop metaphor was born.

Despite all effort, Xerox couldn't sell the STAR in a profitable range [5]. Another company called Apple quickly picked up the concept, released the Macintosh in 1984 and promptly brought the first successful system for the mass-market. The new interface style quickly became the industrie's standard and other companies, like Microsoft, adapted to the new style.

Nowadays the graphical user interface has reached many more devices than the desktop computer. It has found his place on mobile devices, web interfaces, consoles, games, TVs, cars and many more.

The following advantages show why this success pervades every branch of technology [5]:

- For humans, the interaction with GUIs is more natural than with other user interfaces before (humans are visual creatures).

- Symbols get recognized faster than text, therefore the interaction becomes faster and more efficient.
- The visual information level helps the user to remember things more easily which results in faster learning.
- Visual representations, metaphors etc. foster concrete thinking.
- The visual channel can provide visual context.
- Fewer Errors than with older UIs where every interaction (command) has to known by heart.
- Give immediate (mostly visual and auditory) feedback.
- All of the above advantages result in faster use and problem solving.
- Low anxiety concerning use because of the nowadays daily interaction with GUIs.

Nevertheless, graphical user interfaces (can) also have some drawbacks [5]:

- A large design complexity is the result of many different application possibilities and the consideration of all input and output channels of humans.
- There exist inconsistencies in GUIs regarding technique and terminology. This results in different efficiency concerning use.
- GUIs can be more inefficient for expert users than for non-experts, as it is complex to design the interface for both target groups.
- If the interface and its components (e.g. symbolic representations) is not specifically designed for the target group a user is part of, there are high chances of clutter and confusion.
- Because of the many symbolic representations, the Icons must be tested thoroughly.
- There exists a lack of experimentally-derived design guidelines.

The last disadvantage must be explained in more detail. It criticizes, that todays studies in the context of user interface usability are rarely published. This can be traced back to several factors. First, builders of GUIs won't publish their study results because they want to maintain a competitive advantage. Second, the studies are often specific to a specified task and can therefore not be generally applicable. Third, it takes time and effort to publish something and finally, it is also difficult to carry out evaluating studies because of the constantly increasing GUI complexity.

However, this disadvantage does not apply to all sectors in which GUIs exist. One example is the smartphone sector. Two companies, supplying two widely used operating

systems, have released guidelines[3,4] for developers and designers which work with their operating system. Guidelines itself exist since the 1970s and try to promote *good* design. But the implementation of these guidelines don't automatically result in a good interface [15]. It is always dependent on each individual use case in which the interface must be adapted to. So the success of a GUI is more dependent on the designers of the GUI, who in best case are skilled in UI design and/or evaluation. It also relies on their shoulder what happens, when two guidelines suggest different designs. Guidelines will be described in more detail in Section 3.4.2.

2.1.2 Tangible User Interfaces

As in the previous subsection, the structure of this section starts with the definition of it's topic followed by a short historical overview of it's emergence. After that, the advantages and disadvantages of TUIs will be shown.

Ishii [16] defines tangible user interfaces (TUIs) as interfaces which take advantage of haptic interaction, which is very different from graphical user interfaces. The main idea of TUIs is to give digital information a physical form which represent their digital counterparts, in contrast to the visual representation of GUIs (see Subsection 2.1.1). Those representations are directly manipulable and perceptible with our hands and body.

Tangible user interfaces are usually used for a specific application with explicit physical representations, while GUIs serve as a general purpose interface [17]. The TUI's application areas can be very extensive. TUIs can be used for programming, learning, problem solving and planning, information visualization, social communication, entertainment and music and performance [18]. However, the goal of any installation in these areas is usually so different that there are almost no guidelines, and if so, only in the pedagogical field [19, 20, 21]. In addition to traditional interfaces that provide visual and auditory information, tangible user interfaces generate mechanical signals that stimulate human kinesthetic and touch channels [17]. Haptic interfaces also provide humans with the means to act on their environment. The definition of different forms of feedback can be seen in section 3.2.

Feedback can come from different types of TUIs. Ullmer et al. [22] differ between several types of TUIs:

- *Interactive Surfaces*: Tangible objects are being placed and manipulated on planar surfaces (mostly tables). Either the arrangement of objects and/or their relations (e.g., the order of placement) can be interpreted by the installation.
- *Constructive Assembly*: Modular elements are, like physical construction kits, connectible with each other. Either the spatial relation to each other as well as the order of the actions can be interpreted by the system.

[3] See Google's design guidelines.
[4] See the iOS Human Interface Guidelines.

- *Token and Constraint*: Systems of this category combine two types of physical objects. Constraints provide structure (stacks, slots, racks) and limit the positioning and movement of tokens mechanically. They can also assist the user by providing tactile guidance.

TUIs are not always clearly assigned to exactly one of these classifications. Tokens, for example, may act as a constraint on other tokens, and constraints may operate within other constraints [18].

The first tangible user interface was created by Doug Engelbart (mentioned in subsection 2.1.1) in 1968. The computer mouse (see Figure 2.2) is the physical representation of the mouse pointer and enables direct manipulation. The advantage of the computer mouse, its predecessors and competing devices has been proven through many evaluations in the history of HCI [6]. Engelbart's invention changed the face of human computer interaction[5].

Figure 2.2: Engelbart's first prototype of the computer mouse

In the last ten years, the upcome of smartphones and their increasing popularity has let the costs on sensors, microcontrollers, and other electrotechnical objects drop immensely. This allowed products being build with those parts to be sold at a reasonable price [23]. The increased storage capacity, sensing abilities, processing power and connectivity also contributed to the distribution of TUIs as well as new knowledge, on the psychological and social level, led to areas where TUIs could be used effectively.

The reasons for their effectiveness lies in its several advantages:

- The interaction with digital content happens in a natural and intuitive way [23, 24].

[5] See Nicholas Gerbis article on howstuffworks.com.

- TUIs use humans natural ability to understand and manipulate physical forms [16].
- They support the understanding of abstract concepts [20].
- In contrast to GUIs, there exist TUI input methods, where no continuous eye contact with the interface is needed to know the current state of the program [24].
- Technology can be added to objects and environments that users are already used to.
- TUIs allow multiple users to collaborate, communicate, and connect on a with each other [24, 25].

But as with GUIs, TUIs also not only have advantages:

- There is the possibility for a negative learning outcome through misinterpreted physical representations [25].
- TUIs are less portable than GUIs [24].
- Because of the constant mechanical stress through the physical interaction with tangible objects, the possibility for mechanical failures are much higher than with other UIs.
- The efficiency of each TUI must be evaluated very thoroughly. There are only a few guidelines available. None for most of the application domains.
- The development costs are higher than with GUIs [26].
- Because of the type of interaction, low acceptance from users can be a result.

2.2 Musical User Interfaces

The decision not to categorize musical user interfaces, often also simply called musical interfaces, as an own type of user interface, is based on the fact that musical interfaces can be composed of several types of interfaces. Examples for this occurrence will be shown in the following section. This section will continue with the definition of musical user interfaces, followed by an analysis of the advantages and disadvantages of GUIs and respectively TUIs in the context of MUIs.

A clear definition of MUIs could not be found, but the purpose of these interfaces is obvious. They are used to modify or create music through a user-computer interface. Based on the related projects from Section 2.3, MUIs differ in several properties:

- Implementation type: The way the music interface is implemented offers two options. Either existing instruments are being augmented by technological approaches

(examples would be Xiao's [27] *Andantino* and Sello's [28] *Hexenkessel*) or a new form of user interface is being created (like in Jorda's [29] *Reactable* and Smus' [30] *Ubiquitous Drums*). According to Ishii's definition of tangible user interfaces, augmented instruments can be considered as such. New user interfaces can be further subdivided. Distinction can be made between tabletop [29], block [31] and toy [32] interfaces. It must be noted, that this subdivision is in direct relation with the interaction type, which means that choosing one implementation type (e.g. tabletop) can exclude certain interaction types (e.g. wearable).

As already mentioned in Subsection 2.1.2, another division between, *interactive surface*, *constructive assembly* and *token and constraint* has been declared by Ullmer et al[22].

- Interaction type: Either the interaction with the musical-interface is carried out with the fingers and hands or with the whole body. Examples for first interaction type are Pattens *Audiopad* [33] and Newtons *Block Jam* [31], for the full body interaction Vans *Music Jacket* [34].

- Sound generation: There are two possibilities how sound can be created through musical interfaces. The first possibility is the creation through the software of the MUI (digital output). Examples would be Smus *Ubiquitous Drums* [30] and Beckings *Drum-Dance-Music-Machine* [35]. The second possibility is analog sound creation like in the *Haptone* [36] and the already mentioned *Andantino* [27]. Analog sound creation is being used when an existing instrument is augmented via technology and the sound gets created analogy. One example can be the sounding box of a guitar.

- Motivation: The aim of the project also differs from project to project. Some want to give the user a new way of creating music [33, 31, 27], others want to support the user in learning the instrument faster or more efficient [34, 37]. The motivation of MUIs can differ in many ways.

- Target group: The target group is strongly connected with the motivation behind the project. There are interfaces specifically designed for children [32, 26], musicians [34], non-musically trained people [38, 39] or both [29].

- Collaborativity: Some projects are designed for group and collaborative contexts [31, 29, 28], others for one user only [33, 40].

Another possible definition approach is to compare the advantages and disadvantages of GUIs and TUIs in order to establish whether the advantages of one eliminate the disadvantages of the other or reinforce them. On the whole, the change in pros and cons of both interface types is kept in balance when being combined. The following only refers to said changes, advantages or disadvantages that have remained the same are not dealt with.

Beginning with GUIs, further improvements can be expected regarding the information level transmitted to the user. The (most likely) already existing visual and auditive feedback can be further enhanced with haptic feedback. An example for the usage of this comes from the computer game sector, where input controllers inform the user about in-game incidents with vibratory feedback. This additional feedback can also have positive effects on the user's ability to recall certain information. Where visual information already helps the user to remember things more easily, physical objects can support the memory process even more. A negative impact on the advantages of GUIs when combining with TUIs arises considering the low anxiety of usage, which is due to the lack of confrontation and use in everyday life. The disadvantage of possible symbolic confusion expands on the tangible input device(s), where symbols can also be used (on buttons, etc.). This stands in direct combination with the possible misinterpretation of physical forms of TUIs. Other disadvantages of TUIs stay the same when combined with a GUI, other than the advantages. A positive impact from GUIs on TUIs can happen when manipulating physical forms. Additional visual guidance can speed up and clarify the information interpretation by the user. The same applies on the understanding of abstract concepts. In contrast, the advantage of not needing continuous eye contact with a screen is most likely to disappear when those two interface types get combined. But even here, exceptions are possible. A MUI in Virtual Reality for example uses a TUI as an input device and a screen for visual representation. Since the screen is mounted directly in front of the eyes using a head mounted device, direct eye contact with the TUI does not need to (or cannot) take place. The advantage of modifying environments and objects, which the user is already used to, is not as simple as with GUIs, as the visual impact and change on the environment is significantly greater when using a screen. However, due to the wide range of possible combinations of GUIs and TUIs, some exceptions come to light sooner or later.

This sheer range of possibilities can be seen at a research platform for musical user interfaces called the International Conference on *New Interfaces for Musical Expression*[6] (NIME), which was founded in 2001. With the establishment of the NIME, research into new musical interfaces has now a global community and platform focusing on creating new and improving existing musical interfaces. Apart from the NIME are other, smaller music researching communities like the *International Music Computer Association*[7] (ICMA) and the *Sound and Music Computing Network*[8] (SMC), with the NIME being the only community explicitly focusing on interfaces. The NIME community develop in fields such as HCI, design theory and feedback for users [41].

The next subsection will represent projects which are related to this thesis as they are a mixture of GUIs and TUIs for a musical use. Most of them are also linked to the NIME.

[6] http://www.nime.org/
[7] http://www.computermusic.org/
[8] http://www.smcnetwork.org/

2.3 Related Projects

The following examples will show projects with different approaches regarding musical interfaces. The wide range of different forms of implementation should show the possibilities of musical user interfaces from which the project of this work may benefit from. Because of the amount of examples, each project will be presented in brevity.

2.3.1 Audiopad

The *Audiopad* [33] combines the advantages of the modularity of a rotary knob controller and the two-dimensional character of a (tracking) surface. The Audiopad is controlled by so-called *pucks*. Through their manipulation, the Audiopad provides visual feedback on the surface. For each puck, certain functions (e.g. instruments or a microphone) may be assigned by the user. Position changes of the pucks are detected and recorded by the system. This tracking information is displayed visually to the user on the table surface and is characterized acoustically by the sound reproduction of the system. By manipulating the pucks, the sound is individually changed. This combination of physical input and visual feedback provides a high degree of flexibility. While its development (in an iterative design circle), the Audiopad has been evaluated with user participation and redesign according to the results. According to Pattern J. et al, the Audiopad has to be tested while live performances in the future.

Figure 2.3: The UI of the AudioPad in use [33]

2.3.2 Audio D-Touch

The *Audio D-Touch* [40] consists of three tangible interfaces. Each interface contains a set of blocks and a disk on which the blocks are moved upon (see Figure2.4). The blocks embody sounds that can produce musical sequences through different vertical arrangements and represent the most diverse tones in different note lengths. The flat plate conveys the score lines. By arranging the blocks in different ways a (audible) melody is being produced. Their vertical position influences the volume, whereas the horizontal line determines the play time. Multiple effects can occur at the same time. The Audio

D-Touch can be used in different contexts, ranging from performance art, playing to education and composing.

A prototype of each instrument was tested by a group of people with different musical backgrounds. A specific number of participants was not mentioned by the authors. Each participant enjoyed the interaction with the instruments and was able to create interesting and varied compositions. The research results showed that the lack of visual feedback can lead to confusion among users, especially those with no musical background knowledge. The participants also noticed a number of advantages that resulted from not using a GUI and therefore not having to stare at a computer monitor.

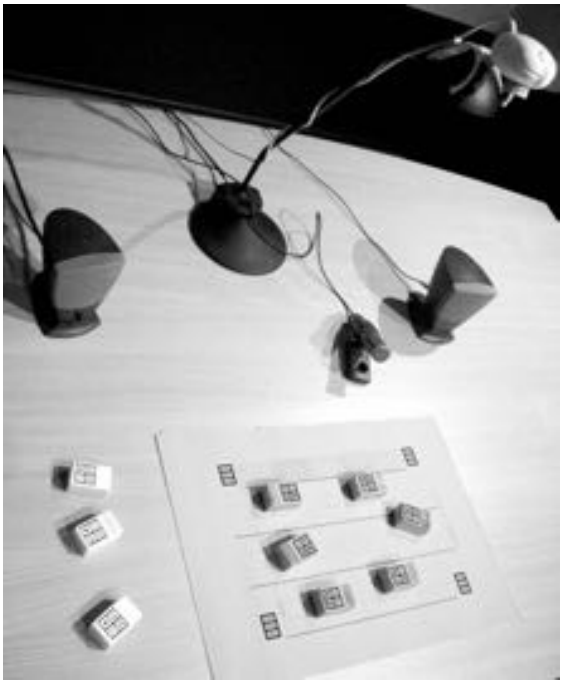


Figure 2.4: Entire set of one of the three interfaces [33]

2.3.3 BeatBearing

Peter Bennetts *BeatBearing* [42] is an interface that allows the user to create rhythms by arranging ball bearings on a grid (see Figure 2.5). A red line goes from left to right across the interface of the BeatBearing, disappearing as soon as it reaches the right side of the interface. Immediately after it appears on the opposite side moving back to the right edge.

If no balls have been placed on the grid, no sound are being generated. A dark gray circle can be seen under each hole. When the user places a ball on the field, the dark gray circle just below the ball turns white. This indicates that the ball is now activated. When the red line passes the ball, a larger colored ring around the ball is triggered, moving away

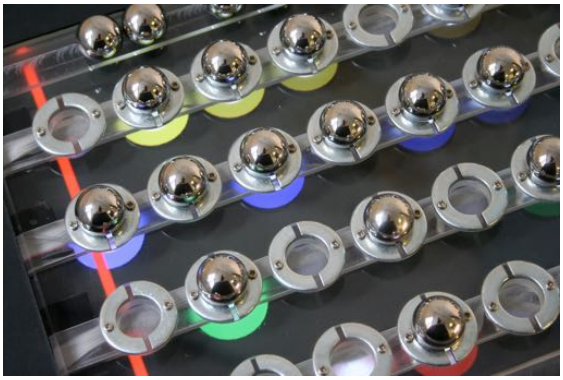


Figure 2.5: The interface ot the BeatBearing with metal in position [42]

from the ball several centimeters before disappearing. Each row has a different color that indicates a different drum sound: kick drum, snare drum, high-hat and cowbell. No user study or evaluation has been carried out in the scope of this project.

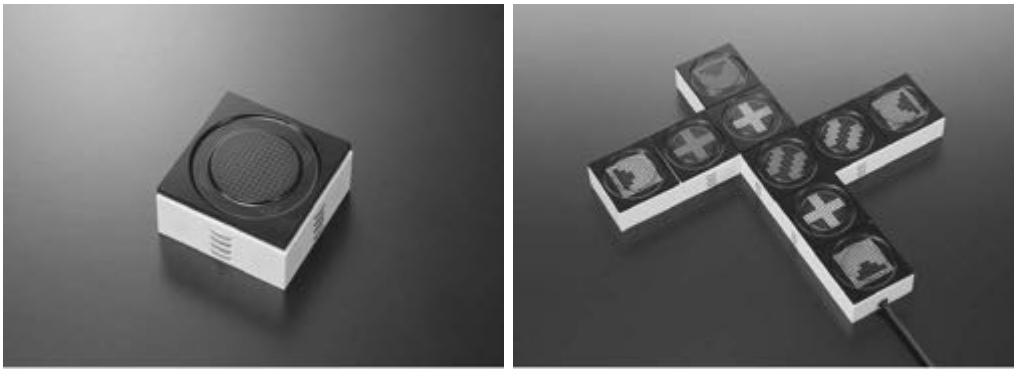
2.3.4 Block Jam

Block Jam [31] uses, as the name suggests, blocks for its interface (see Figure 2.6a). It is designed as a collaborative interface for music. Each block contains a set of possible musical sequences. These can be selected manually by pressing the top of the blocks, also containing a low-pixel display. If the blocks are connected by the side-mounted connectors (see Figure 2.6b), a dynamic structure of musical phrases and sequences emerges, depending on the status of the block. According to the evaluation of the authors, the Block Jam interface succeeds as musical user interface. After a short period of getting to know the interface, the participants were almost unable to move away from building pieces of music. The results in the collaborative area as well as scalability of the difficulty were also satisfying.

2.3.5 Marble Track Music Sequencers for Children

This project by Fischer and Lau [32] is a marble track that acts as a music sequencer for music education for children. Marbles roll along the track and trigger sound events by passing little synthesizers. Those synthesizers can be attached at every position on the marble track and therefore play different sound sequences. This MUI was designed to be used under the supervision of a teacher.

Two user tests were conducted with one music teacher and two children for each test. The children of the first test were five and six years old, the two of the second test eight and nine. The task of both tests was to explain the concept of chords to the pupils and to assemble the song "Happy Birthday". Each test took about fifty minutes. All students were able to reproduce chord-based music on the piano at the end of the tests. Older children helping younger children have also been observed in the course of said tests.



(a) The final block design with connectors on its side [31]

(b) A cluster of connected blocks [31]

Figure 2.6: Block Jam: single and connected blocks.

2.3.6 MusicCube

The *MusicCube* [43] (see Figure 2.7) is a cube object with a button as a speaker that can be pushed and turned. The surface of the cube consists for the most part of rubber. Four of the six sides of the cube represent play lists, which the user can assign an individual color to. To activate the respective play list, the user must place the cube on a flat surface with the cube side of the desired play list pointing upwards. Additional interactions are possible. To skip or rewind, the user presses the cube. The more he pushes, the faster he moves through the play list. By shaking the cube, a randomly chosen song will be played, and so on. Regarding the feedback, visual feedback was built in the MusicCube, through the LEDs of the cube, and auditory feedback, using a female voice. The LEDs of the cube indicate the volume of the music, shuffle mode, current play list (through static light) and current music rhythm (blinking in the same rhythm). The auditory feedback gives information about the current play list, song title, function and volume level.

A study was conducted to compare the interface of the MusicCube with the UI of an Apple iPod. Seventeen participants (five women and twelve men) aged between 20 and 44 hat to carry out 8 different tasks to do so. The user study concluded, that users seemed to appreciate the hedonic (e.g. interest and excitement) value of the TUI, even if it was perhaps too complex in expression. Because of this, a balance should be sought between ergonomic (e.g. support and control) and hedonic qualities to enhance overall attractiveness.

2.3.7 MusicJacket

To practice a music instrument correctly is as important as to practice often. Otherwise, false knowledge or wrong movements are internalized in the brain. At a very basic level, an electronic metronome can be an example. The possibilities of aid by technology can also be extended to the haptic level. The *MusicJacket* [34] is an example for support on

Figure 2.7: The MusicCube Prototype (right) and an Apple iPod (left) [43]

this level. MusicJacket is a wearable MUI to support the teaching of a correct posture and bowing technique to novice violin players. The system uses motion capture to track the posture of the player in real time and gives feedback through seven vibration motors which are integrated in the jacket worn be the player (see Figure2.8).

The authors evaluate their invention by comparing two groups of novice violin players. The first group receives conventional posture teaching whereas the second group was trained using vibrotactile feedback. Van der Linden et al. found out, that the feedback being given is effective at improving novices straight bowing technique even if when they no longer received vibrotactile feedback. None of the participants from the first group showed a comparable improvement.

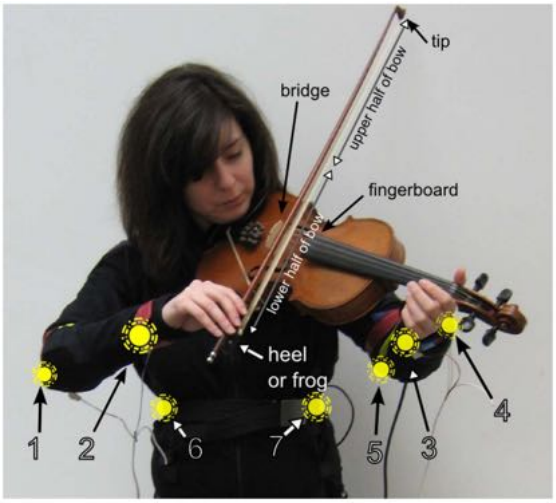


Figure 2.8: The seven vibration motors of the MusicJacket and their positions [34]

2.3.8 reacTable

The *reacTable* [29] is a popular TUI for composing electronic music collaboratively by placing blocks on a round table that also works as a display. The blocks (or pucks) can be interlinked by arranging them in close proximity to each others (see fig 2.9). The resulting sound can be manipulated by adding an unlimited number of new blocks, rearranging the connections and by touching gestures all around a block. The pucks have their own syntax, indicating their functionality. The functionalities consist of audio generators, audio filters, mixers, control filters and global objects (which affect the behavior of all objects within their influence range). Collaborative approaches also include the participation of musicians playing with (analog) instruments. To do so, the reacTable takes the audio input of said acoustic players into account. Apart from local multi-user collaboration, remote collaboration with more than one reacTable, being connected via the internet, is also possible. The reacTable, has been designed for casual users as well as for professionals. Before designing the user interface of the reacTable, Jordà et al. defined a view design rules for the interface. Any textual or numerical information should be avoided, as well as any decorative display. As well as any line, form, animation or shape must have a strictly relevant and informational purpose.

The main feedback the user gets while interacting with the system is visual feedback. Of course the performer(s) also hear their created music, but this auditory feedback is also being presented in a visual form. Those waveforms between the pucks intuitively help to understand the interface, enabling the simultaneously interaction with a high number of parameters.

Since its first presentation in 2005, the reacTable has been exhibited many times on several festivals, conferences and shows. In these contexts several thousand people of all ages and backgrounds have interacted with the system. The feedback has been very positive and enthusiastic, even if not all users fully understood the principle behind the reacTable.

Because of the positive feedback on those exhibitions, the reacTable has been further developed in different forms (mobile app, etc.). Starting in 2009 and its the commercialization is being carried by the spin-off company Reactable Systems[9,10].

2.3.9 Loop

Loop is a prototype created by the authors of this thesis and another member of the course "Projektorientierte Recherche" held at the the Vienna University of Technology. Loop consists of a user interface and a guitar embedded in a mechanical construction (see Figure2.10). The installation is aimed primarily at non-musically trained people and consists of a tabletop, color markings and wooden blocks which make it possible to change the recorded fret, and thus tone height, of every guitar string. Each of the six

[9] See the Music Technology Group's website of the Pompeu Fabra University
[10] Reactable Systems

Figure 2.9: The reacTable interface.

colors stands for one of the strings, each wooden cylinder for a beat and the angle of each cylinder provides information about the played fret. In course of the prototype two of the originally aimed at six string mechanisms have been realized, meaning two string could be played by the user. One loop consists of four 4/4 beats which repeat itself indefinitely. The input of the user is implemented by a system of motors and solenoids on the guitar.

The system was presented at an exhibition at the TU Vienna and during the course itself. The two biggest criticisms were the low visual feedback of the interface and the fact that non-musically trained people could not automatically produce harmonic music through the alternative control of a guitar presented by the UI.

Also, it was found that the guitar body alone was insufficient to enhance the tone in environments with many people present. The sound had to be amplified electronically.

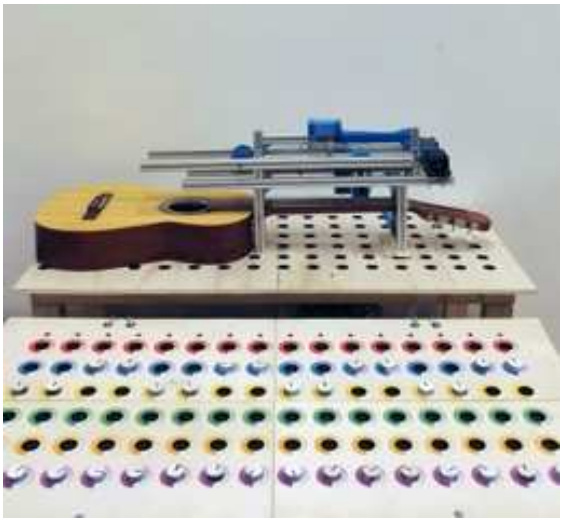


Figure 2.10: Loops UI and guitar construction.

CHAPTER 3

User Interface Design

> In designing new systems and applications, we are not simply providing better tools for working with objects in a previously existing world. We are creating new worlds.
>
> Winograd, 1997, p.153

Author: Jakob Blattner

User interfaces are very important to users. For many people, the interface is not only their window to view the capabilities of the system, it *is* the system [5].

The topic of user interface design is a highly discussed subject in media informatics. One of the reasons for this is that UI Design contains, or can contain, many different fields of knowledge. Psychology, anatomy, game design, assistive technologies and virtual and augmented reality are just a few of those that can be mentioned. Because of this complexity and diversity, UI Design is credited with a separate chapter in this work. The first section contains the the definition and description of *Usability* and *Utility*, both important for the evaluation of an interfaces quality. The second section is deeply connected with the first section and describes different forms of feedback which can be implemented in user interfaces. The third section explains different kinds of design knowledge, which most of them can be seen as basic knowledge for UI design. The last section deals with the acquisition of data to improve interfaces with their help. Unlike the penultimate section, this is data that often can not be owned prior to the design process.

3.1 Usability and Utility

The appearance, layout and navigation of a system can affect people in many ways, but doesn't necessarily result in a positive outcome for the user and other parties involved. Bad design can worsen the interaction between humans and the application in use. People will have greater difficulties doing their jobs if the interface is poorly designed and will also make more mistakes. Bad interfaces can scare the user permanently away from the system or even lead to aggression, frustration, and increased stress. A critical system, such as used in air traffic control, can cause harm to many people if badly implemented. All diverse design-guidelines (already mentioned in section 2.1.1), regulations and principles based on user groups, the psychology of humans, and other characteristics try to avoid negative and support positive outcomes. Positive effects include, for example, productivity advantages. Galitz [5] refers to an actual system in use, which needs to process 4.8 million screens per year. A poor user interface forced the users to spend one extra second per screen. However, due to the high number of screens to be calculated, this value, which initially sounds very small, adds up to almost one additional person-year per calendar-year. This shows, that seemingly small changes can have a big impact.

Possible outcomes of good user interface design can be described through the term *usability*. According to Nielsen [44] usability assesses how easy user interfaces are to use. Usability does not consist of only one evaluable value. Rather, it's composed of several attributes:

- *Learnability*: The software should be easy to learn so that the user can work with it quickly.
- *Efficiency*: Once the user has learned to handle the system, he should be able to achieve high productivity.
- *Memorability*: The system should have a high recognition value, so that the user can quickly work efficiently with it after some abstinence.
- *Errors*: The system should have a low error rate. Thus, users make fewer mistakes when using the software, which makes them hard to recover. A high error rate has a negative impact on satisfaction.
- *Satisfaction*: The system should be pleasant to use.

Another possible positive effect is the so-called *utility*. Utility and it's related word *usefulness* underly no clear definition and were changed throughout the years of research in HCI [45]. In the course of this work, the authors will adapt Nielsens definition of utility and usefulness. According to him, utility describes whether the software in use provides all the features a user needs. Usefulness describes the combination of usability and utility [44].

Usability often is also mentioned in the context with the term *user experience* (UX). UX defines the involvement of a person in a technology, a product or a human-made object [46]. This does not mean that UX only involves the person's interaction with the object itself. Possible touching points can be the objects website, the store, customer service, everything related to the object. However, good usability is a key factor for a positive user experience when interacting directly with the object itself. This relationship between usability and UX makes two terms intertwined.

The following section deals with the subject of feedback in UI design, an interface's property, which use decides between good or bad usability and/or utility.

3.2 Feedback

Author: Raphael Kamper

Designing a tangible user interface, also using visualizations and producing an auditory output feedback is crucial. Mapping, as already mentioned in Subsection 2.1.1, plays an important role, otherwise the users may lose orientation. Mixing different forms of feedback can also be tricky[47]. This present section deals with haptic, visual and auditive feedback in music instruments, excluding other forms such as gustatory or olfactory feedback. In the context of music instruments auditive feedback can be subdivided into primary feedback (such as a metronome supporting the user), passive feedback (noises made for sound generation), and secondary feedback (the actual produced sound) such as [48]. Discussing feedback Sergi Jordà states:

> "To avoid frustrations, a system does not necessarily have to be completely understandable, but it has to be coherent and responsible." [49, p. 5]

3.2.1 Haptic Feedback

James J. Gibson summarizes the "[...] responses, or self-produced stimulation, or proprioception in general." [50, p. 478] as feedback. In his work *Observations on Active Touch* he pointed out the differences of touching (or tactile scanning) and being touched. Braille displays for instance are devices providing haptic feedback and the users perform tactile scanning.

A special form of haptic feedback is vibrotactile feedback. This is often implemented in game controllers, but is not an active touch. People can also recognize different forms or patterns in vibrotactile stimuli [51].

3.2.2 Visual Feedback

There are three different levels for processing visual stimuli in the brain: low-level, intermediate- level and high-level [52]. At the low-level there is distinction taking place between visual attributes (contrast, orientation, color and movement). At the

intermediate-level of processing includes detecting surface properties, global contours (here the law of good continuation takes effect) and, what is really import, the discrimination of foreground and background. At the last stage, the high-level processing, object recognition takes place. All in all visual perception is a highly complex neurological process, including multiple areas of the cerebral cortex and parallel ongoing processes. After an object is perceived as one, it could be matched with past memories and derive meaningfulness from it.

This meaningfulness plays a crucial role in the concept of affordances, also mentioned later in Subsection 3.4.1, and is often misunderstood in the context of graphical user interfaces according to Don Norman [53]. Graphical objects do not afford clicking or such forms of interaction. They provide visual feedback and within the last stage of visual stimuli processing this feedback is interpreted. That is of course dependent on each users individual past experience and logical as well as cultural constraints.

Jeronimo Barbosa et al.[54] proposes the use of visual feedback to achieve more intuitiveness, for example by using visual metaphors as described in Daniel Arfib's et al. work *Expressiveness and digital musical instrument design* [55].

3.2.3 Auditive Feedback

Auditive Feedback is not persistent and, as does music, only exists throughout time but also vanishes with it. Jeppe Veirum Larsen et al.[48] conducted a user study exploring the effects of delayed auditory feedback on non-musically trained people. Their results show, that a delayed feedback, challenged the untrained participants in synchronizing the strumming of a guitar with a metronome, whereas it hardly influenced musically trained participants.

3.3 The Design Process

The design process is closely linked with the development process, since the design has to adapt to the technical conditions and the algorithms behind the user interface to be created (with the exception of the core functionalities) are based on the UI. The development process covers all the stages of software from its inception with requirements definition through to fielding and maintenance [56]. To say that one can exist without the other is simply wrong. Therefore this section covers a small amount of software development thematics but mainly focuses on user interface design.

According to Richter and Flückiger [57], the core software development tasks can be divide into five different phases:

- *Analysis*: The analysis serves to understand the users and the context in which the new system is used. The first step in the development process is to analyze the anticipated users and usage. Nielsen [44] writes that *Know the user* is a fundamental usability guideline. In this context, the term user describes not only the program

executing person, but all people who are influenced by the software. The results of the following methods are described and documented either in the form of graphical representations or natural language.

- *Modeling*: Modeling a whole system requires modeling from different perspectives. Different designs have to be created and their results, in turn resulting in new designs, made. For example, the people in charge model the way the users work, the functionality, and behavior of the system.
- *Specification*: As soon as there is sufficient clarity regarding the intended program, the system is specified for development. There is a fluid transition between the methods of modeling and the specification of the new solution. Results of this phase are specification documents.
- *Realization*: Once the solution has been specified, a software architecture must be designed and implemented. This step is most frequently addressed by software development models.
- *Evaluation*: The goal of the evaluation is to find problems so that they can be resolved in the course of development. The finished system, subcomponents or prototypes can be used for evaluation.

The sequence and number of occurrences of these phases changed in the more thane 60 years of (software) development [58]. From the 50s of the last century onward, software was developed lineary, which means each stage only occurs once in the development and is dependent on the stage being carried out before [59]. The most known example is the so called *waterfall model* which was used in the 1970s. In the following years the desire for an iterative and adaptable development process increased. The basic idea of iterative development is to incrementally develop a system so that the development team can leverage what was learned during the development of previous incremental system versions. The learning happens, in the optimal case, both in the development as well as from the system use. It starts with a simple implementation of a subset of the requests and then gradually improves the system until the whole system is completed. Each iteration makes design changes and adds new features. The result of this desire was the so called *spiral model*. From 1990 on, the user became the focus of discussion and attention and as the development cycle changed, so did the design cycle.

This iterative development/ design approach ensured, that the user could be integrated into the design process [60]. The developer does the full benefit of the user when being involved in and after every phase of development respectively design. In the analysis the users can help in defining the requirements for the system or by allowing themselves to be observed and giving feedback about the problems of the current system (if existing). They can also take part in interviews, surveys or questionnaires. During modeling, specification and realization the users can test mockups, prototypes and provide feedback. Rosenzweig [46] realizes this approach in an iterative design cycle (see Figure 3.1). It comes to

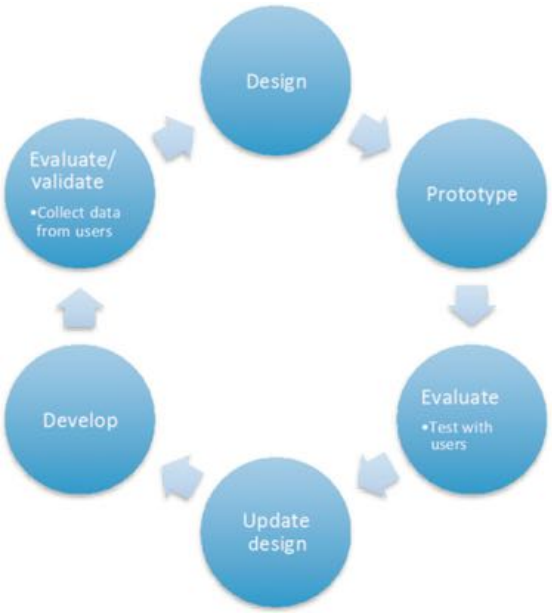


Figure 3.1: Rosenzweigs iterative design cycle. [46]

attention, that there are two stages of designing and of evaluation in the design cycle. This is because prototypes consume less resources in development than finished products. The stages of Rosenzweigs design cycle can be can be categorized based on the fact that two types of knowledge [60] are needed for UI design. The stages *Design*, *Prototype*, *Update Design* and *Development* can be identified as stages where user interface design knowledge (see Section 3.4) is applied whereas *Evaluate data* (with the notation *collect data from users*) and *Evaluate* uses knowledge obtained from information gathering activities (see Section 3.5).

Rosenzweigs iterative design cycle is only one of many possible approaches for designing a user interface. Some are more specific and list the overall sequences of stages to take in the design process (in form of process models), others reflect a general design attitude with a defined goal as part of a project. The, for this project, relevant process models are usable in a user centered design (UCD) context. UCD is a design approach where the user and knowledge about the user is the designers central concern [60]. This includes knowledge about the users ability, need, tasks and environment within which they will use the product. To achieve this, potential users of the product being created are actively and constantly involved in the iterative design process to give information and feedback about the product. This also helps establishing the requirements and design of the product in the early stage of developing.

In the following subsection of this work, different UCD process models will be explained. These process models are then analyzed in the later part of this work (see Chapter 6.1) on the properties of this project, whereupon a model based on the analysis results is

selected for the practical part of the thesis.

3.3.1 Process Models

Starting with process models, the ISO Standard 9241-210 for *user-centered-design of interactive systems* (see Figure 3.2) defines four steps in the design process. In the first step the designers have to understand the underlying context of use, followed by specifying user requirements. The process now iterates between those two and the following two stages, where a design solution is being produced and evaluated, until a solution has met the desired requirements. This solution represents the completion of the design cycle. As already stated at Rosenzweigs design cycle: different evaluation methods will be discussed in subsection 3.5.2.

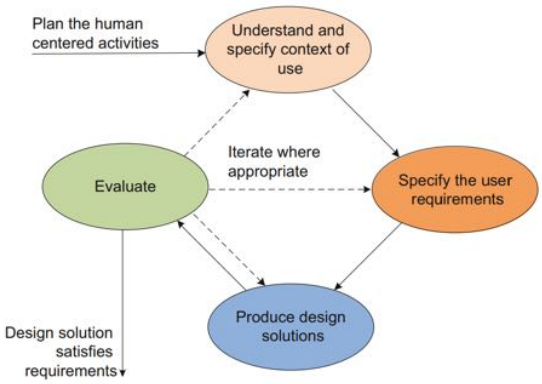


Figure 3.2: The key process steps of the ISO Standard 9241-210 [57]

Another iterative design cycle is the *Star Life Cycle* by Hix and Hartson[61]. The central point of the star is the evaluation. The evaluation is relevant at all stages in the life cycle and not just at the end of the development process [60]. This also means, that the evaluation always has to be carried out between the current and the following phase. Errors are thus contained and high user participation is a must. A certain starting point is also missing in this cycle. Where other processes start with the requirements specification, which is also present in the star, the Star Life Cycle does not. This is because Hix and Hartson intended their life cycle to be equally supportive of both top-down and bottom-up development, additionally to inside-out and outside-in development [61]. One of the drawbacks of this approach is, that project managers often have problems with the highly iterative nature of this life cycle. They find it difficult to decide when a particular iteration is complete, thereby complicating resource management and control over the overall progress of the development process. A solution to this problem is to establish control mechanisms in form of quantitative usability goals that act as stopping rules.

A more specific approach comes from Doborah J. Mayhew by the name *Usability Life Cycle* [63] and includes not only requirements analysis, design and testing techniques, but also organizational and managerial strategies. Mayhew emphasizes that the represented

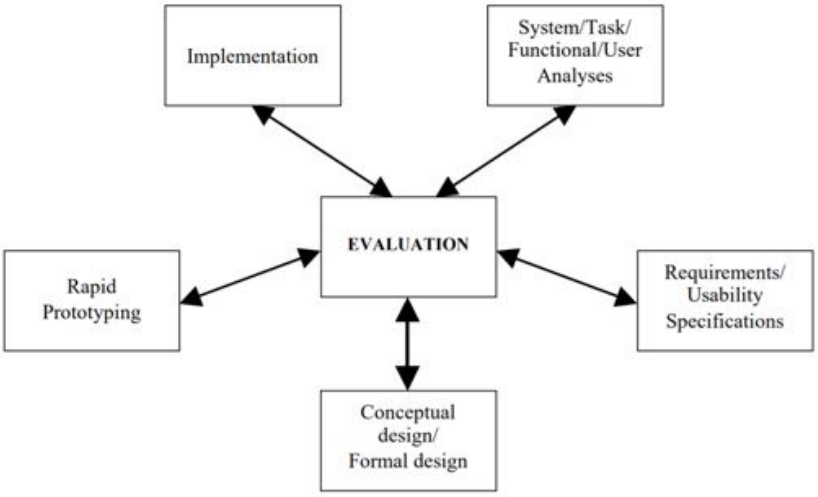


Figure 3.3: The Star Life Cycle by Hix and Hartson [62]

methods can be applied throughout the whole development process. Looking at the life cycle, it is noticeable that it contains many methods and phases that are, in terms of their order, clearly related. Mayhew represents the design cycle from start to finish as a whole. Because of the resulting size, the cycle itself is not going to be discussed, but can be viewed in Figure 3.4. It should be noted, however, that a complicated process such as designing an application may not cover all the circumstances encountered. Full coverage, as Mayhew represents, is thus never entirely possible. Mayhew reported another problems while applying the process in various projects. The established development processes of an organization can not simply be translated into human-centered processes in a single project. In addition, development teams often lack the knowledge to perform UCD activities. But nevertheless, Mayhew emphasizes the importance of all project members carrying out, or at least helping to carry out, UCD methods, as this strengthens the understanding of the users and the underlying context.

The next approach specifies on one certain step in the development process, the design. Alan Cooper [64] states, that *Goal-Directed Design* (GDD) combines methods from different areas and a set of interaction principles and patterns to provide solutions that meet the user's needs and goals. Cooper describes, that the design phase itself is part of an iterative development cycle. He commits five major changes to traditional software development methods in its design process:

- *Design first, Program second*: In the old days, as Cooper stated, the programming began as soon as possible and the design got applied at the end of the work. This should be turned around, designing the products completely before any programming has been done.
- *Separate responsibilities*: The responsibility for the design and the responsibility

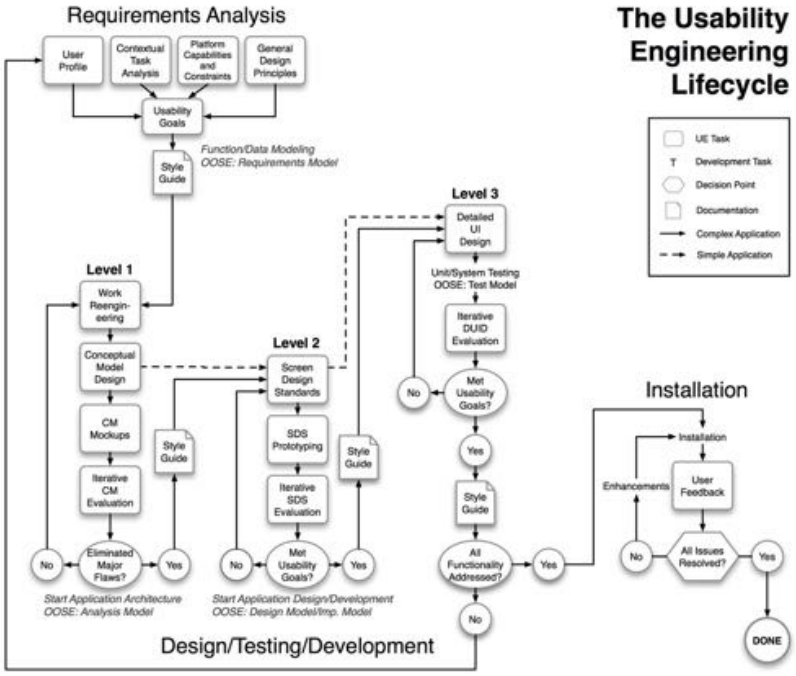


Figure 3.4: The Usability Engineering Life Cycle by Deborah J. Mayhew [63]

for the programming should be separated. Divided responsibilities ensure that significant changes are not made in the middle of production by programmers who want the program to be easy to code. These changes often lead to a different interaction with the product by the user and change the main goal of the designer, the ease of use of the product.

- *Responsibility for user satisfaction*: The responsibility for user satisfaction results from how the product behaves and what it looks like. This results in a clear responsibility for user satisfaction for designers, something the project management has to be clear with.

- *The importance of personas*: Managers and programmers used to talk about the user without being specific. The use of personas, a composite portrait of an idealized user on a sheet of paper, prevents that. Goals derived from these created personas represent the foundation of Coopers process. These goals direct all design decisions and a different to tasks. Cooper defines a goal as an end condition, whereas a task is a process needed to achieve the goal.

- *Work in teams of two*: Two people must be assigned to all project teams: a designer and a design communicator. The designer is responsible for the product design, the design communicator for the description of the product.

Cooper outlines three major benefits: improved product quality, reduced development time and improved documentation. The GDD approach itself can be roughly divided into six phases: *Research*, *Modeling*, *Requirements*, *Framework*, *Refinement* and *Support*, all being part of the design stage before the project's actual implementation. The research phase uses ethnographic field studies to provide qualitative data. During the second phase, behavior and workflow patterns are being discovered through analysis of the first phase. The results are personas and diagram models of different kinds. The third phase consists of the requirements definition in which the connection between user, models, and the framework of the design. In the framework definition phase, the entire product concept, such as product behavior, visual design and physical shape (if applicable) is created by the designers. Similar to the framing phase, the designers are taking care of the design in the refining phase, but with more detail. The last phase is about supporting development. Even a sophisticated design solution raises questions or challenges. The designers should be available for any kind of uncertainty regarding the design to be implemented.

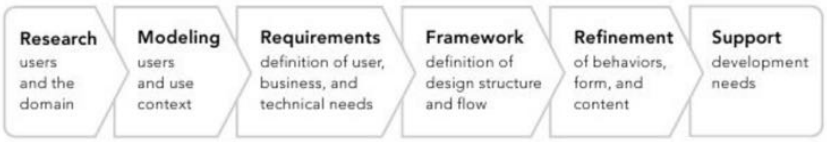


Figure 3.5: The Goal Directed Design Approach by Cooper et al. [64]

To summarize the discussed approaches above, user-centered design rests on three principles: the early focus on user involvement, using measurable usability criteria for evaluation and an iterative design cycle.

3.3.2 Design Attitudes

After these process models, the remainder of this section is devoted to (for this thesis interesting) design attitudes and their goals. The first general approach being discussed is *Participatory Design*. Participatory design is the direct involvement of people in the collaborative design of products they use [65]. In contrast to UCD (explained in the beginning of Section 3.3), the user is not only included in the evaluation by different methods, but can actively participate in the design process. This users involvement should bring more accurate information about tasks. However, extensive user involvement might be costly and may lengthen the implementation period. Additionally, following the users decisions in the design process might not always be the correct thing to do, as Nielsen [44] stated. It must be added, that Nilson also claimed the complete complete opposite. He thereby questioned the degree of user involvement in the design process and which decisions should be left to the user. A decision which is left to every design by him-/herself. It is also important for the design process which users are being selected to contribute to a successful collaboration [65].

Another approach is the integration of *playfulness* into an application. An interface which enables a playful approach invites users to engage in social and physical interaction because it leaves a positive emotion: fun [66]. Barbosa et al. [54] consider playfulness as a state in which people can get. Playfulness can also be used to bring people in a so called *flow*. Flow is a condition in which people become totally immersed in performing an activity due to a perfect balance between the challenges offered by the tasks and the skills possessed by the individual. This implicates, that having a broad target group with different skill levels present, the designers must also implement scalable difficulty and/or to achieve this flow for as many users as possible. It is characterized by a high degree of enjoyment and involvement by the user. If the activity is too easy, the user feels bored, if it's too demanding, the user feels anxious. Barbosa defines four properties, which influence and create the flow state:

- *Control*: the sense of control users feel over the process and the activity's outcome
- *Attention focus*: to what extent users were distracted or absorbed while performing the activity
- *Curiosity*: the degree of imagination and curiosity stimulated while performing the activity
- *Intrinsic interest*: to what extent users were voluntarily engaged and motivated by the activity

Robson [67] reviewed five sound toys for non-musically trained people. His conclusion was, that first-time users respond enthusiastically to the TUIs which used a playful approach. They also place themselves willingly in performance-like situations. Unlike musicians, naive users respond better to the playful qualities of the objects. New interfaces that have no clear rules of usage greatly lower the user's risk of appearing incompetent. Areas of application are in education, to achieve behavior change, for health and rehabilitation but also just for fun [66]. A playful approach reduces the time to achieve the wanted results. Applicable locations can be found everywhere, no matter if at home, in the office, in transport or in public environments.

After having highlighted different properties of the design process, the following section is dedicated to knowledge which build the basis of designing and implementing solutions.

3.4 Design Knowledge

Design knowledge must be obtained by the designers before participating in a design process. Early in the work process the designers should fixate a set of guidelines, principles and theories for the interface to come. They record decisions for all parties to see, facilitates the automation of design and promote consistency and completeness. The product to be developed is largely based on this knowledge. When ignoring or forgetting

them, an unnecessarily amount of resources, like time and fundings, have to be used retrospectively.

The design knowledge explained in this section is subdivided into two further subcategories below. Starting with design principles, containing knowledge based psychological perception and other broad applicable areas. The second subsection contains design guidelines, which form knowledge and procedures for certain scenarios.

3.4.1 Principles

Principles cover fundamentals of UI design such as coping with user skill levels and preventing user errors [65]. They are widely applicable but can be broadly interpreted. The following examples represent an important knowledge base in user interface design.

A very fundamental aspect of humans is their psychological perception. This perception has been summarized in form of four groups by Stone et al. [60]. The first group of their *Psychological Principles* is about the fact that "users see what they expect to see". Two principles have been grouped, the *principle of consistency* being the first. According to it, users find it difficult to handle the unexpected circumstances. That's why it is important to implement consistent design. Similar conditions should use similar actions. This applies to color, fonts, styles, layouts and so on. If red indicates danger, then red should always be chosen to do so in other contexts rather than e.g. green. The other principle is the *principle of exploiting prior knowledge*. If users already have experiences with e.g. calculators, using a calculator icon lets them apply their prior knowledge and experience of a physical calculator to the program. The designers can be build on the users prior knowledge, for example with screen metaphors (see Section 2.1.1).

Users often need to divide their attention between different tasks at the same time. Multiple tasks can not be done parallel, as why only one task can be the center of attention. The second group is about how to channel the attention of users on one task. It consists of two principles. The *principle of perceptual organization* says, that grouping things which belong together helps the user. The attention can now be easier payed to the grouped elements than each individual element. One example for this are tables. The grouping in form of rows and columns allows better visual filtering and comparing values. The second principle, the *principle of importancey* says that the moment something is important for the user, it should be placed in a prominent position. Warning messages or pop-ups are examples for this principle.

The third group of principles is devoted to the visual structure of the UI. The *Gestalt psychology* defines several principles on how humans perceive the world. user interface design adheres to these following principles:

- *Principle of proximity*: Elements that are close together appear as groups rather than single or random elements (see Figure 3.6a).

- *Principle of similarity*: Elements of the same shape or color appear to belong together (see Figure 3.6b).
- *Principle of closure*: Whenever the brain perceives only a part of an element, it automatically tries to create a complete construct. (see Figure 3.6c).
- *Principle of continuity*: This principle states, that humans tend to continue elements beyond their ending point (see Figure 3.6d).
- *Principle of symmetry*: Humans tend to perceive elements bounded by symmetrical borders as coherent figures (see Figure 3.6e).
- *Figure-ground segregation*. When observing figures with two ore more distinct areas, the human mind recognizes one of these areas as the background and other areas as part of an object. The background area gets ignored and the user only sees the object areas (see Figure 3.7).

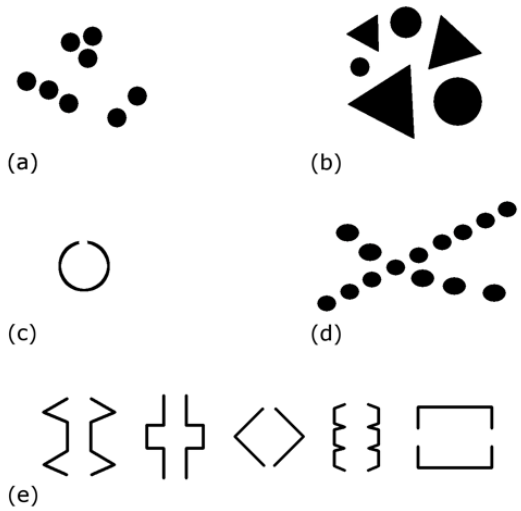


Figure 3.6: The five Gestalt principles: (a) proximity, (b) similarity, (c) closure, (d) continuity and (e) symmetry [60, p.94]

The last group of psychological principles deals with that for humans, it is easier to recognize information than to recall it. Recalling describes the information retrieval from our long-term memory, whereas recognition is the ability to recognize information or events due to familiarity of previously experienced encounters[1].

Shneiderman et al.[65] extend the four groups of psychological principles with their *golden rules of interface design* which are applicable in most systems. One of their principles overlaps with the already explained *principle of consistency*, which is why it has been

[1]See Raluca Budiu's article on memory recognition and recall in user interfaces.

(a) (b)

Figure 3.7: The Figure Ground Segregation. Under some circumstances, a solid object can be perceived as background, causing the actual background to appear as the foreground [68]

left out. They highlight that their list of rules is (and cannot) be complete but has been derived from three decades of experience and covers many aspects of UI design.

1. *Cater to universal usability*: Different user groups in the target group can differ among themselves because of different age, skill level or disabilities. The task of a designer is to enable all user groups with different design features such as tutorials for novices and shortcuts for experts.

2. *Offer information feedback*: Every user action should return some feedback to the user. It is also important, that the user associates the feedback with the correct task the feedback is intended for. The intensity of feedback should also correlate with the user's input. There are different ways of giving the user feedback which has been explained in Section 3.2.

3. *Design dialogs to yield closure*: Sequences of action from the user should be organized into groups with a beginning, middle and end. Giving informative feedback to users at the end of a sequence results in a feeling of satisfaction and accomplishment. An example would be the information that a transaction has been completed.

4. *Prevent errors*: The interface should be designed, so that users can't make any errors. Examples are grayed out menu items and not allowing alphabetic values in a numeric input field. If the user makes an error, the interface should detect the error and offer constructive and non-frustrating solutions. To stay with the previous example, the user should not have to retype the whole input if it contains an error but rather only have to replace the faulty part.

5. *Permit easy reversal of actions*: User actions should be reversible as often as possible. This relieves anxiety because the user knows his action is not irreversible.

6. *Support internal locus of control*: Experienced users want to feel, that they are in charge of the interface and it responds to their actions. Changes or surprises in familiar behaviour is undesired and results in frustration and annoyment.

7. *Reduce short-term memory load*: Humans cannot remember many information comfortably in the short-term memory. The consequence for UIs is, that designers should not create interfaces where they rely on the user to remember information from one screen to the other.

Shneidermans feedback principle is also covered in the *three principles from experience* [60], indicating, that feedback is a very important feature in user interfaces. The other two principles are the *principle of visibility* and the *principle of affordance*. The first principle describes that a control should imply through its design what it is used for, whereas the second principle states that it should be obvious how the control is being used. Those two principles are very important for tangible user interfaces, since affordance overlaps more from the graphical to the physical level than in graphical user interfaces.

The last principle presented in this subsection has been found in the progress of literature research for related tangible user interfaces. Bennett [42] wrote in his paper about the BeatBearing (see Subsection 2.3):

> "One particular design principle that has influenced the development of the BeatBearing is the *principle of containment*. This principle aims to create interfaces that appear (though are not necessarily constructed) in a self-contained manner. This is in opposition to interfaces that are 'split-up', where a user is presented with a separate physical control surface, a screen, and an audio source all in different physical locations. The aim of containment is to encourage interacting with the device in a Heideggerian 'ready-to-hand' manner, and increase the possibility of the user experiencing flow."

After covering the basic psychological principles of UI design, the following subsection highlights guidelines which are, in contrast to the knowledge of this subsection, specifically created for a certain design context.

3.4.2 Guidelines

Guidelines specify different, widely ranged design scenarios and how to behave when a certain scenario appears [65]. They promote consistency among multiple designers in terms of usage and appearance, containing examples and counter-examples from practical experiments and empirical studies. Guidelines are a topic of discussions between designers,

being criticized from some for being too specific. Simply put, they consist of rules for designing user interfaces [57].

Guidelines can be categorized according to their purpose. The following list displays the difference in guidelines, from where they come from and where they apply.

- *Legal Stipulations*: Most of the time those regulations ensure safety of workers in direct contact with the devices in question. An example is the directive 90/270/EEC[2] from the European Union, which deals with safety and health requirements for display screen equipment.
- *Standards*: On national and international level, standards target is to make the use of technology simpler and easier for users. The ISO9241 [3] being one example, defining requirements for ergonomics in human-system interaction.
- *Collections of rules*: They are intended to optimize the development of UIs. Examples are general usability principles, like Nielsen's *Usability Heuristics* [44], as well as specific rules for certain fields like mobile and web applications.
- *User interface patterns*: UI patterns describe recurring or similar design problems and offer proven solutions.
- *Vendor or platform style guides*: A document which describes the prescribed look and feel of an application. Two examples are Apple's *iOS Design Themes*[4] and Google's *Material Design*[5].
- *Corporate style guides*: Rules for corporate design and look-and-feel to match the different applications of the corporation.
- *Project style guides*: Guidelines for ensuring the consistency of the user interface during development.

The following three guidelines by Barbosa et al. [54] serve as examples and are in direct contact with *playfulness* and playful interface design (see Section 3.3.2). These guidelines are used in the specific case of an installation playing music in an infinite-loop, which is being controlled from non-musically trained people.

- *Advanced Looping Capacities:* Looping Capacities define the range of musical possibilities provided by the music installation. It consists of the basic set of functionalities like record, overdub, play, stop, delete and the advanced functionalities. These advanced functionalities go beyond the this basic set, expanding musical

[2] Directive 90/270/EEC
[3] ISO 9241
[4] Apple's iOS Design Theme
[5] Google's Material Design

possibilities. It also raises the curiosity of the user which might yield exploratory use, which in turn has been linked to enjoyment. Additionally, this advanced functionalities can allow the development of new skills for advanced users without compromising the basic features for novices. Therefore it can also help users to find their optimal balance between challenge and skill, which is essential for achieving the flow state (explained in Section 3.4).

- *Low input capacity and direct mappings:* Input capacity defines the number of input controls visible to the user for the interaction (knobs, buttons, screens etc.). Thus low input capacity means minimizing the number of said standard input controls to achieve an easier start for the user by reducing confusion and coupling the reduced number of input controls with the usage of direct mapping. Direct mapping stands for an direct accessible functionality when the user use an input control (e.g. pressing a foot switch to record). This could (again) help users with different skill levels to find the balance between challenge and skills. With direct mapping users could also spend less time searching the user interface for input possibilities. This allows them to better focus their attention on the musical activity.

- *Transparent and intense visual feedback:* Visual feedback should reflect what is going on inside the device, which means it should provide transparent feedback to the user. The visual feedback intensity can range from low (e.g. a single small LED) to high (e.g. a full monitor screen). The importance, properties and implementation possibilities of visual feedback will be discussed in subsection 3.2.2.

After covering the basic design knowledge for this thesis, the following section talks about knowledge which can not be completely obtained before the start of a design process.

3.5 Gathering Information

In contrast to design knowledge, which is already acquired prior to the interaction with the user, this knowledge is mainly user based and must be gathered throughout the design process. This information is then used to determine the needs of users and the quality of the interface, which are different for each project.

This section covers all components which occur in the process of gathering information in a design process. The first subsection contains the definition of the two different types of data which can be gathered. The following subsection explains different possibilities of acquiring this forms of data whereas the last subsection explains the way of summarizing, managing and distinguishing the gathered information.

3.5.1 Qualitative and Quantitative Data

Qualitative and quantitative are the two types of collectable data in the course of an evaluation (explained in the following subsection). Qualitative data is data from users or

other stakeholders that include a rich verbal description [69]. In contrast to qualitative data, quantitative data consists of data which is numeric and can be measured in standard units. Neither qualitative nor quantitative data alone can answer research questions completely alone. The combination of both promises a successful design. Numeric data alone can't bring insight about not measurable behavior and can't sum up knowledge (for example) from specialists gathered in form of an interview. Having no quantitative data, which e.g. gives insights about eye movement on the screen, mouse clicks per second or time between mouse clicks of the user, also misses out valuable information. When considering which methods to use thinking about the final outcome of the data can help. If the designer wants to present how many times a user did something, this would suggest a quantitative approach. Quantitative methods are generally done later in the life cycle and generally collect a larger amount of data [46]. A qualitative approach on the other hand provides a rich description of how the user responded [69]. This approach is often done in the early stages of development or before a new release (if a previous application version already exists) [46].

Methods that provide one of the two types of data are summarized under the term *User Research Methods*. The URMs being used in the context of this work will be explained in chapter 7. Methods like interviews, focus groups, ethnographic research and thinking aloud generate qualitative data. Examples for quantitative generating URMs are surveys, questionnaires and tasks with bio sensing (e.g. heart rate monitoring or eye tracking). After analyzing and evaluating (see section 3.5.2) the gathered data, the outcome results in new requirements and thus new or updated prototypes.

The next subsection treats the process of gathering the two different types of data explained in this subsection of the thesis.

3.5.2 Usability Evaluation

Usability evaluation can be done in two different ways, expert testing and user testing. The evaluation process tries to evaluate the, in section 3.1 covered, attributes of usability: learnability, efficiency, memorability, the number of errors and user satisfaction.

In expert testing, professionals use selected rules to examine a user interface specifically for typical usability problems [70]. An expert finds statistically 25 percent of all problems, three experts already 70 percent. Compared to user testing, expert testing is faster, but only finds potential problems. An example for selected rules comes from Nielsen [71]. It should be noted that most of these overlap with the design principles discussed in subsection 3.4.1 which is why a more detailed explanation is not needed.

- Visibility of system status
- Match between system and real world
- User control and freedom

- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize, diagnose and recover from errors
- Help and documentation

The before mentioned user involved testing is also called usability testing and is of empirical nature. The basic idea of usability testing is to evaluate a product or system by having a real person interact with it and test it as such. Apart from the difference in the gained data (see 3.5.1), there also exist different test designs and test environments. There also arises the question for the ideal number of participating users.

A sufficient number of participants for qualitative methods is, according to Nielsen [44], already achieved with five users, which find most of the existing usability problems in the tested product. Running different tasks in course of the testing, the number of five people per tasks is enough [46]. Of course five people don't guarantee to find all usability problems, this is why a number between five and eight people is suitable for all qualitative methods. In order to get statistically relevant data when carrying out quantitative methods, a number of at least 20 users must participate in the chosen method. In principle, however, applies: the more the better. These tests are often carried out online (e.g. surveys) to gain a big number of people from a big pool of possible participants. Tests like these can also be carried out personally but takes comparatively more effort and time than online.

There are different design possibilities from which empirical tests can choose from. First of all, the user test can be *task-based* or *open ended*. Task-based usability tests try to simulate a certain use case. It determines if the product to be tested is usable in real-life situations where the task provides the user with a structure he or she can walk through. Open ended usability testing on the other hand lets the user explore the product by him-/ herself. This is because there may be too many tasks to choose from or the designers want to know if there are any use cases or problems they oversaw. In both design scenarios *thinking aloud* can prove very useful. Because of the fact, that one can not simply see inside the user's brain when using the product, the user talks what goes to his/her mind when using said product. Thinking aloud is often hard to do for the user while performing the task and also distracts the participant, but has the advantage of expressing thoughts that someone may not remember in hindsight. The user's interaction can, independent from thinking aloud, also be audio and/ or video recorded. This gives the advantage of discussing the interaction in hindsight. In addition to the design options already listed,

tests may or may not be moderated. Non-moderated tests yield the advantage of a more realistic use case scenario where no one helps the user but moderated tests, where the moderating person is interacting with the user, also has its advantages. The moderator can set the participant at ease when nervous, keep the test session in time, can make sure that the technology is working and ask probing and clarifying questions.

The used technology stands in direct connection with the environment in which the tests are being conducted. When the internet can not be used as a platform to carry out tests, a different test environment has to be defined. The test environment can be a laboratory, the future field of use or any other space such as a conference room or an office [72]. The benefits of testing in a lab are, that the environment can be designed to create the ideal testing environment, ensures working equipment, and saves effort finding a location for testing, rounding up the equipment etc. Drawbacks are the costs incurred and that the user might act differently than in the actual environment in which the product would be used. Field testing lets the designer and tester learn more about the actual context of use, therefor gains richer data, and is also cheaper. Disadvantages are the not controllable test environment (e.g. technical equipment, different people are present, etc.), and it's more time consuming because of the additional efforts like traveling, setting up the equipment etc. Remote testing allows a bigger number of participants for an extended time period, informations about the gathered data must be manually added by the test users or gone through in retrospective with the designers. Again, there are disadvantages such as the context in which the data was collected, or the forgetting or unwillingness to manually add information to the data. In the end there are the following resources which conclude the test environment: money, space, time, the kind of application and it's result use of context and how close to this context the data should be collected.

After collecting data with the help of the described methods, the results have to be formed into documents and recordings which can be used in the future course of the design process. The product of this formation are requirements.

3.5.3 Requirements

The diversity of human abilities, motivations, personalities, backgrounds, cultures and work styles are all factors in UIs. Understanding the intellectual, physical and personality differences between users and addressing those needs, is a designer's goal [65]. The needs encountered by the users and other stakeholders are recorded in the form of requirements. This suggests that requirements come in different forms. Wiegers [73] confirms this assumption in his work. According to him, requirements can be divided into five categories, a division according to which this thesis orients itself in its further course:

- *Business Requirements* typically come from the lender, the acquiring customer, the actual users, or a product visionary. Business requirements describe why the customer implements the system.

- *System Requirements* describe the top-level requirements for a product. A system may be all software or it may include both software and hardware subsystems. Humans are also part of a system, so certain system functions can be assigned to humans.

- *User Requirements* define goals or tasks that the users must be able to perform with the product. Therefore, user requirements describe what the user will be able to do with the system. Examples to represent user requirements include use cases (see Subsection 7.2.2), user stories, personas (see Subsection 7.1.3) and scenario descriptions. Gathering all user requirements, or at least trying to do so, is the main reason for iterative design cycles. The reason behind this is, that simply not all user requirements can be gathered at the beginning of a project [60]. To do so, a further understanding and definition of requirements must be done in the design process. A circumstance on which UCD is based upon.

- *Functional Requirements* describe the functionality of the software being implemented by the developers to enable users to accomplish their tasks and thereby satisfying the business requirements [73]. These requirements are *shall* statements, e.g.: *The system shall e-mail a reservation confirmation to the user.*

- *Non-functional Requirements* include performance goals and descriptions of so called *quality attributes*. Quality attributes describe the product's properties that are important to either users or developers. These features include ease of use, robustness, efficiency, integrity and portability. There are also other non-functional requirements which describe design and implementation limitations. Constraints limit the choices available to the designer for design and construction of the product.

Requirements are a major key to success and the foundation of any development activity [65]. A study by the Standish Group shows that incomplete requirements and lack of user involvement are the two most common reasons why software projects fail [57]. The importance of identifying, gathering requirements and the understanding among all the stakeholder is therefore very high. Without adequate requirement definition, the developers are not sure what problems they are solving and when they are done. Another possible mistake in requirements is the mutual exclusion in requirements. This and other mistakes can be prevented by *Requirements Engineering*. Requirements engineering can summed up be as a method which strives to ensure that the needs of users, customers, and other stakeholders are handled, managed, and communicated so that the project team can create an appropriate solution. A more detailed description of Requirements Engineering can be found in Section 7.2.

CHAPTER 4

Machine Musicianship

Author: Raphael Kamper

Gil Weinberg and Scott Driscoll [74] first defined the term *robotic musicianship* in 2006, developing a robotic percussionist called Haile. In this work they distinguished between *musical robotics* and *machine musicianship*, the latter including *rhythmic perceptual modeling*. The term machine musicianship was originally coined by Robert Rowe [75]. According to Bretan et al. machine musicianship:

> "[...] focuses on developing algorithms and cognitive models representative of various aspects of music perception, composition, performance, and theory." [1, p. 100]

In the following sections we provide an overview of music theory, perception and music generating algorithms including common protocols such as MIDI and OSC with a focus on the very basics.

4.1 Music Theory

The field of music theory consists of multiple subjects such as acoustics, notation, harmony and rhythm to name a few important ones. The present section provides an overview of concepts that are meaningful for this work. Notation, for instance, is an important part of music theory, but is not applied during the design process or is necessary to understand this present work better.

4.1.1 Physical Principles of Music

This subsection deals with acoustics and gives an introduction to common terms. The structure of the subsection follows Kurt Haider's work *Einführung in die Musiktheorie*[1] [76].

4.1.2 Sound

Every auditory perceivable vibration generated by an elastic body is referred to as sound. It propagates spatially in wave form, and if those vibrations reach the human ear through a medium (e. g. air) we perceive a sound in the case of regular oscillations or a noise otherwise [76]. The hearing range goes from 16 to 20.000 Hz. The shortest time interval a tone can be perceived by a human is called reaction time and is about 1/15 second. The speed of sound depends on the medium and additional parameters, but at 20° Celcsius it is 343,8 m/s. Sound pressure is specified in microbar (μb) and describes the alternating pressure produced by the air molecules during the sound propagation. The sound intensity is specified in decibel (dB) and refers to the amount of sound energy a sound wave passes per time interval in a region (W/m^2). The intensity of the amount of sound pressure decreases as square of the distance from the sound source, because the sound power spreads spatially around the sound source. Decibel is a logarithmic measure and reflects the sound level difference between two intensities of sound pressure. As a reference value the hearing threshold is defined for 0,0002 μb and 1.000 Hz. This means that the hearing threshold can be expressed as 0 dB. As loudness perception is not objectively quantifiable, the subjective perceived loudness is referred to as phon, whereas 0 phon is equal to 0 dB at 1.000 Hz. This enables the development of a phon scale with the according frequencies. This scale now shows which frequencies are perceived equally in loudness at a given sound intensity. It does not allow any conclusions which phon value is perceived twice as loud or half as loud at a given frequency. Therefore, the sone was introduced. 1 sone is equal to 40 phon, and a sound pressure at the same frequency perceived twice as loud would be 2 sone.

4.1.3 Pure and Complex Tones

Whether or not we perceive a sound event as a noise, tone or sound[2], depends on the oscillation type [76]. An irregular, unsteady, aperiodically type of oscillation is perceived as a noise, whereas a tone is defined by a regular, periodic and continuous oscillation. A pure or sinus tone does not occur naturally or by common music instruments. Usually it is produced electronically using synthesizers. A natural or musical tone consists of multiple sine waves, being partial or overtones of the basic oscillation. From a physical perspective this means that a natural tone is already a musical sound and a complex wave. There are basically two types of sound waves: transverse and longitudinal waves. Solid bodies oscillate transverse, meaning vertical to the body such as a guitar string. Liquids

[1] Engl.: An Introduction into Music Theory.
[2] In German here sound would best be referred to as *Klang*.

or gases (air) oscillates longitudinal, meaning particles move lengthwise like an air column in a wind instrument. The transverse waves transform to longitudinal waves in the air and all waves are transformed into transverse waves reaching the eardrum. Partials or harmonics are integer multiples of the basic frequency also called the fundamental, and all frequencies except the fundamental are so called overtones or upper partials [77]. Not all instruments produce those harmonic frequencies. Membranophones usually create random, and metallophones no harmonic ones. Given the fundamental and overtones and numbering them from 1 to 16, the frequency relations can be calculated in a diatonic interval [76]. Overtones lead to complications when tuning an instrument (see Subsection 4.1.5). Additionally to the underlying physical principles, there are psychoacoustic effects influencing the human perception of sounds (see Subsection 4.2).

4.1.4 Sound Parameters

There are four important parameters describing a sound: (1) pitch meaning the number of oscillations per second, in other words frequency, specified in Hertz (Hz), the lower the frequency the lower the perceived fundamental tone, (2) amplitude of the wave respective loudness, (3) duration of sound perception and (4) timbre or tone quality. Timbre depends on multiple factors, but is basically the composition of a complex musical tone through the overtones.

> "The clarinet, for example, has strong odd-numbered partials. The flute has strong even-numbered partials." [77, p. 11]

Changes in the above parameters during a tone are called envelope [77]. The same fundamental tone played on different instruments most likely results in different envelopes depending on multiple factors such as amplitude, damping or sustain amongst others. The interference of waves leads to either expansion of the amplitude or cancels each other out if the phases are shifted. If waves interfere that only have a minor frequency difference of a few Hertz, it comes to a phenomenon called beat that is usually perceived as unpleasant.

Resonance is crucial from instrument construction to concert hall architecture. The intensification of sound waves, called resonance, is reached if the wave fits convenient spatially in the enclosing space. In instrument construction resonators intensify the sound waves by using tubes, for instance, in vibraphones. In string instruments like violins or guitars, the strings are attached to solid bodies, and the vibration is intensified by the body. In architectural context the combination of resonance and reflection is referred to as reverberation, as sound is reflected by hard surfaces, and absorbed by soft or porous surfaces.

4.1.5 Temperament and Tuning

If tones are tuned regarding their overtunes only, this is called *just tuning*. Figure 4.1 shows the frequency relations for this overtone series. The frequency relations are given

by 1:2 octave, 2:3 quint, 3:4 quart etc. This leads to the problem that you can either have perfectly tuned octaves or quints, but never both, as potencies of two or three can mathematically never synchronize [76, p. 29]. The mostly accepted tuning to solve this problem is the so called *Werkmeister tuning* introduced in 1691. Werkmeister divided an octave into 12 steps, each step being a semitone step. The frequency relation of an octave is 1:2 or in other terms $1 : \sqrt[12]{2}^{12}$. This results in the calculable frequency relation that one semitone step factor is $\sqrt[12]{2}$. Such kind of interval division only works because the human brain tries to correct mathematically not perfectly matching semitone distances (see Subsection 4.2).

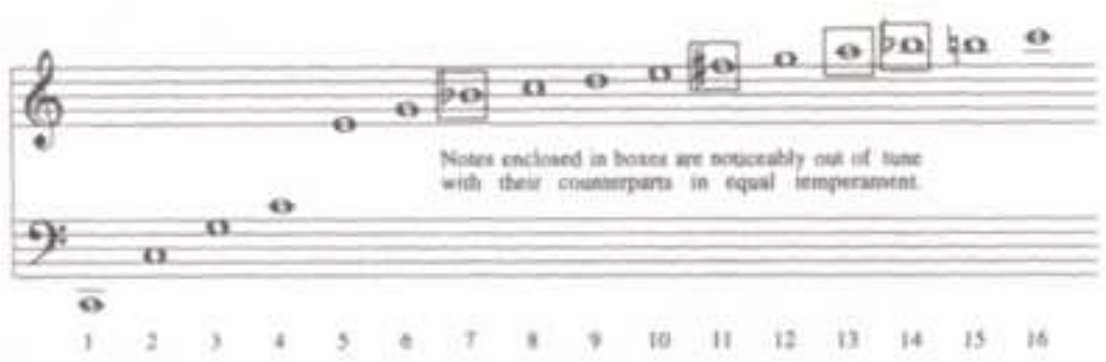


Figure 4.1: Harmonic overtone series [77, p. 11, Example 2-2].

4.1.6 Pitch Space and Harmony

Pitch space deals with, as the name states, relations of pitches in the frequency spectrum (space). There are plenty of scales developed throughout history. In machine musicianship musical scales can play a crucial role when it comes to algorithmic composition and of course in the perception of music, as different scales have developed in different places leading to socio-cultural associations with those scales. Harmony deals with the composition (chords) and relation of sounds [76]. One of the most important relations in harmony is the triad.

4.1.7 Triad

As the name suggests, triads consist of three tones or pitches in this context. The theory behind triads is not trivial and we only give a simplifying summary in this subsection. Starting from a fundamental or root, the third tone and the fifth tone make a triad. Whether or not the third tone is a major or minor third, determines what is commonly known as major or minor triad[3]. Major means a frequency relation of 5:4 and minor 6:5[4].

[3] Deutsch Dur- oder Molldreiklang.

[4] This means an interval size of three (minor) or four (major) semitones.

4.1.8 Chords

All combinations of more than three tones sounding together is called a chord, but it is not a chord's property that those tones produce a consonant harmony [77]. If the highest amount of dissonance is created, it is referred to a as dischord.

Especially in guitar (rock) music there often occurs a special form, the so called power chord[5]. It consists of three tones: root, fifth and octave. This is insofar important as a power chord could consonantly fit into a major or minor scale.

4.1.9 Scales

Musical scales define the number of whole or half tone steps on the scale [77]. The whole tone scale, for instance, consists only of whole tones, as the name states. The major and (harmonic) minor scales differ by the half tone steps from the third to fourth and ninth to tenth semitone within the octave. This also reflects in the according major or minor triads.

The pentatonic scale is considered as one of the oldest musical scales[76]. Ordering of tones is given by starting at a fundamental and going up by a quint five times, e. g. C - G - D - A - E.

4.1.10 Rhythm

As mentioned in the above section, one property of a musical tone is duration. Time (and timing) is a crucial aspect of music. The description of rhythm heavily relies on a notation system, which, as we mentioned earlier, is not part of this work. Nevertheless, we provide a short overview of important issues concerning this work. Tempo is the perception of the music speed and may be given by descriptive words such as adagio (slowly) or presto (very fast) [77], but is usually measured in beats per minute (bpm). The higher the bpm rate, the faster it is perceived and vice versa. Beat perception can occur on multiple levels. For example the Viennese waltz beat is perceived in a different form than most rock music pieces. This is because the waltz beat is perceived at the dotted-half note level and the rock music at the quarter-note level. To understand note levels we need to introduce the term meter and tactus. Tactus is a suitable chosen time window containing notes (or pauses). The tactus organization into recurring patterns is called meter and according to Owen rhythm is defined as:

> "The pattern of musical events in time. Rhythm is perceived as a continuity of sounds of longer or shorter duration. Rhythm may or may not conform to a meter." [77, p. 30]

Weinberg et al. [74, p. 29] argue that rhythmic perceptual modeling is subpart of machine musicianship. It deals with the computational modeling of high- and low-level rhythmic

[5] See the ultimate guitar wiki for further information.

percepts including detection of tempo or beat, but also more abstract and subjective perceptions such as similarity or rhythmic stability.

4.2 Music Perception

It is notable that there are subjective overtones, meaning that humans perceive overtones even if they are non existent like in a pure sine tone [76]. The problems of temperament and tuning were already presented (see Subsection 4.1.5). The reason why intervals can be slightly out of tune is that humans correct the detuning in favor of an integer frequency relation. The correction amounts to 40%[76, p. 30] of a semitone distance. This means that even though the frequency relations do not slightly differ, the closest interval is perceived if the difference is within the tolerance area.

4.2.1 Gestalt Psychology and Synaesthesia

In 1890 Christian von Ehrenfels[78] introduced the term Gestaltenqualität (engl. gestalt quality). It is based on Ernst Mach's findings from 1886, who wrote about Tongestalt [79, p.128] (engl.: tone gestalt). Von Ehrenfels drew the conclusion that the melody or Tongestalt is different from the sum of single tones. So if one reproduces a melody, it usually differs (except for people with an absolute pitch) from its original tone height. This means that she/he is not reproducing the sum of the former perception, but a complex of relations, and those relations are unique for every (memorized) Tongestalt.

Besides the relation of sensing specific stimuli resulting in an according perception, there are more complex phenomenons. When listening to music, this could lead to both auditory and visual perception of music for some people. This phenomenon of sensing or associating specific stimuli along with others is called synesthesia. According to Ramachandran et al.:

> "Synaesthesia is a curious condition in which an otherwise normal person experiences sensations in one modality when a second modality is stimulated." [80, p. 4]

Although the phenomenon is hard to verify, e. g. when people see colors for specific music tones, there are also associations people have, who do not perceive this phenomenon in a musical context. Additionally there seems to be some general associations humans link when it comes to the verbalized description of perception. In his work *Psychologische Probleme* Köhler wrote about what is nowadays known as the bouba/kiki-effect [81], but back then he was originally using the pseudowords *Maluma* and *Takete* [82]. People of all age groups (including toddlers) tend to call the left object in figure 4.2 *Maluma* and the right object *Takete*.

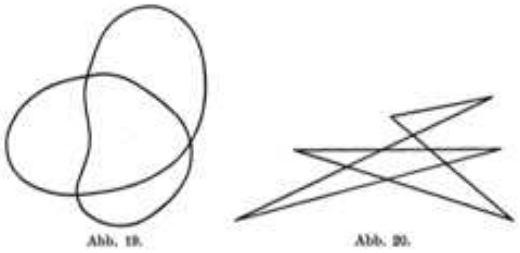


Figure 4.2: Originally used shapes by Köhler[82, p. 153]

.

4.3 Algorithms

Many algorithms are developed especially for the musical context. Sound analysis is one important field of application, but will be left out in this section. Instead, we focus on algorithmic composition. As the mathematical correlations in music are long known, humans built (programmable) machines to create music (see Chapter 5).

4.3.1 Short History of Music Generation Algorithms

Eearly approaches are mainly playback devices. This only changed within the Renaissance and Baroque when new mathematical findings encouraged rapid developments in all kinds of scientific fields and the construction of machines for demonstrative and entertaining purposes [83]. Before those epochs, by around 1000 CE, Guido of Arezzo[6] provides instructions in his work *Micrologus* to support automatic generation of melodies by text, more precise vowels in text that are mapped to different pitches for monodic and polyphonic singing. Arezzo's work can be considered as a first approach of algorithmic composition. In this context the concept of Raimunuds Lullus (about 1213 - 1316) is notable:

> "[...] the "Ars Magna" of Raimundus Lullus effectively realizes the concept of a computer (music) system. The analogies to hardware and software, data memory, program, etc. are evident in the components, definitions and rules of the "Ars Magna." Lullus creates a system that due to its underlying structure (hardware, corresponding to the diagrams), a knowledge base (data, corresponding to the definitions) as well as application instructions (software, corresponding to the interpretation rules) independently generates statements." [83, p. 24]

Based on this work Athanasius Kircher[7] developed a system of algorithmic composition in his works "Musurgia Universalis" and "Arca Musarithmica" to generate four-voice rhythmic patterns[83]. Kircher represented the pitch by numbers leading to a form of pitch classes, which is a concept used in music production and analysis in the 20th century. Additionally, he included combinatorics into his works.

[6] Guido of Arezzo (about 991 - 1033 CE) was an Italian music theorist.

[7] Kircher (1602 - 1680) tried (without success) to decipher the Egyptian hieroglyphs with a combinatorial approach[83].

Wolfgang A. Mozart's piece "Das musikalische Würfelspiel" was published (postmortem) in 1793. Musical dice games were known before, but besides the fact that this is an important approach to encourage audience participation, it is a way to let people without any former musical knowledge create music[84]. By rolling (two) dices the arrangement of previously composed samples is determined. Each of the combined sections need to fit every other section of the whole set, usually resulting in non-complex music pieces.

Lejaren Hiller and Robert Baker already developed Musicomp as the first computer-assisted composition environment in 1961 [83]. With the emergence of algorithmic composition, programming languages and frameworks supporting this task evolved such as MusicN or Computer Lisp Music. Nowadays common programming languages are Pure Data[8] or SuperCollider (see following Subsection 4.4.2).

4.3.2 Contemporary Approaches

Halley Young [85] applied a categorical grammar approach to create music[9]. This attempt differs from generative grammars[10] by creating music objects deriving from and transforming into new objects. A music object could be, for instance, rhythm or a base pitch type (that could be transposed by a function into another pitch space). The output is a MDII file containing the composition. Young suggests that while there is no universal music theory to implement in a generative music algorithms, a meta theory supporting particular styles could be achieved with the application of categorical grammars.

Shingchern D. You et al.[86] presented their work *Automatic Chord Generation System Using Basic Music Theory and Genetic Algorithm* to calculate chords for a given MIDI file's melody based on western popular music. The genetic algorithm produces a solution, a fitness block states the probability to create a following solution, in any case the algorithm stops after a maximum number of solutions. Chord composition takes about 30 minutes for a standard western pop song on common computer hardware.

Theoretically, and many of them have already been applied, many concepts and theories developed within computer science can be used for generating music such as cellular automata, neural networks, petri nets and genetic algorithms only to name a few. Despite those approaches for algorithmic composition we want to mention an easily applicable method.

4.3.3 Markov Model

The Russian mathematician Andrey Andreyevich Markov (1856 - 1922) published a study statistically describing the letters in the poem *Eugeny Onegin* by Alexander Sergeyevich Pushkin [83]. The probability that a vowel follows a consonant, a vowel another vowel etc. changes if the previous letters are taken into account. Markov models are stochastic models describing a process where each process state depends on a former state. It can be visualized, for instance, as a transition graph. Furthermore, hidden Markov models hide those internal states and give so called *emission probabilities*. In the context of algorithmic composition those probabilities

[8] See Pure Data website.

[9] She also provides the python code and audio samples on GitHub.

[10] Generative Grammars in algorithmic composition are based on Noam Chomsky's linguistc model [83] of generative grammars and are as well used for both analyzing music and composing it.

could describe tone pitches or tempo regulation. Already mentioned Lejaren Hiller used Markov Models to compose the Illiac Suite[11], which was performed by a human string quartet.

4.4 Software and Protocols

In this section we want to give a very short overview on current software used for algorithmic composition and common software protocols when processing or transmitting music.

4.4.1 Pure Data

Pure Data[12] is a software system providing both a programming language and an IDE. According to the project's website it is designed for multimedia and music live performances. Programming happens in a heavily graphic centered way by dragging and dropping elements and connecting them via lines. Input can be MIDI files, MIDI keyboards, the actual computer keyboard or microphones and the output is as diverse from MIDI files to messages on a serial port e. g. to a connected Arduino.

4.4.2 SuperCollider

SuperCollider[13] supports algorithmic composition including audio synthesis in a real-time performance also for live coding. It is designed as a basic client server architecture using the Open Sound Control (OSC) protocol and comes with its own programming language sclang. Programming can be done in a common text editor or an IDE.

4.4.3 Open Sound Control

Open sound Control (OSC) is a protocol specifically designed for computer (network) communication in a music context[14]. The message specifications are listed on the official OSC website[15]. The latest version 1.1 was introduced at the 2009 NIME conference by Adrian Freed et al. [87]. It features an URL style symbolic naming scheme like XPath, so for instance an OSC message looks like "/position/cartesian (x y z)" [87, p. 117] to describe a 3D position in the Cartesian coordinate system. OSC can be used instead of MIDI, especially if a full MIDI Implementation would be a design overkill.

4.4.4 Musical Instrument Digital Interface

The Musical Instrument Digital Interface (MIDI) is a protocol that allows electronic musical instruments and computers to connect and communicate with each other. We do not want to discuss the message structure at this point, as MIDI is not going to be implemented within this thesis' work. This protocol made it possible for synthesizers, MIDI controllers and musical tangible user interfaces (see Subsection 2.1.2) to be more portable and affordable. This allows more and more people to learn instruments and to work with MIDI controllers by practicing

[11] See audio samples on youtube.
[12] See puredata.info.
[13] See SuperCollider's GitHub website.
[14] See the OSC website.
[15] See OSC specifications.

with it at home and in the studio. Due to the wide range of possible uses, the MIDI standard is suitable in many scenarios.

CHAPTER 5

Musical Mechatronics

Author: Raphael Kamper

As stated in the above chapter Gil Weinberg and Scott Driscoll [74] defined the term robotic musicianship. They originally identified the two main parts: *machine musicianship* and *musical robots*. Nowadays the latter term is referred to as *musical mechatronics* [88, 1]. This chapter deals with the core aspect of musical mechatronis: physical sound generation. We chose to follow Barton et al. [88] classifications of robotic music, historically evolved from the mechanic and mechatronic music instruments.

Scott Barton et al. [88] differ mechanic, mechatronic and robotic music instruments in their work building a robotic zither, where mechanic and mechatronic instruments could as well be autonomous or cooperative musical machines. Cooperative musical instruments process both human and machine input signals. Autonomous means that the machine follows a set of instructions. Those instructions could include different parameters such as tempo, pitch or volume. A music box would be an autonomous mechanic music instrument. Systems such as GuitarBot [89] or Mechbass [90] are autonomous mechatronic music instruments. While the instructions and sound generation for the music box are controlled solely mechanic, the mechatronic examples have both mechanical and electronical control systems. An example for a cooperative (and autonomous) mechanic musical instrument is the pianola, developed in the late 19th century. A more recent one is the Marble Machine. The latter mechanical apparatus produces the sound, but the human player can always interfere and interact. The difference between a robotic and a mechatronic music instrument lies in the sensors providing feedback to the mechatronic system itself, usually achieved by sensors processing environmental input signals. This is one possibility of giving a rough classification of music instruments and installations. It is not a strict one and the autonomous or cooperative aspect overlaps the classification of mechanic, mechatronic and robotic music instruments.

The terms mechatronic and robotic are often used interchangeable in literature. Nevertheless, we use Barton et al. [88] classification. When only designing an autonomous mechatronic instrument, it might not necessarily play a role, whether it senses and processes input signals from its environment through sensors, but if one wants to provide interaction possibilities with the instrument, this is crucial. As we are designing an interface to interact with the instrument,

we use this form of classification. One thing all those instruments have in common is that they are considered to be autonomous. The term autonomous was used in the last millennia and ever since its meaning has changed depending on the context. All instruments, whether they are mechanic, mechatronic or robotic instruments, could also be classified as autonomous or cooperative instruments. In this environment autonomous does not refer to the power supply, but to the music generation. A barrel organ is not a fully stand-alone instrument. The player or a motor supplies the power by turning the crank handle, but cannot influence the music itself (except tempo depending on the organ's architecture). This is predetermined by the pinned barrel meaning that the music generation is done autonomously. Contrary to autonomous instruments are cooperative ones. They allow a human player a form of interaction to influence the generation of music. An example would be a machine plucking guitar strings and a guitarist fretting the chords (see Section 5.5.8 for a likewise approach). Barton et al. define it as follows:

> "A *cooperative* musical machine, an idea we introduce here, requires both human and machine input as parts of a symbiotic whole. Such a system embraces what machines do well, such as complex polyphony and temporal precision. Simultaneously, it alters the affordances available to a human performer, who may subsequently direct her attention towards timbral, articulatory and gestural nuance." [88, p. 320]

When designing a musical interface the concept of autonomous or cooperative instruments takes effect, but in this chapter we are not going into more detail as it can be considered separately from sound generation. The following sections provide examples for a better understanding of mechanic, mechatronic and robotic instruments presenting related projects and shortly discuss the problem of classifying music instruments.

5.1 Classification of Music Instruments

There are different concepts and classifications of music instruments around the world, and according to Margaret J. Kartomi the concept of instruments:

> "[...] refers to the dominant or competing views in a society of the meaning and significance of instruments as cultural phenomena, including the hierarchical ratings of instruments." [91, p. xv]

The oldest known classification system is from China[1] and dates back to 23rd century BCE [91]. It was circle based on cardinal points and seasons between the cardinal points. Four circle segments represented material such as metal or bamboo instruments. The Eurocentristic classifications of percussion, strings and wind instruments, amongst others, dates back to the fifth century CE and the Late Roman theoretician A. M. S. Boethius. Most of those schemes are visualized in a tree structure, but there are also different forms. The Hindu-Indian classification is also circle based with segments. For instance divided into hollow or solid instruments, and those segments again subdivided into solo or accompanying instruments. Javanese instruments were classified by material such as bronze, leather or wood.

An excursus on why people classify music instruments or their perceived environment at all, as well on the detailed classification of music instruments, is out of the scope of this work, but as Kartomi pointed out:

[1] The original source is unknown, but it seems to be from the time of Emperor Shun (2233 - 2188 BCE)

> "[...] the schemes that we habitually use affect the way we perceive the world and understand it. In the case of musical instruments this includes the way in which we create and respond to music itself." [91, p. 3]

Consequently, when designing a music instrument or interface it is worth keeping that in mind. In the following sections we present different music instruments and installations with a focus on their underlaying physical principles. Therefore, we ordered them by the Eurocentristic based scheme into string, percussive, keyboard and wind instruments.

5.2 Mechanic Music Instruments

The history of mechanical music instruments dates back to the ancient world[92, 93]. Heron of Alexandria (ca. 60 CE), for instance, drew sketches of a wind powered organ or a water powered trumpet player. With the rediscovery and translation of the works from ancient mathematicians and engineers in the Renaissance, besides many other mechanical machines and installations, mechanical music instruments were rebuild, and new ones were invented. Later in the 17th and 18th century was the heyday of sophisticated clockwork musical instruments[94]. In the 16th century the first barrel organs were constructed. One important effect of those mechanic music instruments is that they represent a form of recorded music and are unique, historic sources[2]. In the late 19th century player pianos were invented [95]. Similar to barrel organs they were based on punched rolls. Already in 1886 Richard Eisenmann experimented with electromagnets to induce sustain into the piano strings. Famous composer such as Johan Sebastian Bach, Wolfgang Amadeus Mozart and Joseph Hayden wrote pieces especially for musical clocks or glockenspiel [96]. Following subsections present a musical clock, the player piano and further selected mechanic instruments.

5.2.1 Musical Clocks

Musical clocks have a long history and are known since the 16th century. Until the second half of the 18th century musical clocks were very expensive and also used as a status symbol [96]. Later in the 19th century they were more common and used to entertain an audience in taverns or event locations. They played popular opera parts, dances or flute concerts, and the barrel was usually commissioned work. An underlying clockwork was moving the pinned barrel. The techniques of barrel creations were often treaded confidentially as they were a business secret. An advantage of this approach is that multiple pieces could be created on one barrel and select the according piece by shifting the barrel. The pins regulate the pitch, sequence and duration of the tones. There are multiple mechanisms depending on the concrete implementation, but a common concept is that those pins active a lever starting the actual sound generating mechanism such as striking a key, membrane or activating bellows.

5.2.2 Pianola

It is a controversial discussion, who invented the first player piano. As early as in 1825 Clementi, Collard & Company built a player piano [97] and in 1863 a french inventor, Fourneaux, created the Pianista [95], but those devices were not fully functional. However, in 1897 the American

[2] One cannot conclude that music composed and played by a human musician sounded exactly the same at the time, but some high quality assembled mechanic instruments are getting close to it [94].

engineer E. S. Voley received the patent for a pneumatic pianola [96]. Pianola was originally the trademark of an American company, but is used to describe all kinds of automatic pneumatic pianos. Punch bands or rolls, often created by hand, are used as the input for the pianola. In early versions a clockwork drove the paper roll, but this was soon attached to the pneumatic mechanism to control the speed. A pair of bellows, either operated by hand or foot, is used to move the piano hammers. With the breakthrough of pneumatic player pianos, compositions were made that could not be played by a human pianist. In 1912 a concert by the London's Symphony Orchestra was held at the Queen Hall in London featuring a pianola [96]. This concert was again held in 1913 in Paris by the Lamoureux Orchestra also featuring a Pianola.

5.2.3 Further Instruments

One of the oldest autonomous instruments is the Aeolian harp[3] [92]. Medieval sources tell about artificial singing birds moving their wings [96]. In this regard the installations of the engineer and landscape gardener Salomon de Caus in the late 16th and early 17th century are worth mentioning. He adopted and rebuild some machines originally described by Heron of Alexandria, but also invented the so called Phantastic Machina [97]. This installation consists of a water driven barrel organ and a moving statue. Sea organs[4] are also a notable example making use of the natural sea stream, weather and bypassing boats to create sound.

5.3 Mechatronic Music Instruments

As mentioned in the introduction of this chapter, the term mechatronic and robotic are used interchangeable in literature. Mechatronic instruments in this section consist of mechanisms to play an instrument, supported by electro(-magentic) components such as motors, sensors or solenoids. Already in 1880 J. Carpentier created a melograph [96] to record a played composition on any keyboard instrument and a melotrop to playback the so recorded piece. In the melograph electrically driven paper sheets are moved at constant speed and a electromagnet assigned to a key pushes down the paper onto a waltz covered with black ink. The black ink places on the so produced paper were later cut out and used as input for the melotrop.

5.3.1 The Original Encore Automatic Banjo

The Encore Automatic Banjo[5] patented in 1892 is a system augmenting a human-playable banjo [98]. The original patent describes an electric driven star wheel to pick a string and solenoids with a fixed position to manipulate the pitch according to a perforated music roll. This exact system is not known today, but it seems that electronic problems resulting in fire hazard, and noisy solenoids interfering with the actual played tone, led to a pneumatic system in 1896. The development of this installation underwent company division and multiple redesigns in a relatively short amount of time leading to a first exhibition at the Paris Exposition in 1900. The installation was designed as an entertaining slot machine playing one out of five selectable songs after inserting a coin.

[3] Aeolus was the God of wind in the greek mythology. The Aeolian harp is known since the ancient Greek world and constructed in a way that wind causes vibration of strings and therefore produces a sound [92]. The underlying physical principle is the Kármán vortex street.

[4] The sea organ created by Nikola Bašić in Zadar is the most recent constructed sea organ.

[5] See the mechanical music press website.

5.3.2 Experiments with mechanically-played Violins

In 1920 C.V. Raman [99] built an apparatus to imitate a human violin player and to conduct physical experiments. It takes long practice to learn a good bowing technique and Raman argued that from a mechanical perspective it is simpler to move the violin instead of the bow. The bow is mounted on a lath with weights attached to regulate the bowing pressure. Bowing speed is controlled by an electric motor with according tachometer. Raman conducted several experiments varying the bow position, pressure and speed, while measuring the pitch. His four main findings show correlations of (1) bowing speed and bowing position, (2) bowing speed and bowing pressure on the intensity. A position near to the bridge results in higher intensity, but also more pressure or higher velocity is needed, (3) bowing pressure and pitch are correlating non linear and (4) muting the string leads to different correlations of pressure and pitch. When constructing a bowed string instrument, those findings will be useful along with formulae Raman provided to calculate the minimum bowing pressure for a desired amplitude at a specific bowing speed.

5.3.3 Further Instruments

There are many other electronic instruments such as the first electronic piano introduced in 1898 [100] or the Banjorchestra, developed in 1914 and based on the Encore Automatic Banjo presented in the previous subsection 5.3.1.

5.4 Robotic Music Instruments

In literature music instruments are often referred to as robotic if they imitate human behavior or are designed to be perceived as humanoid. Those design approaches are however self-limiting, and therefore, we decided to use a different classification. Following Barton et al. [88] arguments, we define a robotic music instrument as an mechatronic music instruement, additionally, capable of sensing information about their environment and to interact based on the gained information, in other words (environmental) feedback.

5.4.1 Feedback in Robotic Instruments

Depending on the conclusion that could be drawn by the processed input signals Barton et al. [88] distinguish low-level and high-level feedback. Low-level feedback deals with the instrument's own state such as auto-tuning, and high-level feedback takes the whole environment into account, which could lead to improvising based on sensing sound from a human musician. For a better understanding the following subsections provide examples of low-level and high-level feedback.

5.4.2 Low-Level Feedback

Jim W. Murhpy et al. [101] discussed their approach to self-tune a guitar string. Basically there are two ways of auto tune a string. Either measuring the tension of the string (which seems to be more rare) or by sensing its vibration. The tension approach needs some additional information about the string such as length, material etc. The vibration measuring approach entails to actually play the string, tune it, and play it again to verify the result. Additional low-level feedback information would be all tone related parameters of the instrument such as duration, intensity or pitch.

5.4.3 High-Level Feedback

High-level feedback takes the musical environment, at least partly, into account to enable a proper reaction such as stop playing, improvising or imitating the sensed input. High-level feedback requires interpretation of the measured signals in a specific context, for instance, to detect a chord out of multiple single notes played at a time or rhythm classification and detection of changing tempi, but also by sensing a human player and drawing conclusions from his or her physical behavior. The marimba player presented in the following subsection 5.4.5 is an example of high-level feedback, as it is capable of playing along and improvising with a human player based on the instrument's musical output. The played music is analyzed and the robot interacts accordingly to the interpretation of the sensed information.

5.4.4 Vibraphone Robot

Pan et al. [102] developed a humanoid vibraphone player (see Figure 5.1). It is equipped with an audio and visual processing module. The audio module consists of a microphone for beat and amplitude detection. Based on the measured values, predictions are made for synchronization tasks. Volume change leads the visual module to focus on the human player. It detects nodding of the human player and the mallet. The robot reacts to volume changes and would either play a solo or together with a human player depending on the humans behavior. When the human player decreases the volume under a specific threshold, the robot starts interacting, otherwise it assumes the human plays a solo. It is limited to detect and play only monophonic sounds.

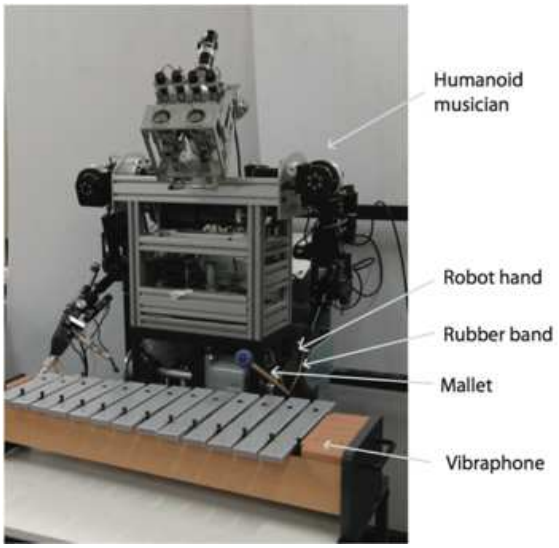


Figure 5.1: Humanoid musician [102, p. 167, Figure 2].

.

5.4.5 Shimon

The robotic marimba player Shimon, developed by Guy Hoffman et al. [103, 104], is capable of improvising with a human music player. In their early work they used the MIDI output from an

electric piano to analyze the piano player. With further developments the robot is used in live music performances with a jazz band[6]. They argue that:

> "Most computer-supported interactive music systems are hampered by not providing players and audiences with physical cues that are essential for creating expressive musical interactions. For example, in humans, motion size often corresponds to loudness, and gesture location to pitch. These cues provide visual feedback and help players anticipate and coordinate their playing." [103, p. 134]

Based on the MIDI signal there is a rhythm, tone or chord detection as input parameters for the implemented algorithm and output being a played chord, tone or melody. The improvisation system is based on standard jazz music theory. Their findings suggest that when a human tries to improvise based on the robots playing, visual cues seem to be helpful if the tempo changes, especially at low tempi, but humans use visual information only additional when they are able to, and otherwise fall back to rhythmic and auditory cues.

5.5 Related Projects

In this work we focus on string instruments, but there are many more types of instruments such as percussive or wind instruments. In the following subsections we present a selection of notable, influential and also novel approaches in music robots, mechanical and mechatronic installations. As stated above the classification of instruments is a complex issue and we are particularly interested in reflecting the so far mechanical, mechatronic and robotic developments, and for the sake of convenience we follow the classification of Ajay Kapur [95]. In each subsection the projects are ordered chronologically.

5.5.1 String Instruments

String instruments could be distinguished further into plucked, bowed, striking or wind powered string instruments, and to generate sound, force must be applied by at least one of those techniques to let the string vibrate. This vibration is either transmitted to a resonating body or to sound pickups to, if necessary, intensify the sound. In any case the pitch is determined by three main factors of the string: tension, length and linear density. A higher pitch can be produced by putting the string under tension, shortening the string length or using a different string with less linear density, and vice versa to generate a lower pitch. Following subsections give an overview of approaches representing the development process over the past decades.

5.5.2 Violin MUBOT

Developed in the 1980s the MUBOT (Musical Robot) is a mechatronic music instrument capable of playing a recorder, violin or cello [105] by according slight modifications. We focus on the violin MUBOT as the cello version works very similar. The bow is positioned on a fixed track, moved with an electric motor and an underlying pantograph mechanism. To select a string the violin is rotated. Pneumatic activated cylinders with attached rubber fingertips are positioned in half-tone steps along the violin's neck. A tone is played by activating the according cylinder pushing down the string on the neck and striking the bow.

[6] See the video of Georgia Tech's Robotic Musicians and Musical Cyborgs performing *Steady as She Goes* together with Shimon.

5.5.3 LEMUR GuitarBot

One of the most popular guitar robots is the GuitarBot[7] (see Figure 5.2) developed by Eric Singer et al. [89] in 2002. It is designed to play four strings. The same mechanism is applied to all four strings, which could be spatial separated from each other. The string is stretched within a aluminum base, and tension could be regulated. The string plucking mechanism is realized by a servo motor with four nylon picks attached resulting in novel, faster ways to pluck the string compared to a human player. To manipulate the pitch a movable bridge is positioned by a servomotor and a belt. With this approach two octaves could be played per string. Time needed from lowest to highest pitch position takes 250ms. Additionally, a solenoid, with an attached damper is used to stop or prevent the string from vibrating. The GuitarBot is controlled via MIDI input signal to support the twelve tone per octave arrangement, but it could be adapted to support alternative values. The electromagnetic single coil pickup was especially designed for this setting allowing a connection to an amplifier and loudspeaker.

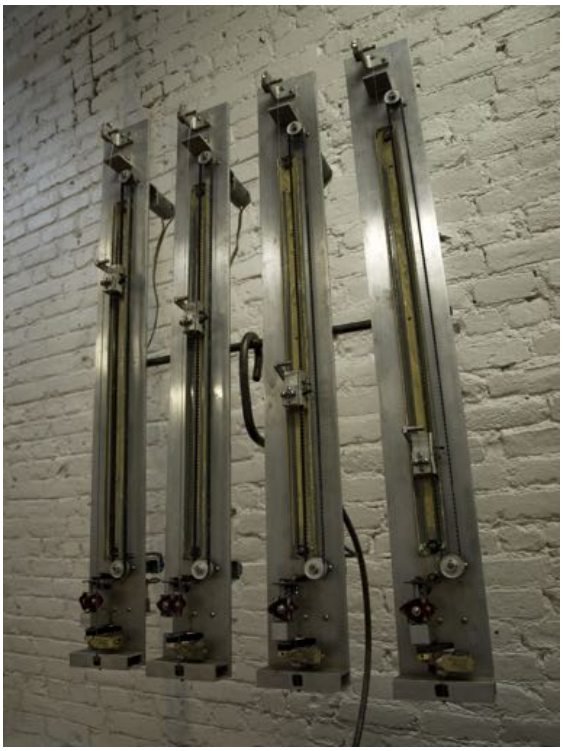


Figure 5.2: LEMUR GuitarBot (image source).

5.5.4 Plink Jet

The Plink Jet[8] (see Figure 5.3) created by Lesley Flanigan et al. [106] in 2007 is an DIY music instrument design approach using inkjet printer and common guitar hardware. It does not make a demand to be an accurate, fine tuned instrument, but is a playful approach of re-contextualizing

[7] See video of GuitarBot performing *EmergencyBot TV Theme*.
[8] See demo video of Plink Jet.

artifacts. The fretting is realized by the sliding cartridges and the strumming by a metallic strip attached to a stepper motor. It is designed to control four strings at a time. The same mechanism is used for all strings and they could be spatially separated from each other. Sound pickup is done by a piezoelectric microphone used for acoustic guitars, and the output signal could be received by a standard single quarter-inch output jack. Plink Jet can be operated at three different modes: (1) played by a microcontroller with predefined samples, (2) played by a human or (3) a combination of both. A three-way switch is used to determine the mode per string. If human playing is enabled, the cartridge position is regulated via buttons integrated in a potentiometer. Pushing left and right leads to an according response of the cartridge. The speed of the stepper motor to pluck the string is regulated via potentiometer. Besides the user input no environmental information is sensed.

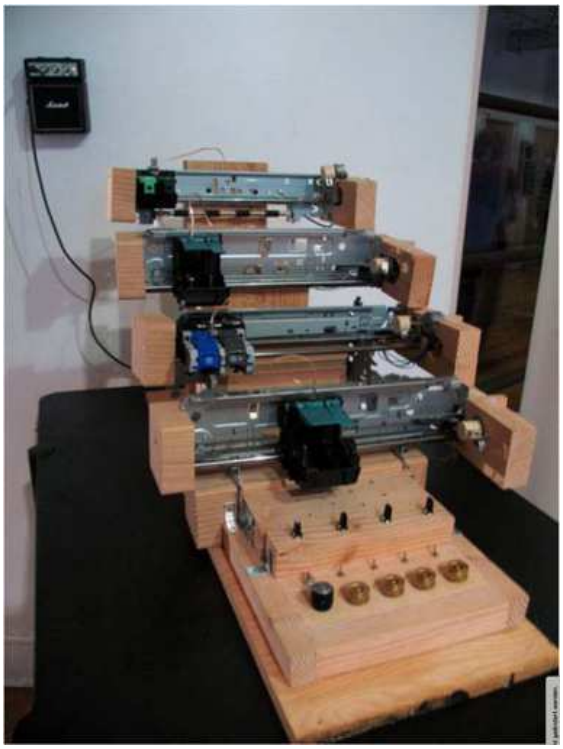


Figure 5.3: Plink Jet[106, p. 350, Figure 1].

5.5.5 MechBass

James McVay et al. [90] designed MechBass[9] (see Figure 5.4), a mechatronic bass guitar installation, in 2012. It consists of four single, electronically independent string units mounted vertically stacked on an aluminum rack. Its four main parts are (1) fretting, (2) plucking and (3) damping mechanism as well as the (4) optical pickup. The input signal is sent via external software using the MIDI protocol. The fretting mechanism consists of two solenoids, pulling down an acrylic plate to shorten the string's length. The solenoids are placed on a carriage system attached to a belt movable via stepper motor to a specific position relating to the according pitch.

[9]See video of MechBass performing *Hysteria*.

Plucking is achieved via stepper motor with five fixated picks. The damper mechanism reacts to the MIDI NoteOff signal and prevents the string from vibrating. By using a servo motor with a felt-padded arm, multiple degrees of damping could be applied. Instead of using a conventional electromagnetic sound pickup mechanism, which is not possible in this setting due to interfering of high electromagnetic noise generated by the used components, an optical pickup approach is implemented. An infrared LED is placed beneath the string and a phototransistor above it. A vibrating string leads to varying amounts of light received corresponding to the different frequencies of the according pitch[10]. The control system for the solenoids and actuators is realized by a an Arduino microcontroller with an attached JM2 actuator management board. The output signal from the optical pickup needs to be amplified and sent to a loudspeaker. MechBass was shown at exhibitions and used during live performances with human musicians.

Figure 5.4: MechBass[90, p. 4, Figure 9].

5.5.6 StrumBot

The above installations have in common that the mechanisms are built to play only one string. The StrumBot[11] designed by Richard Vindriis et al. [107] (see Figure 5.5) in 2015 uses a different approach. It was designed to play at more or less quiet public places such as venues or cafés. It is controlled via a MIDI input signal and demands not to pluck every single string separately, but to strum all six guitar strings at once like human musician would do it. This approach enables different musical expressiveness compared to the above projects:

> "StrumBot can perform slides, vibrato, muting techniques, pitch bends, pluck power variances, timbre control, complex chords and fast strumming patterns." [107, p. 146]

The strumming arm consists of two servo motors controlling a pantograph similar mechanism to strum a string and to regulate the position along the string's length where the attached pick hits the string. Two picks are used, one for down- and one for up-strumming. The pitch is regulated per string with one servo motor controlling the position of a carriage containing a clamp to fret the string. This system is twice as fast as the previously presented ones comparing the speed to travel one octave. The clamp again is driven by a servo motor and could be controlled for variable dampening control. For string selection a special algorithm was developed to convert

[10] See this guide by Steve Hobley to create a optical bass pickup from scratch.
[11] See demo video of StrumBot.

the MIDI notes to relating guitar chords trying to play the lowest note beneath the fifth fret. A Teensy 3.1 microcontroller was used with an attached Pololu Mini Maestro 24 to control the servo motors. Active electromagnetic guitar pickups are used to generate the output audio signal.

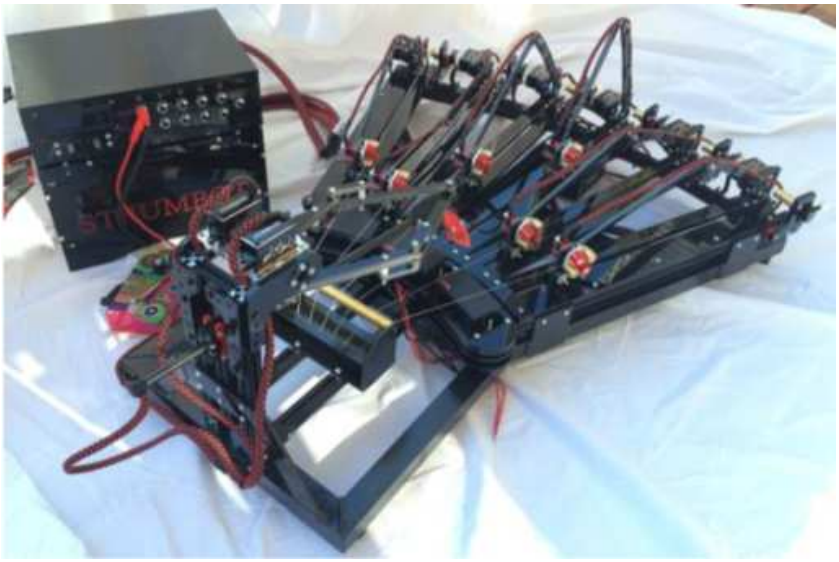


Figure 5.5: Strumbot [107, p. 147, Figure 2].

5.5.7 Cyther

Cyther[12] (see Figure 5.6) developed by Scott Barton et al. [88] in 2015 is a zither cooperative playable by human musician and a machine. It includes a tension and pitch based auto-tuning system, optical and electromagnetic sound pickup system. 20 Push solenoids are used for either striking or damping each one of its ten strings. microcontroller, software and motor control are self-assembled for this specific purpose. The software used for improvising is not specified despite frequency analysis is applied and rhythm, pitch and timbre are controllable. The tuning works together with the optical pickup system and string's tension measurement. The tension system is described by Barton et al. as follows:

> "As the tuning machine rotates to tighten the string, the ball end of the string applies force on the spring cap, which compresses the spring. As the spring compresses, the wiper on the potentiometer moves, which, with the appropriate circuitry, produces a varying and measurable voltage." [88, p. 322]

The worm drive controlling the string's tension is adjustable via motor. The electromagnetic pickup is used to generate the audio output signal via standard 1/4" jacks to be amplified and sent to a loudspeaker. The cyther was presented in artistic live performance scenarios.

[12] See video in live performance.

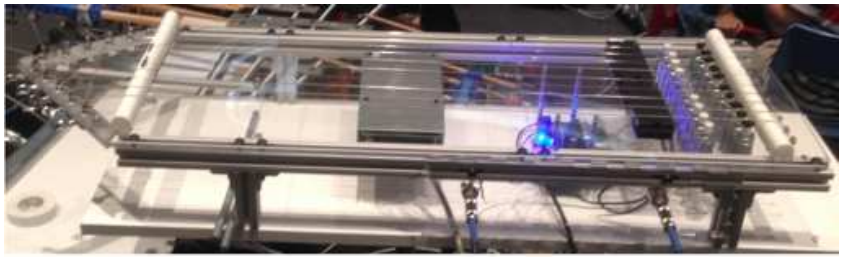


Figure 5.6: Cyther[88, p. 319, Figure 1].

5.5.8 RAEG

The Robotically Augmented Electric Guitar for Shared Control[13] (RAEG) developed by Takumi Ogata et al. [108] in 2017 allows a human guitarist to play the instrument while the instrument plays the strings either preprogrammed or computer generated. This approach uses a 3D printed hammer, controlled via servo motors to hit the strings and another servo motor to damp each string. Motor control is done by an Arduino Mega. The human guitarist could either choose to auto generate the string hammering or via external software using Max/MSP. This is a cooperative approach where machine and human complement each other to get the full potential out of the installation.

5.5.9 Keyboard Instruments

As shown in the above Section 5.2.2 player pianos and the mechanic constructions were developed two centuries before. The autonomous playing pianos evolved over time from piano rolls to floppy disks or CDs as an input and from mechanical to electronic devices [95]. Keyboard instruments are sometimes used to demonstrate a robot's capability of fine, exact motor movements. This also means that the mechatronic complexity of this construction is out of the scope of this work, but when designing a mechatronic music instrument this approach could be taken into account. As the player piano was already presented the following subsections show human imitating approaches.

5.5.10 WABOT-2

Shigeki Sugano et al. [109] developed the WABOT-2 (WAseda roBOT-2) at Waseda University, Tokyo, in 1985. This humanoid robot was designed to show the:

> "[...] 'soft' functions of robots such as dexterity, speediness and intelligence by the development of an anthropomorphic intelligent robot playing keyboard instrument." [p. 90][109]

WABOT-2 consists of two main parts, (1) the finger and arm movement and (2) its vision system. The input signal for the movement to play an organ comes from the visual system. The visual system reads a musical score sheet and those sensed instructions are used to determine the ideal finger key combination with a special algorithm. The finger movement is realized by multiple electric motors. Additionally, WABOT-2 could use its feet to include the organ pedals into play.

[13] See demo video of RAEG.

5.5.11 The ACT Hand

The Anatomically Correct Testbed (see Figure 5.7) (ACT) developed by Ada Zhang et al.[110] in 2010 demands to play the piano like an expert human. It implements the structures and biomechanic principles of the human hand to achieve this goal. This approach is special in a way that instead of designing artifacts and using components to mimic human physical behavior, it aims to imitate a human (body part) itself. The input signal to control the hand is a previously recorded MIDI file converting the signal to the according motor commands to play the keys. The ACT robotic hand was used in a conducted Turing test with human piano players. The result suggests that the audience could distinguish significant differences between the play, but could not determine which player was human.

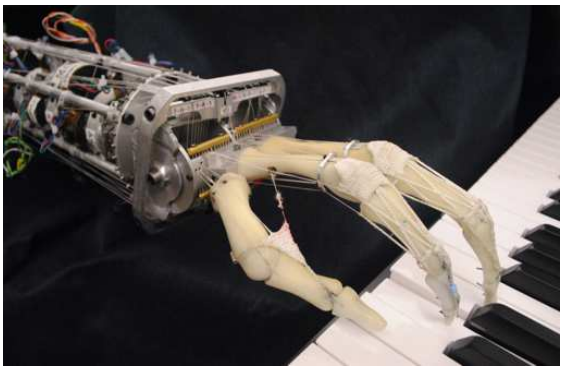


Figure 5.7: ACT Hand[110, p. 3536, Figure 1].

5.5.12 Percussive Instruments

The scope of percussive instruments is a fairly broad one. A common distinction in the classification are membranophones such as all kind of drums or idiopohnes such as glockenspiel or marimba. As the name states a membranophon consists of a membrane that is tensioned to some degree and vibrates. In contrast the idiophones vibrate usually as a whole. Following subsections show current robotic drummer approaches and mechatronic glockenspiel.

5.5.13 Nico

Christopher Crick et al. [111] have created the robotic drummer Nico in 2005. Their work's main goal was to test social synchronization tasks with humans by not only sensing the auditory environment. Therefore Nico plays drums together with another human and a second human conductor. Nico is equipped with an auditory and a visual processing system. The visual system consisting of a camera, which is used to detect the conductors ictus under the condition that the hand of the conductor moves below and above an imaginary line to indicate the beat. Two microphones represent the auditory system and detects the beat and its intensity. A beat is only classified as one if it is above a specific intensity level. Using a not closer described learning algorithm, after a test phase Nico tries to predict the next beat and to synchronize its drumming with the humans.

5.5.14 Haile

Gil Weinberg et al. [74] developed the percussionist robot Haile[14] in 2006 to play a Native American pow-wow drum. This instrument is used as a multi-player one and supports cooperative drumming. Haile has two arms with different mallets. One is bigger for better visibility and louder sounds, and the other one is smaller for faster and more quiet sounds. The mallets are controlled via solenoid and a spring to pull them back from the membrane. Servo motors control the position of the mallets to play more at the edge or in the center of the drum. The input signal to control the actuators could be sent via Max/MSP from an external device, but it also supports six interaction modes. A microphone and respective audio analysis allow (1) imitation mode of the human player after they stop playing, and (2) a stochastic transformation to apply multiplications or division on the perceived human play and then repeat it. The (3) perceptual transformation mode plays similar rhythms as the human played before. (4) Beat detection mode analyses the given beat for a few seconds and then locks it, as humans tend to adjust their beat to the Haile's one. The (5) simple accompaniment mode uses fix prerecorded MIDI files and (6) perceptual accompaniment mode, wherein the user input is analyzed at the same time Haile is playing along with the human drummer. Therefore short sequences are looped while analyzing the input and creating a proper response as another loop. It was used in live performances.

Figure 5.8: Haile (left) performing with a human player (image source).

5.5.15 The Closed-Loop Robotic Glockenspiel

Jason Long et al. [112] created a mechatronic glockenspiel (see Figure 5.9) in 2016. Beneath each glockenspiel's key a push solenoid and a electromagnetic pickup coil are placed. A wooden ball is attached to the solenoid to provide the "most agreeable timbre." The control system is

[14] See demo video of Haile performing with a human percussionist.

realized with two Arduino Due microcontrollers for processing the input MIDI signal. The output signal gained from the electromagnetic pickup is filtered (low-pass, noise, high-pass) and amplified before it is routed to a 1/4" panel-mounted TS socket. Additionally it has a MIDI output. As onset, amplitude and pitch detection are already implemented, the device could be used as a MIDI controller.

Figure 5.9: Closed-Loop Robotic Glockenspiel [112, p. 4, Figure 3].

5.5.16 Wind Instruments

John Henry van der Meer gives an overview on music instruments in his work *Musikinstrumente: von der Antike bis zur Gegenwart* (en.: music instruments: from ancient to present times)[113]. He classifies the instruments into woodwind instruments such as flutes and brass instruments like horns and trumpets. For flutes he describes an evolution of the nowadays known flute instruments in the Romantic era (early 19th century). In this period many new valves were invented. Throughout ancient ages flutes were made of wood and over time the number of holes increased. Organs were already present in ancient Greece using water for hydraulic pumps. The origin of bagpipes is not known, but they were seemingly present in the ancient ages, in any case throughout the medieval ages in Europe. In 1582 an automatic trumpet machine was invented in Augsburg[15]. Following, we show an automated bagpipe and a flutist robot.

5.5.17 McBlare

The robotic bagpipe player, McBlaer[16] (see Figure 5.10), was developed by Roger B. Dannenberg et al. [114] in 2004. It uses a standard Highland Bagpipe. A custom-built air compressor powered by an electric motor takes care of the air supply. The chanter control, respectively finger control, is done with electromagnet actuators pulling down metal plates with attached rubber circles to

[15]This piece can be seen at the Kunsthistorisches Museum Wien. A picture of the machine is provided on the museum's webiste.

[16]See video of McBlare performing *Highland Laddie*.

seal the hole. The input signal comes from a MIDI sequencer or a especially designed GUI, but in theory all MIDI controllers could be used as input devices. McBlare was designed for the twenty-fifth anniversary of Carnegie Mellon University's Robotics Institute and later shown at exhibitions.

Figure 5.10: McBlare[114, p. 4, Figure 4].

5.5.18 Anthropomorphic Flutist Robot

The Waseda Flutist Robot was originally developed in 1990[17]. The latest revision[18] (see Figure 5.11) was presented in 2007 by Jorge Solis et al. [115]. It is a humanoid robot consisting of an air pressure control system responsibly for nine bellows sending air into a pipe moving a cylinder attached to a set of levers. Those levers enable the moving parts such as fingers and tongue. The system is controlled via personal computer receiving a MIDI input signal and feedback data through a pitch evaluation system. The MIDI input stems from a recorded professional flutist to avoid monotonous sounds. The played sound is analyzed using the Cepstrum method supported by the information from the MIDI input signal. The pitch detection is used for a quality analysis of the played note to autonomously improve the sound quality by changing the volume level respectively the air pressure. The different version of the Anthropomorphic Flutist Robot have been used for multiple purposes and the latest revision WF-RVI [116] was built in 2010, and underwent constant improvements. It is regularly presented at respective conferences ever since 1990, and shown at exhibitions and live performances including human musicians.

[17] See the development history on the project website.
[18] See video of WF-4RV performing *Autumn leaves*.

Figure 5.11: The Waseda Flutist Robot No.4 Refined IV (image source).

5.5.19 Other Instruments

Before the availability of music software or even personal computers in the 1970s, turntable robots were invented [95]. They were capable of playback samples and applying modifications such as reverse play and speed regulation by using a special protocol to perform the according actions. One famous, mentionable example of an mechatronic device is the Marble Machine[19] of Wintergatan. It is basically a mechanic device, but also includes an electric bass. The human player uses a crank handle to drive the machine causing marbles to fall on percussion elements and the bass' strings. It is both an autonomous and a cooperative approach. Besides the cranking the machine plays one predefined song fully autonomous. It is also cooperative because the marbles determine the tempo and string played, but the human player can choose the pitch by selecting the fret. Additionally the machine can be paused and the player is able to play as she or he likes. The sound pickup is done by multiple microphones.

[19] See the video of Wintergatan's Marble Machine.

Part II

Methodology

Author: Jakob Blattner

Developing and designing a product confronts the creators with a number of different methods that can be applied to create this product so that it will meet its technical, functional and social requirements [117]. The first chapter of this part analyzes all of the discussed design processes in Section 3.3, their respective use in the context of this thesis and select one process model which suits this context the best. Additionally, all design attitudes (see Section 3.3.2) will be analyzed. All design principles and guidelines, which have been treated in Subsection 3.4.1 and 3.4.2 respectively, will be considered in the design cycle and underly no further analysis, since these were already selected in advance for the topic of this work. The second chapter deals in detail with each individual phase of the selected design process and discusses methods used for development in each phase, how these are carried out and what results are expected from them.

The importance of this step is significant, as the further course of development is based on the methods and model chosen. A retrospective change (if made) would not only have a negative impact on the time required but may also have a negative impact on the quality and results of this work.

CHAPTER 6

Design Process and Attitudes

Author: Jakob Blattner

As well as in Section 3.3, the analysis of the user interface design methods starts with the process models. The first section contains the discussion of all presented process models in Subsection 3.3.1, followed by an argued selection of one of them. The second section discusses the possible problems and advantages of the, in Subsection 3.3.2 presented, design attitudes.

6.1 Process Model Selection

The first model to start with is the *Goal-Directed Design* Model from Alan Cooper. As already stated Cooper explicitly focused on the design phase of the iterative development cycle. The main problem with this approach is that Cooper's process is based on a larger group of designers and programmers. The team of this thesis consists of two people, a subdivision into designer and programmer would not make any sense, since all project and design decisions as well was the programming were made together. In particular, this stays in direct conflict with Cooper's definition of *work in groups of two* and *separate responsibilities*. However, this does not mean that the other amendments proposed by Cooper can't be applied. *Design first, program second* is a change that has been incorporated into the design as well as the use of personas. The advantages for completing the user interface before the programming are obvious. Forcing an interface into a framework already defined by the software being used brings compromises with it. These compromises can fall to the disadvantage of the future user, as design decisions are left out in the favor of already existing, programmed solutions. When designing before programming, no decision is limited by this circumstance and the UI can be produced in full favor of the users. Cooper explains, that the advantage of personas is a great addition to the development process. Stakeholders are made aware of the difference to their own previous knowledge and other prerequisites. The immediate consideration of these key points of the target group results in fewer design cycles and thus faster and higher quality results, as why personas are integrated into the design process of this thesis.

The next process that will be discussed is Mayhew's *Usability Life Cycle*. As already described, the cycle is very strictly defined, which is, amongst others, visible through organizational and

management activities. Besides the fact that it is not possible to completely cover the cycle, it is also not possible to switch back from each phase to the previous one. An example of this is the installation phase where is no way back to requirements analysis. In a project that has never been done in this way before, with very unclear requirements and specifications, this can lead to an unsatisfactory result if strictly adhered to. This and the strict procedure of this model disqualifies it for further application.

Rosenzweig's *Iterative Design Cycle* has fewer phases compared to Mayhew's model. Nevertheless, the strictly prescribed sequence, arranged in a circle, is apparent. It is not possible to skip any phase, to update and re-evaluate a prototype after evaluation or to carry out only one type of evaluation (using quantitative or qualitative methods). The two types of evaluation are of course applied and adopted, but the time of use, as well as the sequence of steps, is chosen at one's own discretion and adapted to the respective situation.

This leads to two remaining process models: the *Star Life Cylce* by Hix and Hartson and the *ISO 9241-210* process model. These two models are compared on the basis of their structure and phase sequence. As already stated, Hix and Hartson didn't implement a certain starting point to support different ways of development. This variable entry option offers a high degree of adaptability for various projects. A high degree of flexibility is also provided by the variable sequence of phases. This would make it possible to create a first prototype before performing a task analysis, which can pose difficulties since a device to carry out the task of this work has never been realized before. In contrast to the ISO model, the star life cycle has two different implementation phases. *Rapid prototyping* should mean that implementation is not started immediately without evaluating it with users. The ISO model leaves more room for freedom. it also leaves further flexibility for the methods to be applied in each phase. Hix and Harson leave the sequence of phases in the star life cycle nearly completely to the designer, but the content of many phases are predetermined and thus stricter than in the ISO model. "Nearly" because an evaluation has to take place after each phase. This is not always a sensible decision to make. Evaluations after said difficulties of this project in terms of requirements and task analysis do not bring any advantage. For an evaluation, a device that supports the assumptions from the phases would have to already exist, since it is up to the authors to create it. The problems discussed also occur in the first two phases of the ISO model, but it is not necessary to carry out an evaluation immediately afterwards.

The general freedom in the respective phases and the more sensible phase sequence for this project lead to the decision that the ISO model will be applied for this project.

The following section is dedicated to the review of the described design attitudes from Subsection 3.3.2.

6.2 Chosen Design Attitudes

After a process model has been selected, there are different design attitudes that will also be analyzed for this project. The attention is mainly drawn to possible problems and explicit advantages in the context of this work. The order of the presented attitudes orients itself upon Section 3.3.2, in which the respective attitudes were discussed.

Participatory Design brings difficulties with it. Firstly, a person (or several persons) must be found who participates in the project of this master thesis and fulfils the characteristics of the target group. Since these in turn, with the exception of previous musical knowledge, are very broadly diversified, the participation of a person not automatically guarantees to be successful. A certain degree of unselfishness must also apply, since the resulting installation will be used in a

context where very few people, mostly museums, can benefit from the installation. In addition, the authors cannot serve with any material reward for their participation. Finding several people for this project is also very unlikely. Even if all these circumstances would not apply, the question arises, quoted by Nielsen, whether the user is always right with all his statements about the project. This ambiguity and the difficult preconditions exclude the application of participative design.

Playful Design is very important for this master thesis. In combination with an explorative approach, the initial interaction of users is facilitated. Robson (see Subsection 3.3.2) underlines the positive effect of the playful approach on non-musically trained people. An interface unknown to the user with which to interact in a playful way, adapted to the application environment, reduces the risk of appearing incompetent. The mentioned time savings in interaction is also extremely practical in this case, as it is likely to be set very low in the museum context. With the playful approach, however, great attention must be paid to the variable difficulty level. The interface must provide an easy entry combined with a scalable difficulty. If this is the case, a flow-state can be achieved and the interface, depending on the user's sense of difficulty, can be fun.

CHAPTER 7

Design Phases

Author: Jakob Blattner

After defining which process model to use and which of the design attitudes can be applied in the process of the practical part of this thesis, each individual phase of the chosen ISO process model will be gone through. This includes every design method used in the respective phases.

A look at the graph of the ISO model, visualizing the development process, shows, that some intermediate steps are noted in addition to said phases. The first of these steps is at the very beginning of the process and is named *Plan the human centered activities*[118]. The purpose of this step is planning when and how the human centered activities will be integrated into the overall project, as this is the same aim of this chapter.

7.1 Understand and Specify Context of Use

This phase serves to identify users (and other stakeholder), their tasks, characteristics and the technical and physical environment in which the system will be used. The acquired information serves as a basis for the agile design process and is grouped in three profiles.

The so-called user profile includes all relevant characteristics of the user group. This includes knowledge, skills, experience, training, physical characteristics (for example disabilities), preferences and abilities of users. Another profile is the environmental profile which describes the technical environment in which the system will operate. It includes the hardware, software and other materials being used and other relevant characteristics of the physical environment (lighting, spatial layout, furniture and thermal conditions). The last profile is the user task profile. It is made of a description of tasks the user has to carry out, including the frequency and duration of tasks and interdependencies. All theses profiles should be described in sufficient detail to support requirements, design and evaluation activities.

Because of the fact, that the user profile itself is not as clear as in maybe other projects, the profile will be implemented in the form of personas. The environmental profile can also only mainly be filled with assumptions, as specific details vary from museum to museum. This problems are the

reason, why the proposed profiles are combined in one working document, consisting of questions and requirements which are collected in the course of this work.

This document, which start as outlines and get more and more detailed, reworked and updated as the design process continues. The collection of information for filling this document is based on different methods of information retrieval. In the phase which gets represented by this section, three methods are used to gather data for the working document.

7.1.1 Literature Research

The first method is the literature research and is held at the beginning of every design process. It serves the purpose of becoming acquainted with topics that are, directly or indirectly, affected by the system or the system itself is based on it. The research ranges from similar projects carried out in the past and ends with theoretical foundations of affected topics. Current or completed projects can be analyzed and critically evaluated on the basis of the technologies used. This means, that literature research can cover a wide range of topics. It also includes the finding and definition of principles, guidelines and any other knowledge from Section 3.4. Because of the wide range of information gathered throughout the literature research, it provides information for every profile explained in the paragraph above.

The research itself has been conducted on several academic online databases and search pages like Google Scholar[1], ACM Digital Library[2], IEEE Xplore Digital Library[3] and the website of the library of the Technical University of Vienna[4]. Specific websites for Musical User Interfaces, especially the archives of the NIME[5], have been searched. Apart from the internet, books were borrowed from the aforementioned library of the TU Vienna and from the Institute for Design & Assessment of Technology for research purposes. The found and used literature has been managed with the reference management tool Mendeley[6]. The results of the literature research are the presented in the first part of this thesis.

7.1.2 Interviews

Interviews are one of the most frequently used user research techniques [69]. In the broadest sense, an interview is a guided conversation in which one person seeks information from another. There are a variety of different types of interviews which can be conducted. The end result of a set of interviews is an integration of perspectives and knowledge from multiple people. Expert interviews, as conducted in the course of this thesis, have specialists as interviewees, answering questions which came up during literature research. As well as the literature research, interviews help gathering a broad range of information.

The authors prepared one interview guideline (see Appendix B) with the two main topics composition and interfaces, but adapted the questions for every interview with respect to the interviewee's expertise. The interviews took place at the interviewee's offices and at the rooms of the Multidisciplinary Design and User Research Group of the Institute for Visual Computing and Human Centered Technology. Every participant signed a consent form (see Appendix

[1] https://scholar.google.com/
[2] https://dl.acm.org/
[3] https://ieeexplore.ieee.org
[4] https://www.ub.tuwien.ac.at/
[5] http://www.nime.org/archives/
[6] https://mendeley.com

A.1), granting us the permission to record the interview and to take notes, using their data in anonymized form.

During each interview two persons were present in addition to the interviewee. The first person acted as interviewer, while the second took notes. All conversations were recorded using two smartphones.

7.1.3 Personas

A persona is a description of a fictional person to describe a specific kind of user group and thereby mainly contributes to the creation of the user profiles. This representation is necessary, since it's impossible to speak with every future user but a depiction is still needed. Personas bring many advantages with them. The potentially abstract connection between the designers and users is more easily overcome. They also get every team member to think about the same persona instead of everyone having his/ her own vision of the future user(s). With specific targets to focus on while producing solutions, the outcome will assure greater success than without. Personas can also be used as a discussion tool and for new team members to quickly adapt mentally to the target group. Creating one persona per user type is the absolute minimum, because multiple personas cover a greater range of characteristics for each user type and therefore prevent the design to be focused on just one certain kind of user. The product can change throughout the development time. As a result, personas have to adapt to those changes and thus must be updated to reflect them. It is also important to note, that personas do not replace conducting user research activities.

7.2 Specify Requirements

The ISO Norm 9241-210 originally defines the name of this phase as *Specify User Requirements*. As explained in section 3.5.3, user requirements are not the only kind of requirements which exist. To enable a more specific and thus argumentative better coverage and understanding of aspects regarding the project of this thesis, not only user requirements, but all types of requirements will be defined. The entire spectrum of requirements also enables with the concrete notation and extension of the working document defined in the previous section. To ensure a seamless management of requirements, *Requirements Engineering* will be conducted in the course of this work.

7.2.1 Requirements Engineering

Requirements engineering is a collection of methods and descriptions to collect and manage requirements. According to the International Requirements Engineering Board (IREB), requirements engineering consists of four main activities: [119]

1. *Determination of requirements*: An important activity in requirements engineering is the determination of the requirements for the system to be developed [69]. There are three different kinds of sources of requirements: stakeholder, documents and existing systems in action. During the course of the project, existing requirements will be updated or deleted and new ones will be added. Therefore requirements and their documentation must be updated throughout the project [118]. Most requirements are learned in the first design phase of the MUI (see Chapter 11) and then kept up to date.

2. *Requirements documentation*: A documentation technique is any type of more or less formal representation that facilitates communication between the individual stakeholders and increases the quality of the documented requirements [119]. In principle, all techniques can be used to document requirements, from natural language descriptions in prosaform through structured natural language texts to more formal techniques (e.g. state diagrams). The authors will document the requirements in the form of short descriptions and an enumeration of all related requirements.

3. *Auditing and reconciliation of requirements*: The testing and coordination of requirements in requirements engineering is intended to ensure that the documented requirements meet specified quality criteria, such as the following:

   - Uniqueness and consistency
   - Clear structure
   - Modifiability and Extendability
   - Completeness
   - Traceability

   These quality attributes must be fulfilled and looked after throughout the development process.

4. *Requirements management*: The requirements management defines the priorization, versioning, management of changes and measurement of requirements (e.g. choosing how a requirement will be measured, setting limits, etc.). In the course of this thesis, every group of requirement will have it's own priorization (in three steps). The versioning of a requirement is also recorded in the course of the work. It should be noted that if there is no version number at the end of a request, this is the first version of the request. The measurements for certain requirements are, if necessary, documented into the requirements itself.

The second source for the working document will be discussed in the following subsection.

7.2.2 Use Cases

Use cases are primarily a way of expressing the requirements of a system, especially those related to a user's behavior [120]. They represent what that the system does and how it behaves for its stakeholders. Not all requirements can be well described as use cases and don't benefit from being forced into the narrative structure of use case descriptions. An example would be the requirement "The programming language used must be Java". This is why only user requirements will be transferred into use cases in the course of this work. A use case consist of a number of elements:

- *Name*: Each use case should have a name that indicates what is achieved by its interaction with the users. This name should be unique.
- *Brief description*: A brief description of the use case and (if necessary) the affected user.
- *Flow of events*: A description of what the system does in regard to the use case. It should not be described how the system solves specific problems. The description must be understandable by all stakeholders.

- *Special requirements*: This part of the use case lists all requirements, such as nonfunctional requirements, on the use case that are not considered in the flow of events, but that need to be taken care of during design or implementation.
- *Precondition/s*: The preconditions define a constraint of events that happened before the start of the use case.
- *Postcondition/s*: The postcondition defines a constraint of events which happen after the use case has terminated.
- *Extension point/s*: A list of points within the use case at which additional behavior can be inserted.
- *Diagram/s*: UML diagrams that illustrate aspects of the use case, such as the structure of the flow of events or the relationships involving the use case.

A benefit from implementing use cases in the development process is the detection of problems, which, if not detected, can lead to interruptions in the development process. In the case of interruptions, changes must be implemented and resources distribute differently. The later requirements become known, the more problematic and expensive the development becomes. The part of the development which concerns the production of the design solution is the topic of the following section.

7.3 Produce Design Solution

This design phase is all about the production of the system in a form and ways, that it meets the specified requirements and working document created in the previous phases. Topi and Tucker [118] define six main steps in designing usable software, which partly overlap with already discussed topics and which the authors will also orientate upon:

1. Structure solutions around key tasks and workflows based on the context of use and user requirements specification. A key issue is the *allocation of function*, which describes the decision of which tasks should be automated by the system and which should be under the control of the user.
2. Design the interaction, user interface, and navigation from the user's perspective and keep them consistent. Pay attention to other software the user is likely to use, and be careful when introducing a different/new type of user interface.
3. Keep the navigation consistent. If different teams design different parts of the system use style guides for a consistent user interface.
4. Follow interface design best practice. There are many guidelines (see Subsection 3.4.2) which are internationally agreed best practices for achieving usable hardware and software.
5. Produce sketches and mockups early to test assumptions. This enables the designers to show proposed designs to users and other stakeholders to obtain feedback before a certain design is agreed upon.
6. Keep testing solutions with users until the quality criteria are met. These evaluations can take place throughout the design process.

In the fifth step mentions mockups and sketches, which are part of the prototyping process. Prototypes are specific tangible representations of interactive systems, which support creativity, communication and early evaluation in a human-centred design process [121]. They allow users to experience th product before the creation of it is actual complete. The purpose of prototypes is to generate and express ideas, to reflect on them and to evolve these ideas into a working system. There are several types of prototypes which differ in their precision of detail. At the beginning of the design process, low precise prototypes are being used to save costs, evaluate design possibilities and expose errors. Further in the design process, prototypes become more precise to further specify all interface elements in their content, size, and position. Through their interactivity, prototypes also provide a look and feel of the future product and thus give another possibility to evaluate the future haptical and visual design of the prototype.

The following paragraphs describe the difference between the different kinds of prototypes which have been used in the course of this thesis, starting in an chronological order.

7.3.1 Sketches

Sketches are rapid prototypes, a kind of prototype which are created quickly for a short usage and can be thrown away after they have fulfilled their purpose. Sketches can be created by pen on paper (see Figure 7.1a) or cards in a short amount of time. The detail of sketches is, compared to other prototypes, very low (see Figure 7.1a). They serve the quick documentation of new ideas, comparison and combination of two designs and the support of the thinking process.

7.3.2 Wireframes

Wireframes increase the level of detail compared to sketches. They can be created either on the computer using drawing programs or on paper. They are supposed to give a first feeling of the design and features of the product by using concrete interface elements integrated into a sketched or printed frame. Colors, fonts and pictures are still not an issue, whereas navigational elements, structure of the current UI and combination of UI elements are important.

In the context of this work, wireframes were increasingly used in combination with three-dimensional mockups and step-by-step in creating the software-side solutions (using Unity3D, see Subsection 12.1.4) of the GUI.

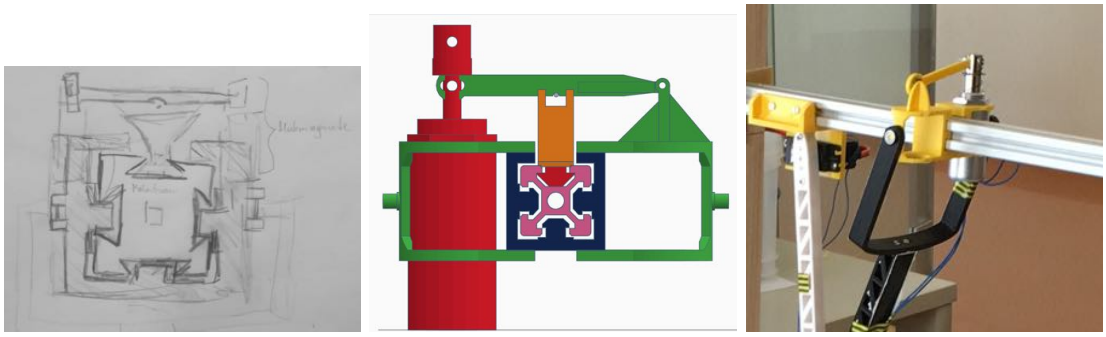
7.3.3 Mockups

In contrast to the previous prototypes, mockups can, but don't have to be three-dimensional. They display a representation of the future product, but do not include full functionality. The purpose of mockups is to create a look and feel that helps to uncover potential problems and inconsistencies. To compare multiple ideas, multiple mockups can be created, which in turn help to give a deeper understanding of each idea itself.

During the development of this work, three-dimensional mockups were created for most of the input and mechanical components (see Figure 7.1b). This had the purpose to either test the look and feel of the component or test the functionality of it.

7.3.4 Functional Prototypes

A functional prototype is a realistic and specific interface that contains the main functionality of the product to be created. This type of prototype serves the demonstration purpose of the



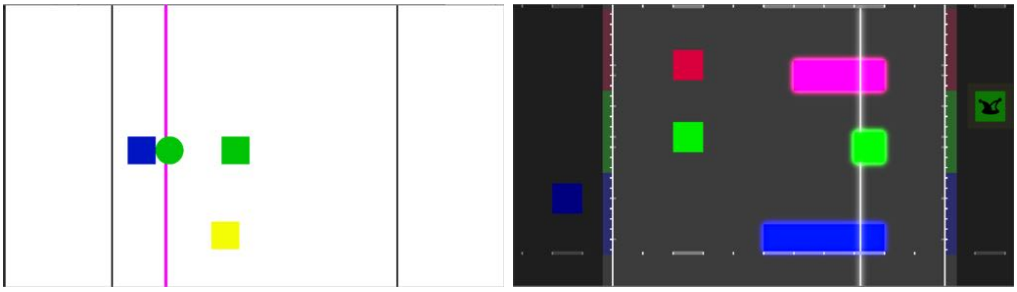
(a) Sketch (b) Mockup (c) Hi-Fi Prototype

Figure 7.1: Three different types of detail regarding prototyping

designed technology. However, the functionality of a prototype is more advanced than the design. The level of detail of functional prototypes can be divided into low-fidelity (low-fi) and high-fidelity (hi-fi) prototypes, where hi-fi prototypes are even closer to the functionality and design of the finished product than low-fi prototypes (see Figure 7.2).

In this work the functional prototypes have been developed with the help of different soft- and hardware components. The hardware being used for the creation of the mechatronics mechanism ranges from 3D prints over different types of Arduino boards[7], stepper and servo motors, different kind of cables, infrared LEDs and other electric components. The software created for this mechanism was written in Python (for more information, see Part III). The hardware used for the MUI itself were also 3D prints and IR LEDs, wood, metal products, a beamer, camera module etc. (see Part IV). The software used for the GUI of the TUI was Unity3D, written in C#, in combination with the use of the uniducial library[8]. Software used for both parts of the project were Cura[9] and tinkercad[10], both for the development of 3D prints. Github[11] and git[12] were used for the version control of the created software. Further information about the prototypes can be seen in the respective parts of the thesis.

[7] https://www.arduino.cc/
[8] https://code.google.com/archive/p/uniducial/
[9] https://ultimaker.com/en/products/ultimaker-cura-software
[10] https://tinkercad.com
[11] https://github.com/
[12] https://git-scm.com/



(a) Lo-Fi Prototype (b) Hi-Fi Prototype

Figure 7.2: Prototypes in the course of the GUIs development.

7.4 Evaluate

The evaluation applied in this thesis is oriented towards the user. The so called *user-centered-evaluation* is an evaluation based on the user's perspective and is the main source of feedback in human-centered-design [118]. Due to the fact that the theory behind usability and user evaluation has already been discussed in Subsections 3.5.1 and 3.5.2, only the selected test properties are listed in this section. It must be noted, that the mechanical part alone was, in its iterative creation process, not part of an evaluation where any user was present. This is because of the fact, that the requirements for the mechanical part were solely based on benchmark values and had never any purpose of being directly used by the user.

Two evaluations were held in the process of this work. One for every design iteration taken. In order to gain additional information and feedback about the installation, the first user evaluation participants were experts and non-experts alike (see Chapter 11). One of the experts for the first evaluation was asked to participate in advance. The same expert sent out invitations to low semester students, of which noone came. This is why the participating non-experts, as well as the rest of the experts, were asked to participate via hallway-recruiting. Hallway-recruiting (derived from the official term *hallway testing*) describes the act of asking people on public places to test a certain object. At the second evaluation, all employees of different institutes of the Technical University of Vienna (who had no connection to UIs or similar) were asked to participate in the user test by e-mail. Additionally, the authors asked colleagues to participate in the user test. An important condition here was that none of the participants had any kind of information about the installation. Furthermore, the author who did not know the current participant carried out the respective user test, while the other author watched and took notes. Both evaluations were conducted in a technical laboratory for students and employees of the Institute of Visual Computing & Human-Centered Technology of the Technical University of Vienna. The test environment in the lab was not ideal, as the surroundings were not the same as in the future use context. Nevertheless, measures have been taken care of to exclude noises, uninvolved laboratory users or other distractions in advance. Before the users took part in the test itself, they were asked to sign a consent form after which the test personal explained the goal of the research. The user tests of the first iteration were conducted as a mixture of open-ended and task-based testing. At the beginning of the test, the participants was given enough time to interact with the system all by him-/ herself. As soon as the user had finished, she/ he was given short tasks to use certain interaction methods which haven't been used in the previous interaction with the installation. The tasks were abolished in the user tests of the second evaluation, as the TUI was in a further development stadium than in the first evaluation. The first evaluation was conducted

with mockups which simulated the interfaces look and behavior. No experts nor users got any help from the test personal. This can be justified by the future use context, where no help or clear tasks can be given either (see Chapter 11). Both participant groups were asked to express their thoughts aloud. Apart from thinking aloud, video and audio recording has also been conducted. In addition to all qualitative user research methods being used, logging provided quantitative data (see Subsection 3.5.1) for future analysis, in which the data was visualized via Python[13] and the use of a plotting library[14].

After every phase of the creation process has been covered by this part of the work, the following part now starts with the documentation of the praxis part of this thesis.

[13] https://www.python.org/
[14] https://matplotlib.org

Part III

Practical Part - Mechatronics

While the outcome of this work, the design of a prototype, of course is a coherent system it consists of two main parts: (1) the mechatronic sound generating system and (2) the musical user interface. The procedures differ, as we did not follow the ISO model within the following mechatronic iterations chosen in Section 6.1, but nevertheless followed an iterative design approach for the mechanical part (see Figure 7.3 shows the separation of the two parts and the schedule). The first iteration of this part after the literature review was finished before the expert interviews. That was necessary to gain first insights when construction a mechatronic instrument and put specific questions to the according interviewees. Also we did only evaluate the iterations with respect to our own arbitrary defined requirements. Those requirements are to support a tempo up to 180 bpm (see non-functional requirement 8.), because this is within the presto (fast) musical tempo range (see Section 4.1). If designing a mechatronic instrument, users should also be capable of playing fast. Regarding pitches precision must not exceed the auditory threshold of roughly (±8 cents) (see non-functional requirement 9.) between two frets following Vindriis et al. [107] StrumBot approach. Additional requirements are found in Appendix D.

We chose a guitar (see non-functional requirement 5.) as it is a commonly known instrument in many musical genres. This means that people have an expectation of how a guitar sounds, whether or not they are capable of playing the guitar. As we want to support users in creating good sounding music, we assume it would be advantageous to tie up with their previous expectations.

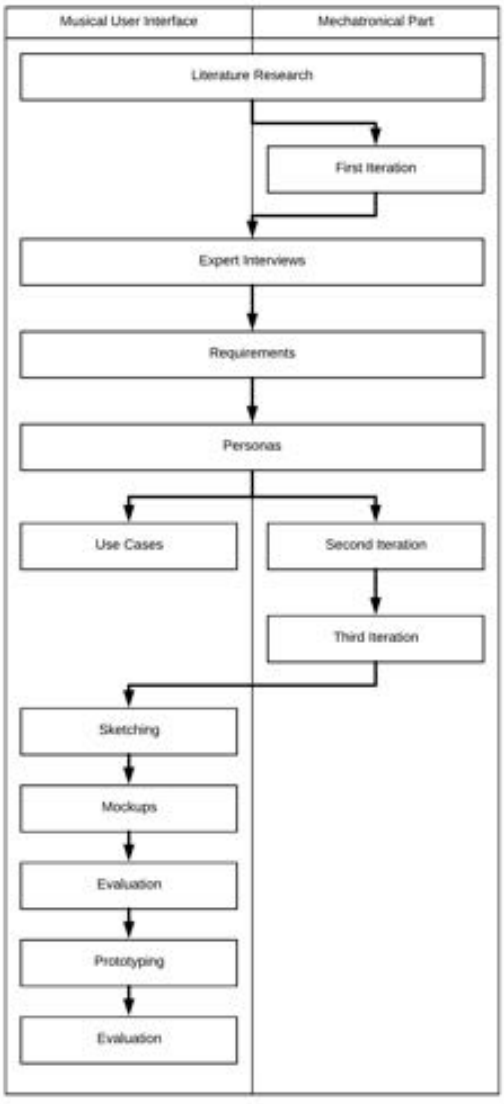


Figure 7.3: Course of this project.

CHAPTER 8

First Iteration

Author: Raphael Kamper

Based on the findings from our literature review (see the following section for a summary) we decided to implement a module based system. A module consists of the mechatronic components to play a string. This means a carriage system transporting electromagnetic push-pull solenoids to adjust the pitch, and a servo motor because of its speed and noise properties. When using a fret based system to determine tone height, there is no need for a precise positioning of the carriage system. It just has to be placeable between the frets. A second servo motor is used for damping the string and a stepper motor for string plucking.

8.1 Literature Review Findings

This section provides a short summery including the key findings of Chapter 5 on which we built our presumptions for this iteration.

Generally speaking, all of the shown projects do have an exhibitional character. They are all presented for indoor uses. This is not a necessary but a convenient limitation, as power supply, general infrastructure (e. g. audio equipment) and (a specific) audience are found at event locations or museums. This is common for other music performances and concerts. Most instruments are designed to be played by one person. There are of course exceptions, and some are more often used as accompanying instruments. Music is a highly cooperative interaction, but still the instruments are usually played by a single person, whereas this is different for percussion, and only rhythm has to be synchronized by the musicians, but not tones and harmonies. Cooperative approaches on classic instruments could be of interest for musicians (novices and professionals) as it opens up a new way of playing. Nevertheless, whether or not a cooperative or single player instrument is created, has to be taken into account from the very beginning of the design process.

From a mechanic perspective the instruments are usually manipulated to be played automatically. When the installations goal is to replay already existing pieces, it is more common for string instruments to rebuild the instruments instead of building a surrounding device fully capable of playing it. The often built-in core components are stepper and servo motors or solenoids. Additionally, feedback position systems are used in almost all modern approaches.

If no resonating bodies are existent, sound pickups must be implemented. An optical pickup approach seems to be a good low-cost alternative compared to microphones or other electromagnetic sound pickups. High-Level Feedback (musical environment, see Section 5.4.3) would be a more like a nice to have for our context, although it seems to be a promising feature.

8.2 Underlying Principles

A typical guitar scale is somewhere between 620 mm and 650 mm, with a common length being 648mm[1]. There is a formula to calculate the fret position for a given scale. As we know from the Subsection 4.1.5 in a tone system where an octave is divided into twelve semitones the factor per step is determined by $\sqrt[12]{2}$. This means that the fret position is calculated by $S - \frac{S}{\sqrt[12]{2}^f}$ with S being the scale length in millimeters and f being the fret. So when calculating the first fret position for a guitar with a scale of 648 mm the fret position is $648 - \frac{648}{\sqrt[12]{2}^1} = 36.37$ mm.

We chose to use an approach with a servo motor[2] inspired by the StrumBot [107]. The advantage of using a servo motor compared to a stepper motor is speed.

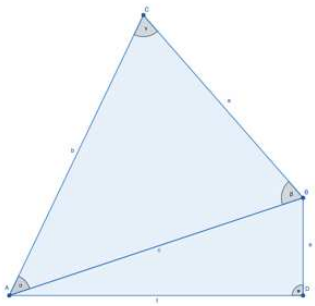


Figure 8.1: Basic geometry sketch.

Figure 8.1 outlines the underlying geometry. Line f from point A to D represents the center of the aluminium rail. Point B is the center of the servo motor's rotation gear. Line a stands for the accordingly attached arm and line b for the arm attached to the carriage system. Using 13 frets an interval of one octave of power chords could be played. So the maximum length of f is calculated as $648 - \frac{648}{\sqrt[12]{2}^1 3} = 314.18$ mm. The arms a and b must be long enough to reach point B. Line e has a length of 36.38 mm. By applying the Pythagorean theorem the length of c can be calculated.

The concrete length selection for a and b is arbitrary, but b must be longer than $a + e$, otherwise β could not be greater than 90°. As the length of c is given, and by using a fixed length for a, applying the law of cosines, b can be calculated when providing the angle β. We used a python script[3] to test multiple values for a and b. Acceptable values would be within a maximum cent

[1] See Strings and Wood website for further information.

[2] We are using a Feetech Servo Motor FT5316M with a torque of 15.5 kg/cm. We did not know the total weight of the solenoid carrier at the point of constructing a first prototype and therefore estimated the torque value. It turned out to be sufficient, although when designing the arms as an isosceles triangle, a servo motor with higher torque will be needed, when β (see Figure 8.1) gets close to 90°.

[3] See our repository on GitHub.

Fret	**Pos.** (mm)	**Actual** (mm)	**Freq.** (Hz)	**Actual** (Hz)	**Cent Diff.**
0	0.00	0.00	82.41	82.41	0.00
1	36.37	36.93	87.31	87.39	1.58
2	70.70	69.79	92.50	92.36	2.71
3	103.10	101.81	98.00	97.77	4.10
4	133.68	132.59	103.83	103.61	3.67
5	162.55	164.55	110.00	110.46	7.16
6	189.79	190.79	116.55	116.80	3.77
7	215.51	213.68	123.48	122.95	7.33
8	239.79	241.77	130.82	131.46	8.44
9	262.70	260.55	138.60	137.83	9.61
10	284.32	283.72	146.84	146.59	2.87
11	304.73	306.18	155.57	156.23	7.30
12	324.00	323.40	164.82	164.51	3.23
13	342.18	343.70	174.62	175.49	8.59

Table 8.1: Fret positions and cent differences for frequencies.

difference of ±8 cents (although that seems a bit strict as the ear tries to correct this differences, as described in Subsection 4.2) to ensure the frequency divergences would not exceed the auditory threshold.

To provide more stability to the attached arms, an offset of 50.0 mm is included in f. Otherwise β could not exceed 90° through the physical barrier caused by the servo mounting material. We choose the values 275.32 mm for b and 206.10 mm for a resulting in a theoretical maximum difference of nine cent, which seemed sufficient (it soon became clear that without advanced servo control those calculated values are not achievable in practice). Table 8.1 shows the calculated positions per fret, the actually achieved positions by the servo motor as well as the correct frequency for the according fret, the actual frequency according to the actually achieved position and the cent difference of the two frequencies. The values in the table are calculated by applying the following equations.

The length of a and b is fixed and β is given as it is set via the servo motor.

α is calculated by applying the law of sines:

$$\alpha = \arcsin \frac{a \cdot \sin \beta}{b}. \tag{8.1}$$

Given α and β, γ is calculated the follwing:

$$\gamma = 180° - \alpha - \beta. \tag{8.2}$$

c is calculated by applying the law of cosines:

$$c = \sqrt{a^2 + b^2 - 2ab \cos \gamma}. \tag{8.3}$$

e has a fixed length of 36.38 mm and a right angle to f, and therefore, forms a right angled triangle with c. So f is calculated by applying the Pythagorean theorem:

$$f = \sqrt{c^2 - e^2}. \tag{8.4}$$

$f - 50.0$ (we used an offset of -50 mm for the servo motor behind the nut or rather fret 0) is the actual position of the plucking mechanism carriage system. An open string's frequency is tuned to a specific value e. g. 82.41 Hz for the low E-String on a guitar. The wave length produced by a string is determined by the length of the oscillating part times two. On an open string it is the scale length l multiplied by 2. On fret x it is scale length l - position of fret x times 2:

$$\lambda = (l - f_x) \cdot 2 \tag{8.5}$$

Wave speed v is calculated by the wave length λ (m) of the open string and the frequency f (Hz)

$$v = \lambda \cdot f, \tag{8.6}$$

the wave speed v is constant for this string. So to calculate the frequency f for a given wave length λ the equation is

$$f = \frac{\lambda}{v}, \tag{8.7}$$

To calculate the difference d of two frequencies in cent, the following equation is applied

$$d = [1200 \cdot \log_2 \frac{f_1}{f_2}]. \tag{8.8}$$

By adjusting β and recalculating the according values a minimal cent difference per fret can and a maximum cent difference for all frets can be calculated as shown in table 8.1.

8.3 Implementation

Our approach basically consists of four main components: (1) an optical pickup, (2) a fretting mechanism, (3) a plucking mechanism and (4) a damping mechanism. As this module based approach does not consist of a resonating body, a sound pickup mechanism is crucial and we started by prototyping the optical pickup. Afterwards we started designing the fretting mechanism. We decided to 3D print most of the parts, as this would allow us fast adaption and necessary improvements. We did not solder any parts unless necessary because of vibration or moving items, but used breadboards instead. For servo motor and solenoid control we used an Arduino Mega Rev. 3 board. The stepper motor was also controlled via Arduino using an additional Pololu A4988 motor driver.

8.3.1 Optical Pickup

Based on Steve Hobley's tutorial[4] on how to build an infrared string bass guitar, we prototyped and reproduced this approach. We mounted six guitar strings on a wooden board (see Figure 8.2), wired two common infrared photo transistors via resistors and capacitors to a low voltage audio power amplifier (specifically we used a LM386N-3[5]). The output pin of the amplifier was connected to a standard 3.5 mm stereo jack of standard active desktop loud speakers with volume control. We used Lego bricks as mounting devices for the infrared photo transistor and the infrared LEDs. As a power supply we used a common 9 V battery.

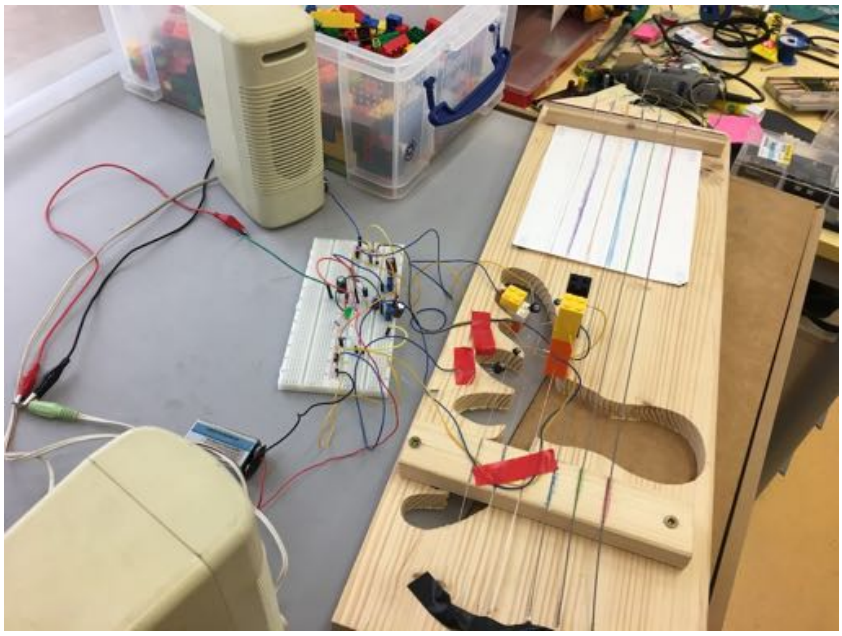


Figure 8.2: Optical pickup. Prototyped on a breadboard using Lego blocks as LED mounting devices, connected to a 3.5 mm stereo jack.

8.3.2 Fretting Mechanism

The fretting mechanism is the core part of the module. It must allow fast and accurate positioning of the carrier to an according fret. The design of the first prototype underwent a long evolution process. We started 3D printing small parts to test our assumptions and immediately redesign the model including improvements, following a rapid prototyping approach. The next subsections show the prototyping process and the finished prototype.

[4]See the article on makezine.com.
[5]See the manufacturer's data sheet.

8.3.3 Evolution

The first approach shown in Figure 8.3 was to mount an arm onto a servo motor, connect a second arm to the servo arm and a carrier to the second one. The servo motor[6] was controlled via Arduino[7] by simply connecting it to the power pins and a digital I/O pin.

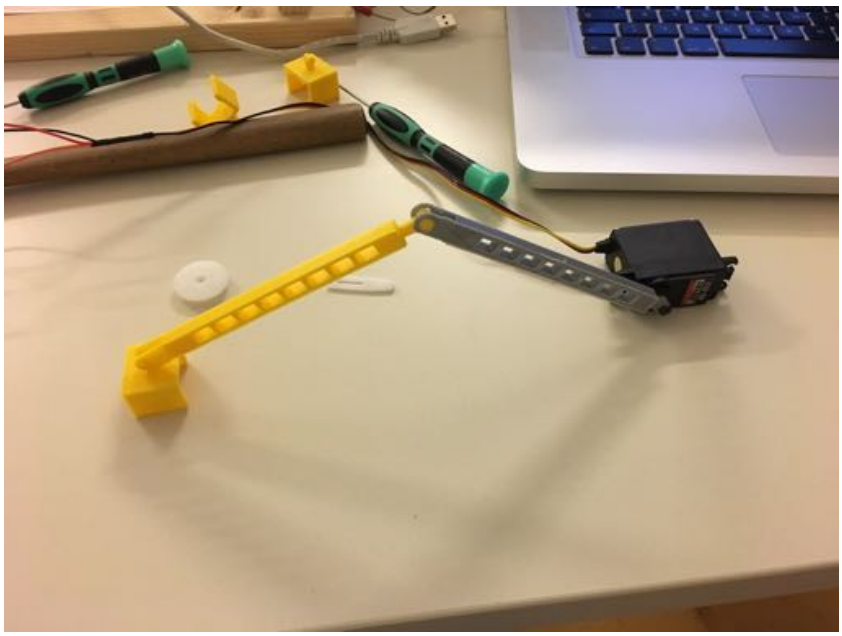


Figure 8.3: First servo arm approach.

While the basic concept was successfully tested, it became very clear that friction is an important factor to consider. Therefore, we decided to reduce it by adding ball bearings to the joints (see Figure 8.4). We mounted the construction onto an aluminum bar with a 20x20 B-type slot 6 profile. Those lightweight bars can be used as guide rails and are easily mountable with common screws and T-nuts to future prototype extensions. Furthermore, we used a high torque servo motor with a torque of 15 kg-cm.

On the carrier we mounted a pull solenoid[8] (see Figure 8.5) to implement the actual fretting mechanism. The solenoid was controlled by the Arduino via TIP120 Transistor[9].

[6] We used a Hitec HS-422 servo motor for no particular reason, but because they were storing in the institutes lab.

[7] We used an Arduino Mega 2560 Rev. 3 board as it was clear from the very beginning, that we are going to need a lot of I/O pins. Furthermore, a servo library is included (see reference in including examples).

[8] We used an intertec ITS-LZ 2560-Z-12VDC pull solenoid. The maximum pull force is 22 N at the last mm according to the data sheet and above 10 N in for the last 3 mm. We measured the weight needed to pull down the low E-string with a common kitchen scale by adding weights to the string still mounted to a guitar. This resulted in a weight of about one kilogram which translates roughly to 10 N for our needs.

[9] We basically followed this instructables guide.

Figure 8.4: Second servo arm approach.

Figure 8.5: Solenoid carrier.

As the servo positioning did not work precise enough, we decided to use frets (see Figure 8.6) allowing a small positioning error.

8.3.4 Plucking Mechanism

We took the plucking mechanism described by James McVay[90] creating the MechBass as a template. It is feasible to use a stepper motor, as fast speed and precise step control is necessary to guarantee a string plucking. We used the Adafruit Stepper Motor NEMA17[10]. It rotates 1.8° per step. So the number of steps per revolution is:

[10]Model XY42STH43-0354A (see data sheet).

Figure 8.6: 3D printed frets.

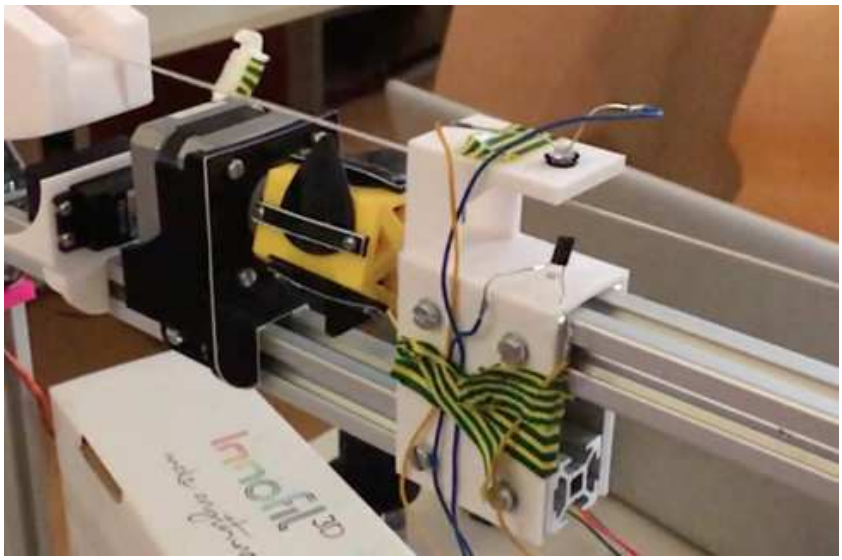


Figure 8.7: String plucking mechanism.

$$spr(\text{steps per revolution}) = \frac{360}{1.8} = 200. \tag{8.9}$$

We chose to attach five plectrums as McVay did. The number of plectrums effects how often the string can be plucked in an according time frame. Additionally, construction issues have to be taken into account and five seems to be feasible amount as the number of steps to pluck a string should be an integer (otherwise microstepping is needed, which is not supported by all stepper motors and stepper drivers). This results in:

$$\text{spp (steps per pluck)} = \frac{200}{5} = 40. \tag{8.10}$$

When designing a plucking mechanism to play strings, timing is an important factor. We are not mechanics or electronics students, so the following concept is simplified. The maximum speed of

a stepper motor depends on four factors: (1) voltage, (2) ampere, (3) induction and (4) steps per revolution[11]. The maximum speed of a stepper motor is calculated following:

$$\text{maximum speed} = \frac{V}{2LI \cdot spr}. \tag{8.11}$$

The minimum time it takes to achieve a single step is:

$$\text{minimum time} = \frac{2LI}{V}. \tag{8.12}$$

Those equations indicate that higher voltage or lower current would fasten up the stepper motor. When supplying power it is easy to regulate voltage or current, but supplying higher voltage to the stepper motor will result in a higher current, therefore some stepper motor drivers are capable of limiting the current. We used a Pololu A4988 stepper motor driver to achieve this[12]. According to our motors data sheet the specifications are 12 V (we are supplying 24 V), 350 mA (0.35 A), 33 mH (0.033 H) and 200 steps per revolution. Speed and time are calculated following:

$$\text{maximum speed} = \frac{24}{2 \cdot 0.033 \cdot 0.35 \cdot 200} = 2.59 \text{ revolutions per second}, \tag{8.13}$$

$$\text{minimum time} = \frac{2 \cdot 0.033 \cdot 0.35}{24} = 0.0009625 \text{ seconds} \tag{8.14}$$

Multiplied by 40 (steps per pluck) this results in $0.0009625 \cdot 40 = 0.0385$ seconds to pluck a string. In comparison 180 beats per minute equals 0.3 beats per second, so the plucking time value seems sufficient.

8.3.5 Damping Mechanism

The damping mechanism consists of a servo motor with an attached arm and foam material to suppress noise when hitting the string.

8.3.6 Prototype

After the individual components were low-fi prototyped and short term tested, we designed[13] a 3D model combining them into one module (see Figure 8.8). Figure 8.9 shows the resulting functional prototype.

[11] We used the following stepper motor calculator to gain a basic understanding of the correlations for those values.

[12] Pololu provides a helpful tutorial on how to adjust the maximum current.

[13] We used tinkercad for providing an online editor supporting collaborative work.

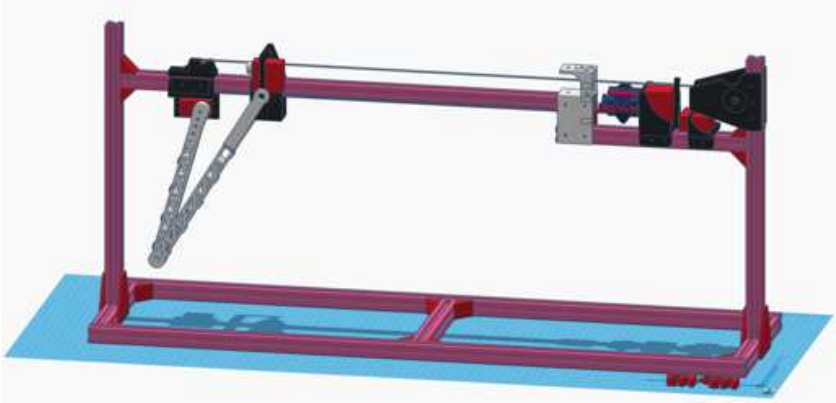


Figure 8.8: 3D model for one string module with the improved servo fretting approach.

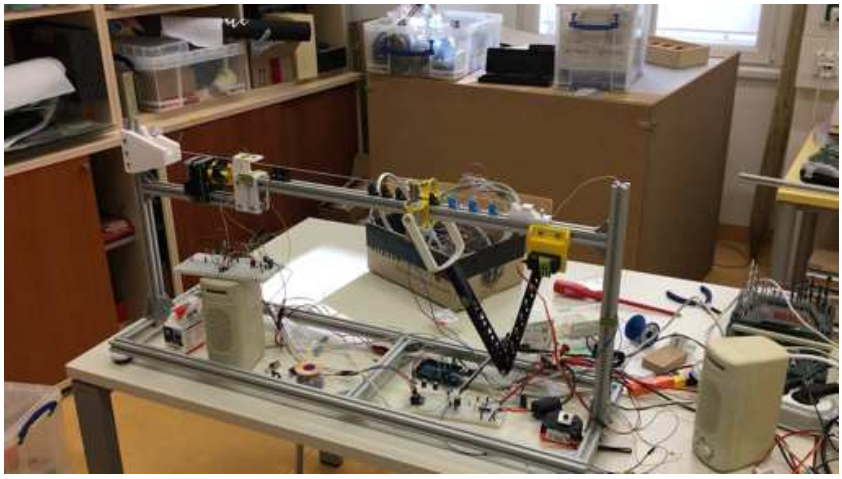


Figure 8.9: Prototype of the first iteration.

8.4 Findings and Improvements

While we accomplished the hard coded play of the well known german children's song *Alle meine Entchen*, it soon became clear that this approach will not be sufficient. The most important issue is the precise positioning of the solenoid between two frets. Only with the Arduino servo library and without additional feedback about the actual carrier position, overshooting caused unsatisfying results. Speed also plays a crucial role. The maximum speed achieved by the servo motor lies at 120 beats per minute, which equals two beats per second. The solenoid became too hot and the 3D printed material (PLA) deformed. The plucking stepper motor does not reliable perform exactly 40 steps without additional feedback. The Damping mechanism works as intended. To attack those problems we refined this iteration, as the basic concept seemed to work, but fine tuning was still needed.

8.4.1 Feedback Positioning System

We designed a simple feedback system consisting of multiple infrared LEDs (one per fret) and an infrared photo transistor (see Figure 8.10). A digital 16 channel multiplexer[14] was used to control the LEDs to spare I/O pins on the Arduino. The concept of this positioning system was that one infrared LED is linked to a fret. The carrier containing the photo transistor sends the feedback if it reached a position underneath a LED. Therefore, via multiplexer exactly one LED was turned on and the servo moved in the according direction as long as a given threshold for the analog transistor signal was reached. It is basically an advanced light barrier. This approach has an impracticable disadvantage of being too slow. Therefore, we came up with the idea of calculating an adjacency matrix.

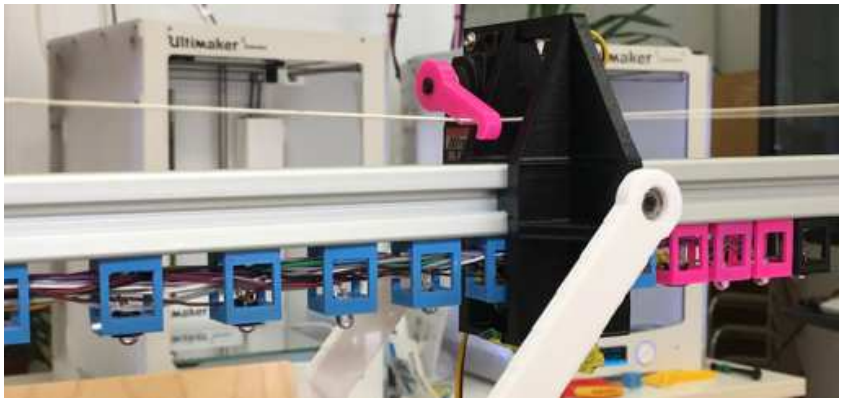


Figure 8.10: Infrared LED feedback positioning system and servo motor replacing the pull solenoid.

8.4.2 Adjacency Matrix

Incremental servo position setting takes too much time, so we calculated a matrix (see Table 8.2) slowly moving to each fret and from this fret jumping to every other possible position looking for the other frets. If the value of the light barrier is above a given threshold, we assume that we found the according fret and now know the servo position in micro seconds to set when jumping from fret x to fret y. This basically solved the positioning problem, although every now and then overshooting occurred.

8.4.3 Solenoid Replacement

According to the data sheet the solenoids can reach a maximum temperature of 130° Celsius. This seems to be enough to deform the print material (PLA). Therefore we decided to replace the solenoid with an even cost efficient component, a servo motor[15] with an attached arm to push down the string (see Figure 8.10).

[14] We used the the Sparkfun CD74HC4067.

[15] We again used the HS422 servo motor, which provides a torque of 3.2 kg-cm.

Frets	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	181	189	196	202	206	211	216	220	224	228	232	235
2	167	0	186	196	202	207	211	216	219	224	228	232	235
3	166	178	0	194	201	207	212	217	220	224	228	232	235
4	165	177	186	0	200	207	211	217	221	224	228	232	236
5	166	177	185	195	0	206	211	216	220	225	228	232	235
6	168	177	185	195	201	0	210	213	221	225	229	233	235
7	168	179	185	194	200	206	0	215	221	225	229	233	236
8	168	180	186	194	200	206	211	0	220	222	229	233	236
9	169	180	187	192	200	205	211	216	0	224	230	234	236
10	167	180	186	193	199	206	211	215	220	0	229	232	236
11	167	180	186	194	199	206	210	214	221	224	0	234	236
12	166	178	186	193	199	203	211	215	220	222	229	0	236
13	167	178	187	193	200	204	211	215	219	222	229	232	0

Table 8.2: Adjacency Matrix. Values are in tenth of micro seconds.

8.4.4 Maximum Speed

This is determined by the distance between the first and the thirteenth fret, the length of the servo arm and the carrier arm and the servo motor specifications. The distance between the first and the thirteenth fret is 305.82 mm. The total radius for β (angle adjustable by the servo motor) is 88° to cover this range. According to servo motor specifications the time to perform a 60° rotation is 160 ms at 6 V. So to perform a rotation of 88° it takes:

$$\text{time for } 88° = \frac{160}{60} \cdot 88 = 235 \text{ ms} \tag{8.15}$$

Ignoring the time for plucking and damping the string this equals 4.2 beats per second or 255 bpm. This seems to meet our requirements, but here physics comes into play and tells us otherwise. In on-load operation we only managed to achieve a speed of 400 ms from first to thirteenth fret. Combined with the plucking and damping time this results in about 500 ms per beat or 120 bpm. This problem can only be solved by a servo motor with a better performance (higher speed or torque).

8.4.5 Plucking Consistency

To ensure consistent 40 steps per string plucking, we decided to use a light barrier. For this iteration at this point it was clear, that we need to redesign the whole fretting mechanism, and therefore this task was shifted into the next iteration.

8.4.6 3D Printing and Rapid Prototyping

3D printing to support rapid prototyping turned out to be quite efficient. Platforms such as Thingiverse[16] provide a great variety of models, easily adaptable, and usually published under a creative-commons license. We redesigned our 3D module shown above in every of the following iterations, which turned out to show potential weaknesses or design errors before actually constructing parts.

[16] See thingiverse.com.

CHAPTER 9

Second Iteration

Author: Raphael Kamper

The main issue with the first iteration's design was speed. Therefore, we decided to redesign the fretting mechanism, specifically the carrier positioning. We replaced the servo motor and its attached arms by a belt driven stepper motor approach. All other components remained untouched during this iteration.

9.1 Underlying Principles

The actual fretting mechanism stayed the same, a servo motor pushing down the guitar string, but the carrier was redesigned to be attached to a belt. A gear is mounted onto the stepper motor driving the belt and moving the carrier accordingly.

We already know how to calculate the stepper motor speed from the first iteration (see Equation 8.11). When the stepper motor rotates the mounted gear, the belt moves a certain distance depends on the number of tooths and the tooth width. We chose a common GT2 belt[1]. We approximated the gear's shape as a circle with the circumference (C) given by the number of tooths and their width:

$$C = \text{number of tooths} \cdot \text{tooth width}. \tag{9.1}$$

The gears radius (r) is:

$$r = \frac{C}{2\pi}. \tag{9.2}$$

The distance each step (1.8° for our motor) causes the belt to move is calculated the following:

[1] Specifically a GT2 belt 6mm wide

$$\text{step distance} = \frac{2r\pi}{360} \cdot 1.8. \tag{9.3}$$

Step distance must be smaller than the minimal fret distance ($f_{13} - f_{12}$) which is 18.19 mm. The total distance between first and last fret ($f_{13} - f_1$ is 305.82 mm (Neglecting all spatial restrictions, the ideal gear supports positioning between all frets with the least amount of steps [2]). This leads to a theoretical gear with 1818 tooths and a radius of 578.69 mm. While a stepper motor with this gear attached could manage to move the belt in 0.016 s over the total distance, this is clearly impractical. Therefore we decided to print a standard T100 gear with the following dimensions:

$$C = 100 \cdot 2 = 200 \text{ mm}, \tag{9.4}$$

$$r = \frac{200}{2\pi} = 31.83 \text{ mm}. \tag{9.5}$$

This leads to a step distance of

$$\text{step distance} = \frac{2 \cdot 31.83 \cdot \pi}{360} \cdot 1.8 = 1 \text{ mm}, \tag{9.6}$$

and simplifies the verification of the carrier positioning. We used the same stepper motor as in the first iteration, therefore all specifications and resulting values such as minimum step time remain. The time it takes to move the carrier over the total distance from fret f_1 to f_{13} can be approximated following:

$$\text{steps per total distance} = \left\lceil \frac{\text{total distance}}{\text{step distance}} \right\rceil \tag{9.7}$$

$$\text{time per total distance} = \text{steps per total distance} \cdot \text{minimum step time} \tag{9.8}$$

Those equations neglect an acceleration factor that is needed when driving stepper motors, because starting a stepper motor at full speed in on-load operation has a high failure potential, but provide a first reference value. Given that the minimum step time (see Equation 8.14) is 0.0009625 seconds this results in:

$$\text{steps per total distance} = \left\lceil \frac{305.82}{1} \right\rceil = 306, \tag{9.9}$$

$$\text{time per total distance} = 306 * 0.0009625 = 0.294525 \text{ seconds}. \tag{9.10}$$

[2] We wrote a python script to calculate specific gear configurations.

9.2 Implementation

As already mentioned we used the same stepper motor setup for the carrier positioning as we used for the plucking mechanism during the first iteration. A T100 3D printed gear is mounted on the stepper motor and a GT2 belt is attached. On the opposite site a T20 gear was mounted on a 3D printed fixture with a M3 screw (see Figure 9.1). The belt tension is controlled via a 3D printed belt tensioner[3] and the carrier is attached to the belt by a clamp. We relied on a software library to control the stepper motor acceleration[4]. Still at high speeds positioning errors occurred. Despite a push button for calibration to find the first fret we implemented a feedback positioning system to determine the exact position of the carrier.

9.2.1 Feedback Positioning System

Rotary encoders are suitable for this specific problem. A rotary encoder basically creates grey code[5], so it's possible to determine the rotations direction and, depending on its resolution, the rotation angle. Most common rotary encoders are limited to operating below 60 rpm. In our setup the minimum step time of 0.0009625 s with 200 steps per revolution leads to

$$\text{revolutions per second} = \frac{1}{0.0009625 * 200} = 5.1948, \tag{9.11}$$

which translates to approximately 312 rpm ($5.1948 \cdot 60 = 311.68$). In order to still allow the usage of a common rotary encoder, we modified the gear transmission ratio by adding a T16 gear, driving a T100 gear (see white gear in Figure 9.1) with the attached rotary encoder. The gear ratio factor is now $\frac{16}{100} = 0.16$ and applied to the rpm results in approximately 50 rpm ($311.68 \cdot 0.16 = 49.87$). This reduction of rotation speed brings a loss of the rotations angle resolution. The encoder we used[6] has a resolution of 20 pulses per rotation. Which translates to 40 segments per revolution and a resolution of $\frac{360}{40} = 9°$. The resolution must be below the minimum fret distance of 18.19 mm to confidently determine the carriers fret position. As one step means a rotation of 1.8°, a resolution of $\frac{9}{1.8} = 5$ mm is reached without the modified gear transmission ratio and $\frac{5}{0.16} = 31.25$ mm which is above the minimum fret distance.

[3] See this belt tensioner on thingiverse.
[4] We used the AccelStepper library for stepper control.
[5] See this tutorial for further explanations.
[6] See distributors website for further information.

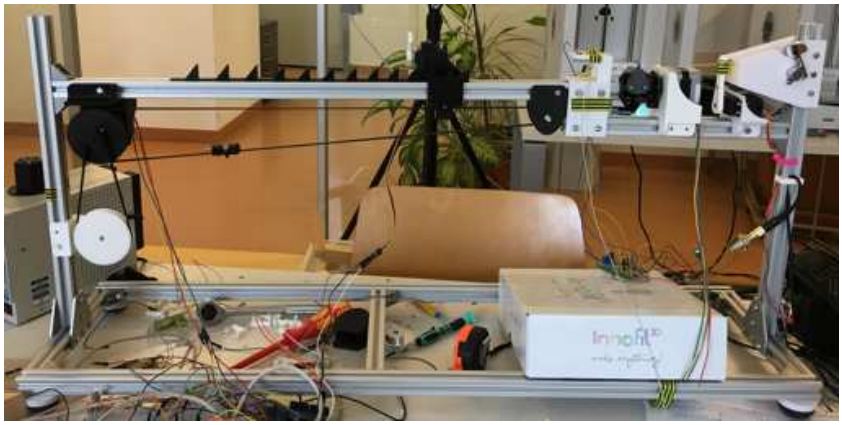


Figure 9.1: Second iteration: stepper motor approach.

9.3 Findings and Improvements

With the T16 to T100 modified gear transmission ratio we were not able to reach a sufficient precision for carrier positioning. Therefore we tried to design a light barrier based step counter (see Figure 9.2) to directly measure the steps. We used a transmissive optical sensor[7] and 3D printed a gear rotating between the sensor.

While this approach worked at lower speeds, we were not able to get sufficient results at the intended operating speed and this iterations approach turned out to be not applicable for our design goals, and does not meet our requirements with respect to speed and precision.

[7]See datasheet for further information.

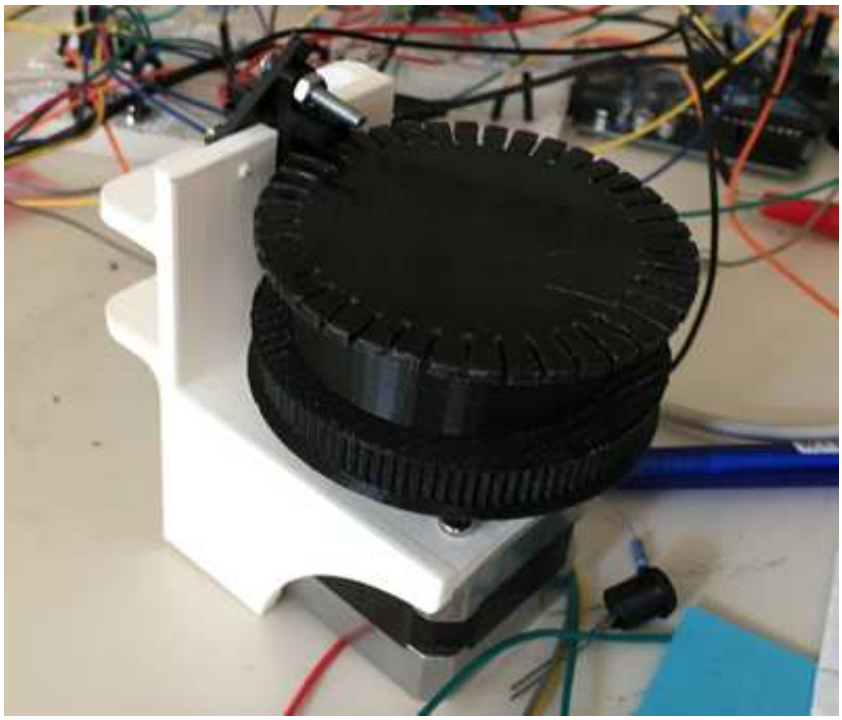


Figure 9.2: Second iteration's step counter.

CHAPTER 10

Third Iteration

After it became clear that, within the limits of our financial possibilities, the usage of servo or stepper motors to move a carrier to a specific fret is not applicable for this work, we decided to rethink the whole fretting mechanism. Based on the knowledge we gained throughout the previous iterations, we decided to replace the carrier approach by an one servo motor per fret approach. This would eliminate precision problems, as no carrier positioning is needed, and increase speed, but hardware costs will rise. We found the MG996R servo motor to be within our financial possibilities.

10.1 Underlying Principles

With a perspective to precision and speed, this is a very simple approach, as we still use the 3D printed frets and position one servo motor per fret (see Figure 10.1).

10.2 Implementation

Despite the redesign of the fretting mechanism (see Figure 10.1), we further improved the plucking mechanism. The damping mechanism remains unchanged, but is now also controlled via servo controller instead of direct control via Arduino.

10.2.1 Fretting Mechanism

One servo motor controls a single fret (see Figure 10.2). A 3D printed servo horn extension is placed on all servo motors. The servo motors are controlled via a Pololu Maestro Mini 18 and 24 servo controller, attached via serial pins to the Arduino.

It takes about 0.06 s to pull down a string and another 0.06 s to perform the damping action. Combined with the 0.04 s to pluck a string this sums up to 0.16 s which translates to 375 bpm.

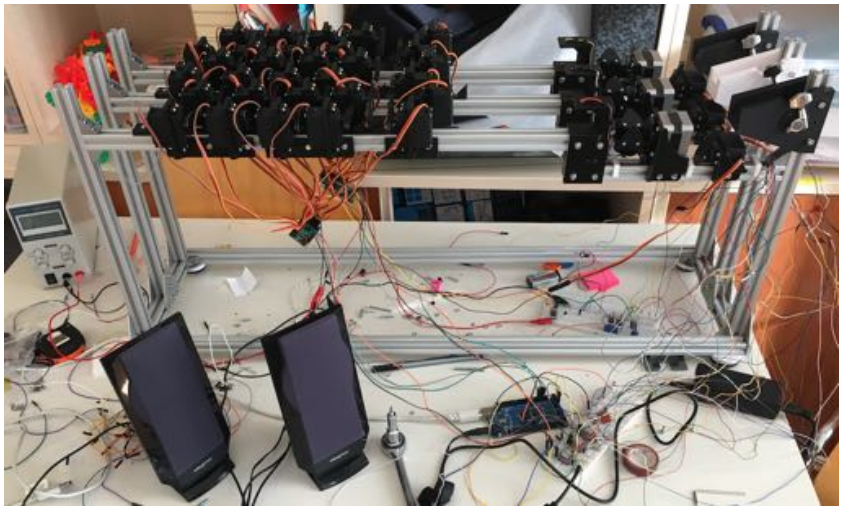


Figure 10.1: One servo per fret approach.

Figure 10.2: Third iteration: One servo per fret approach, servo horn extension.

10.2.2 Plucking Mechanism

Although the self printed light barrier step counter approach did not work as intended in the second iteration, we successfully adapted this approach for the plucking mechanism. A disk with holes for every 40 steps is placed between a transmissive optical sensor (light barrier) as shown in Figure 10.3. The stepper motor operates as long as a certain threshold value is returned from the light barrier.

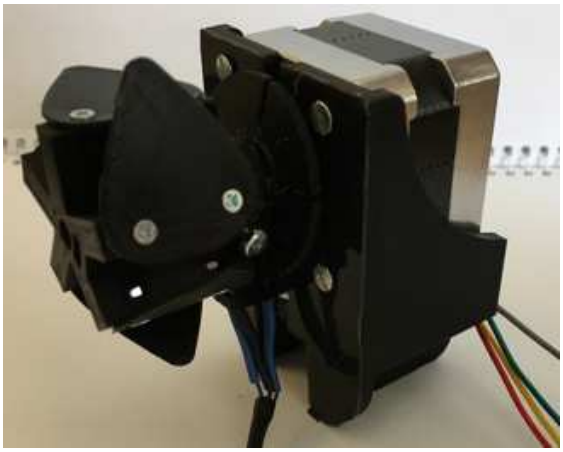


Figure 10.3: Third iteration: Light barrier, stepper motor.

10.3 Software

As it turned out, that this iteration has the potential to fulfill our requirements we needed to design the communication with the user interface. The easiest way to establish a connection between an Arduino Mega and a common laptop is by creating a serial connection. As timing is crucial in musical context, we decided to run all calculations on the user interface laptop and only send commands that are immediately executed on the Arduino. The source code for this iterations can be found on GitHub[1].

10.3.1 Command Message

Our research regarding music interface protocols lead us to the conclusion that they are all overdimensioned for our purpose. Therefore, we decided to define a simple message containing all the values necessary to play multiple strings. A command message consists of eleven comma separated values terminated by a newline character. The structure of the message is:

- message id,
- bpm value,
- fret on string 0,
- string 0 note duration,
- string 0 damping,
- fret on string 1,
- string 1 note duration,
- string 1 damping,
- fret on string 2,

[1] See the source code.

- string 2 note duration,
- string 3 damping.

The Message id is an integer, increased for every sent command and identifies a single command. After executing the command, the Arduino sends back the exact same message. Despite logging issues, this is also used to prevent potential timing issues. In the case of a software error on both the Arduino or the laptop, this could lead to playing false notes in terms of height, duration or start time. This bears the potential of confusing the users, as the secondary auditive feedback (see Subsection 3.2) does not match their input. Therefore we decided to not play a note if something went wrong. One indication is that the laptop wants to send a note, but did not get response message for the previous one. That indicates a timing issue on the Arduino and the note is skipped.

The bpm value is not processed further in the current state, but could be used for a check whether it is below the the maximum possible speed determined by the motors. Setting the tempo is mentioned in Subsection 12.1.13. Appended to the message id and bpm values there are three values per string. The fret on the string, the duration of the tone and if the tone should be damped or not after it was played. The musical user interface is described in Section 12.1.1.

10.4 Findings and Improvements

The approach in this iteration fulfilled our requirements regarding speed and precision as expected. While the theoretical maximum tempo is about 375 bpm, but 350 bpm turned out to be more reliable when testing fast play in on-load operation. The cheap servo motors we used are suitable for the purpose of pushing down a guitar string, but plenty of them had jitter issues and might not be suitable for other purposes requiring higher precision.

There are limitations by the form factor of the motors, the higher the fret the smaller the distance to next fret. This means that the motor must fit in between the distance of the second next fret (as one motor is placed on the left and one on the right side). This would require special and therefore more expensive hardware, or other constructional modifications.

We decided to 3D print all models in black color, as it was advised to us during the expert interviews (see Subsection 11.2). For a user it is hard to tell which fret actually is played by just looking at the sheer mass of black motors. Lightning up a LED on the active motor could support a better understanding of which motors are involved in producing the sound.

Stepper motors create an electromagnetic field. In this last iteration we routed the unshielded cables from the optical pickup underneath the stepper motors. This produced a constant background noise in the audio output via loudspeakers. For future work a proper cable management and the use of shielded cables are highly recommended.

While the first iteration's approach still could be used for accompaniment music, we decided to drop this idea and focus solely on melody composition. This decision was based on our lessons learned so far. We heavily underestimated the sheer complexity of hardware engineering and necessary music theory for algorithmic composition and wanted to the test melody composition isolated from potentially auto-generated accompaniment support. If this approach turned out to be successful, additions can be easily made due to our anticipatory module based design.

Part IV

Practical Part - Musical User Interface

After the practical part about the mechatronical components has been covered, this part is dedicated to the musical user interface, consisting of tangible and graphical UI components (see Section 2.2). It shall also be noted, that the mechatronical part of the project and the MUI part didn't start at the same time (see Figure 7.3). The decision-making process for this process model and a description of the respective design phases and a more detailed description of the methods being used can be found in Chapter 6.

Each chapter consists of one design cycle held. The cycles differ in the applied research methods and thus also in the collected information about the project. Each chapter describes the applied methods in chronological order. The findings and their effects on the further development and design process of the respective method are captured at the end of each chapter under the name *repercussions*. These repercussions can, as soon as the production of design solutions and the evaluation in the design process took place, be seen as the recurrent two phases *understand and specify context of use* and *specify the (user) requirements* from the chosen ISO process model (see Section 6.1). The in Section 7.1 explained working document is the listing of all questions and requirements which took place in the course of the practical MUI part. These questions and requirements are also being updated according to the listed results in the repercussions. The state of the working document after each method can be seen in the Appendix at the end of this work.

CHAPTER 11

First Iteration

Author: Jakob Blattner

As described in Section 7.1, the first step of the first design iteration, and therefore the project, was to get an overview of all related topics, other similar projects, potential problems and possible approaches. For this purpose, a literature search, expert interviews and the creation of personas were carried out. After those methods had been completed, the researchers took care of creating requirements by applying requirements engineering and creating use cases (see Section 7.2). Those methods were followed by the production of a design solution (see Section 7.3), including the usage of sketches, wireframes and mockups. The last step of this design cycle was an evaluation of the created design solutions. The evaluation was conducted in a laboratory, with experts and non-experts as participants. All further information regarding the evaluation phase of a design process can be found in Section 7.4.

Before continuing with the description of the literature search, the expectation and knowledge of the authors before the start of this thesis will be described. Before this work, a project was carried out within the frame of a course at the Technical University of Vienna, from which this thesis arose (see Subsection 2.3.9). The *Loop* was a musical user interface with a token and constraint approach (see Section 2.1.2) where music could be produced on two guitar strings (still attached to the guitar itself) by placing and turning wooden blocks in predefined positions. From this project, certain requirements for the non-mechanical part of this MUI became obvious:

- The lack of visual and haptic feedback was hampering the understanding of the exact function of the user interface. This meant, that the lack of clear visibility of the system status (see 3.5.2) had to be fixed, and the haptic feedback had to be optimized.
- The already mentioned input method limited the user in his/ her interaction with the system and gave the user a restricted feeling of his/ her possibilities. As a result, the token and constraint approach had to be changed or needed to be optimized.
- Another problem was an already documented disadvantage of TUIs. The willingness of users to interact with the prototype was very limited. For another prototype, special attention should be paid to possible improvements to reduce this problem.

- As soon as an interacted with the Loop took place, many users quickly came up against a limit. The creation of a melodious music exceeded the limits of what was possible for almost every user. As a result of this, the creation of such melodious music should be supported by the prototype to come.

Additionally, the authors predefined a non-collaborative approach for this project, where the main target group are non-musically trained people (but musicians are not excluded). Furthermore, the dedicated environment in which the installation will be used, has been defined as a museum or other comparative environments. These requirements, which will be described in detail in section 11.4, were the basis of this project and were extended in the course of this work. After the origin of this basis has been explained, the following section begins with the first conducted research method: the literature research.

11.1 Literature Research

A comprehensive research was conducted on the topic of musical TUIs and the possible existence of a similar project (and their further investigation), followed by other relevant topics such as music theory, interaction possibilities, design processes, guidelines, etc. Since the description of these subjects has been covered in the theoretical part of this work (see Part I), this section defines the outlines of the project and their decision taking process based on the knowledge gained in the literature research. Starting with the dis- advantages of different user interface types discussed in Section 2.

11.1.1 Dis- Advantages of Graphical and Tangible User Interfaces

The interface which will be created in the course of this work can profit from nearly all benefits that come with GUIs. One exception is the low anxiety of users when interacting with GUIs. This is due to the combination of TUIs and GUIs, where TUIs neutralize this positive effect. From the advantage of immediate feedback and all advantages which stay in contact with visualization this project benefits the most.

Regarding the disadvantages of GUIs, the high chance of confusion for users outside the user group is not applicable due to the fact, that the UI is designed for people without any previous knowledge of music and music theory. The inefficiency for expert users can be left out, for the target group are non-musically trained people and therefore non-experts. Two further disadvantages, both relating to symbolic representations of the UI, can be prevented, or at least can their chance of occurrence be greatly reduced, by an iterative design process. With this, icons, symbols, and the design in general can be tested, analyzed, evaluated, and if necessary changed in each design circle. The last disadvantage, the lack of existing guidelines, is especially the case with this project. This can be explained with the very specific purpose of this installation. All other GUI disadvantages also affect this project. The few guidelines which could be found during the research, are recapitulated in Subsection 3.4.2.

Unlike the advantages of GUIs, not all the advantages of TUIs affect the project done in the course of this work. The positive impact on collaborative interaction will not be advantageous, due to the already defined requirements which state, that the resulting application will not be designed for more than one user. Another not applicable advantage is the extension of an environment and objects the user is already used to. The target group of this work is, apart from the fact that they must be non-musically trained people, too freely defined to limit oneself to. Thus,

the environment (like the defined musem) in which the installation will not be known to most of the users and does therefore not enhance an already known environment, as possible with TUIs. The objects that will be used for the installation are developed especially for them, the probability that one of the users knows a similar object that is not used for the installation is therefore extremely unlikely. The advantage of not needing continuous eye contact from users, can only be applicable to wearables and other TUIs which use the whole body as an input device, like the MusicJacket in Subsection 2.3.7. Only if the resulting installation follows one of those two approaches, said advantage will apply. Although half of the positive aspects of TUIs can not exploit their advantage, the other half has at least the same importance. This concerns an intuitive interaction, the possibility to manipulate and understand physical representations, and the understanding of the underlying abstract concepts. The difficulty of designing those physical representations is reflected by the disadvantage of possible misinterpretations of them. This disadvantage, combined with the TUI's possible low efficiency and the low acceptance of users, must be prevented through analysis and evaluation of each of the MUIs design circle. The bad portability, possible mechanical failures and high development costs must be compensated by the remaining advantages.

The following subsection highlights the most interesting properties of the fount related projects in the course of the literature research.

11.1.2 Related Projects

The projects in question are being described in Section 2.3. These projects have been used to define different properties of MUIs in Section 2.2, which will now be used to define the future result of this project, highlight and exclude different possibilities and extend the current requirements. It should be mentioned in beforehand, that the described MUI properties are closely linked together, so that choosing one property, can automatically disqualify another.

Additionally to these properties, this subsection shall also highlight interesting components of the related projects for possible future use in the context of this thesis:

- The implementation of the loop functionality and the visualization of active positions by the BeatBearing [42]
- The extensive positively used visual feedback by the reacTable [29]
- The way of changing settings by the MusicCube [43]
- The usage of icons from the BlockJam [31]
- The abstract use of tones and other musical parameters by the audio D-touch [40]

11.1.3 Implementation Type

Since this thesis does not extend an existing instrument, this form of implementation can be removed.

A further distinction is not as easy and is directly related to the interaction type of the MUI. The authors of the thesis excluded the toy-approach, although the playful approach (see Subsection 3.3.2) is a very important design attitude for this work. The result of this work is not intended to serve as a toy or to be perceived as such, but to stimulate the user's curiosity about music and

creativity. Seeing the future MUI as a toy might result in a different perspective from the user, which in turn devalues the installation in its purpose.

Blocks which can be connected with each other to generate music, like the Block Jam [31], are difficult to be design for a museum context, since it can bes assumed that the blocks will get lost sooner or later. Orientating itself towards the distinction of TUI input methods according to Ullmer et al.[22], the same argument disqualifies the *constructive assembly* approach. The *token and constraint* distinction was used, as already mentioned, in the previous project to this thesis, the Loop. The described disadvantages of this input method, the rather limited possibility of interaction, was a disadvantage that can be avoided by choosing the tabletop or, according to Ullmer, the interactive surface approach. This approach has further advantages, such as a large area where visual feedback can be shown and the use of, relative to blocks, cheap TUI tokens. In addition, by consciously designing the appearance of the installation, the initial interaction of potential users can be promoted.

11.1.4 Interaction Type

The chosen tabletop approach from the previous subsection limits the interaction type of this subsection to the finger and hands and automatically disqualifies the whole body interaction. But there are further concerns that make the chosen interaction method appear the better choice. Haptic feedback, for example, is difficult to achieve when using whole body interaction, except when choosing wearables, instead of e.g. the Microsoft Kinect[1], achieved with wearables, which bring new challenges with them. With wearables the system needs to be adaptable to different body sizes of users, potential hygiene problems, a bigger space for interaction is needed, and an increasing complexity of mapping musical parameters to the users body arise. Furthermore, it can be assumed that a whole body interaction in a public space further hinders the willingness of potential users to interact with the system, as they possibly don't want to appear ridiculous to other people. The system must also provide the user with assistance in creating melodious music. Without or only with limited haptic feedback, this is harder than with it.

The interaction type also raises the question of which music parameters can be adjusted by the user and how the system can assist the user with certain input parameters such as pitch and tone length through restrictions and/or assistance. Possible solutions to this problem will be discussed throughout this chapter.

After highlighting the interaction types, the following subsection highlights the two different ways of generating sound.

11.1.5 Sound Generation

Sound can be produced either digitally or analogously. In the context of this work, the authors decided to use analog sound generation. Although this involves a considerable amount of extra work, the analog sound generation is intended to make the user feel more connected to the melody she/he has created and thus will generate more interest in music itself. In addition, what is heard can be traced back to what has been produced (with the help of additional feedback), which may promote even more curiosity and support the effects of the applied playful approach.

[1] https://developer.microsoft.com/en-us/windows/kinect

11.1.6 Motivation

As already mentioned several times in this chapter, the interest and pleasure of the non-musically trained user is to be encouraged through the interaction with the system. The playful approach is a very important approach in this context, as not only knowledge (which is not the main focus of this work) but also fun and enjoyment are encouraged through playfulness. These emotions are in turn associated with music, which is exactly the goal of the system and motivation of this work.

11.1.7 Target Group

The target group is directly connected to the motivation and the sound generation of this work, as the main target group is defined as non-musically trained people. This definition does not exclude musicians, but is the behavior of the system and the complexity of the input designed for non-musicians. It should be mentioned that the term non-musically trained people does not necessarily mean that a person has no experience with music at all. Since everyone has been to school and there has come into contact with music and music theory, the term non-musically trained people is somewhat variable. There are no further restrictions for the target group such as age, gender, occupational group or similar, which makes the target group very diversified. This diversification can prove to be a difficulty in the course of the thesis, as narrowly defined target groups are, as soon as their , easier to satisfy.

11.1.8 Collaborative Approach

Since the previously defined properties of the system to be developed already involve a great complexity, a collaborative approach has been dispensed with. With regard to the reactable[29], it should be noted that the round shape of the table invites collaborative interaction. Thus, a round form must be avoided.

This also illustrates the importance of physical properties of the installation, which are not only limited to collaborativity. These physical properties are also not limited only to shapes, but also to surfaces, materials, etc. The choice of the right physical characteristics can therefore strongly influence the success of the installation and, among other things, positively influence e.g. the fear of the initial interaction from possible users.

The following section contains the documentation of the expert interviews, for the execution of which the acquired knowledge of this phase was a prerequisite.

11.2 Expert Interviews

Author: Raphael Kamper

We conducted four expert interviews, with three experts teaching at the Technical University of Vienna in the fields of Human Computer Interaction, Interface and Interaction Design as well as Musical Interfaces and Generative Music. One expert works at the University of Vienna and developed a new music instrument. The interviews lasted between forty minutes and one hour. Every interview started with a short introduction on our topic, including the presentation of a ten second video, showing our current prototype to provide the participants a better understanding of the otherwise very abstract description of a guitar robot.

Accordingly to our interview guideline we will outline the core messages gained from the conducted interviews, also quoting the participants if their statement is of special interest for our work.

Important thoughts, suggestions, and comments expressed by the experts are summed up in the penultimate subsection. The last subsection contains the repercussions of the design process. At the this stage of the project, this means that the most important statements by the experts will be highlighted including the opinion of the authors to the respective subject.

11.2.1 Music Theory

The use of musical scales like the pentatonic or major scale is an option to determine a set of valid tones to support the user in creating good sounding music[2]. For accompaniment music it is a common practice to either use a fixed set of previously composed samples (and transpose them if needed) or restrict the possibility of chord composition based on the used musical scale. The concrete idea of using the three deeper strings of a guitar to play chords and the three higher ones to create a melody was considered a valid approach.

11.2.2 Algorithms and Protocols

The MIDI protocol is a widely used and established standard, but for our purpose OSC could be an option due to higher flexibility to create self defined messages fulfilling our needs.

> "But you're using OSC? [...] If you know the MIDI protocol, then it's no problem. [...] Then it's more simple to use OSC [...] you're putting together your own messages." (Expert Interview: Expert One, 08:02 - 08:46 - translated from German by the authors)

> "[...] Well, MIDI, it's obviously, it is a super simple, well defined interface. OSC would maybe provide the possibility to directly generate something like control messages [...]" (Expert Interview: Expert Four, 17:50 - 18:30 - translated from German by the authors)

Algorithmic composition is a very complex topic. Suggestions were made from using previously composed samples, using statistical methods and machine learning approaches. One simple method mentioned for both, melody and chord composition, was the Markov model.

11.2.3 Instrument

Context is one of the most important things to consider when designing the instrument and the user interface. It depends whether the installation will be used by multiple persons in a museum or will be used by a single person during a life performance. The requirements for the interface will be different in the mentioned situations. When designing the interface for many users with different knowledge and expectations e. g. in a museum, it is crucial to decide if the user should be able to imitate a specific melody, or to compose music in an explorative way, while figuring out how the interface works.

As our mechatronic construction does not have a resonating body, and we are using an optical pickup, the acoustic signal needs to be amplified and played back through a connected loud

[2]The definition of good music of course depends on the user's perception, but he or she will probably have an expectation about how a guitar sounds, and further will be used listening to music composed under the application of classical rules for harmony and music theory.

speaker or headphones. The views on the usage of speakers or headphones are controversial. This depends again on the installation's context. If the installation could interfere with a different one e. g. in a museum, headphones are necessary. Three out of the four interviewed experts suggested headphones, as users are probably more likely to interact with the interface if there is no fear to fail in front of an audience. One participant argued that as our installation will not be known to the users and a potential audience, there is not going to be a fear to fail, keeping users from interacting with the audience.

11.2.4 Users and Interaction

In a museum with many different exhibition pieces the time a user is willing to spend interacting with a single installation is limited. Estimates of an average interaction time can't be made, but one expert who already organized musical events, suggested a reference value of three minutes per installation. If a user is not able to understand the interface within this timespan, chances are high, that he or she gets frustrated and stops interacting. This implies the necessity of a intuitive, easy understandable interface. Also the maximum time spent plays a role. If a user spends 15 minutes in a museum with one installation, which can be considered a lot of time, he or she should experience a sense of achievement.

One problem that might occur, this again depends on the museum's context, is that usually you learn from early childhood not to touch anything in a museum. Headphones could give a hint to the visitor that he or she is allowed to touch the installation and additionally minimize the fear of failing in front an audience. The instrument should not start playing by itself to gain attraction, because this could make the interface harder understandable.

When designing for an installation in a museum classes of school children have to be taken into account. This means that the installation must be robust enough to deal with a not so careful handling. Additionally, if for instance tokens are used, there is the possibility of theft of small items belonging to the interface. One possibility of figuring out different interaction forms could be testing with extreme users to also address users not in the target group of museum visitors.

11.2.5 Usability and Input Methods

All input methods need to be evaluated through user testing. There are no specific schemes or guidelines for musical user interface design like when to use a slider, or a potentiometer. One expert argued that tangible user interfaces could simplify the understanding of the interface in comparison to whole body interaction through touchable objects:

> "No abstraction step, we want a representation [...], of what we do, to touch and give them a meaning, a tennis racket for instance [**note:** referring to the Wiimote], or also as an abstract metaphorical thing, where I can try what to do with it, but just the hand is deficient I guess." (Expert Interview: Expert Three, 05:51 - 06:14 - translated from German by the authors)

Another expert suggested using both hands for the interaction, because usually people are really good at coordinating their arms and fingers and are capable of much more than just using a touchscreen. Also most instruments are played with both hands. This argument was confirmed by a second expert who also suggested using game controllers:

> "[...] most of them going into this kind of museum, played computer [...] this is something with both hands, relatively independent [...] maybe this would be the best interface for you. A game controller. Everyone knows how to hold it [...]" (Expert Interview: Expert Three, 06:43 - 07:10 - translated from German by the authors)

Another interesting comment came from one of the interviewees when talking about the accompaniment. To set the accompaniment, the expert suggested a knob, where the accompaniment chord can be set. To give the user additional feedback and a possibility to remember a previous settings, the knob should display the base tune of each settable chord next to the definable knob position.

11.2.6 Playful Interfaces

In principle when playing a music instrument, the player or user, creates challenges by him- or herself. This means that, as the user tries to create new sounds, harmonies or melodies, those attempts could already be considered as challenges. Additionally there lies a danger in adding a game on top of a music instrument, to make the whole installation just a rhythm game for instance.

11.2.7 Feedback

Feedback is crucial in a musical interface. It should be as clear as possible how the interaction with the user interface correlates with the manipulation of sound parameters. One expert stated:

> "[...] especially if you want to define something like a melody, let's say with three stones on a table. Then it must be, in my opinion, every time in interactive cases, guarantied somehow that it's as simple that the person is able to associate [...] somehow after a certain time [...], because only then I can start to plan actions, if I do this, then maybe this is going to happen. I think this is most important at all in such scenarios." (Expert Interview: Expert Four, 08:01 - 08:48 - translated from German by the authors)

The auditory feedback from the instrument itself must be the primary feedback, but other form, like visual feedback may also play an important role. In music instrument design additional feedback is given by design, but in our case it should be seen as an advantage to define the feedback in a decisive way. Design is always authoritarian, so sometimes it is necessary to just implement ones thought of how it should be and evaluate the approach.

Besides the direct acoustic feedback of the sound being played, there is also additional feedback for suggestions or errors. Error here does not mean the playing of the user, but the handling of the interface or failure of the mechatronic part. A musical interface should avoid additional auditory feedback, and the feedback should not be intrusive:

> "[...] insofar, an interface can of course allow to give feedback, but, as always, an interface should just sound in a way, that it does not divert attention of being an interface. This is valid for every, probably forever [...]" (Expert Interview: Expert Three, 15:50 - 16:15 - translated from German by the authors)

11.2.8 Additional Notes

Color could play an important role in the mechatronic design as well as in the interface. The mechanical part should not contain many different colors, as there is a chance that users might try to give a special meaning to a color, which may lead to confusion regarding the interface. Mapping of color and music e. g. dark red is C3 with a frequency of around 130 Hz is scientifically not accurate.

One expert suggested a play and record mode, whereas the user can first record specific sounds and play them back later. Whilst the sound creation process direct acoustic feedback is provided to the user. This means that every manipulation of the interface immediately leads to playback of the corresponding sound in the record mode.

Two experts suggested a loop mode with one expert formulating very concrete ideas and sketching the interface (see Figure 11.1) for better understanding:

> "And, that one quasi has for instance those four tones and loops so to say over them, then one has quasi a loop, that is somehow recognizable, it is immediately clear, if I push this up, then the according tone changes quasi. This would be for instance a really cool start I think." (Expert Interview: Expert Four, 09:11 - 09:25 - translated from German by the authors)

The expert later refined his idea:

> "If one now really has four stones and that is somehow [hums four tones: low - high - low - high], then you loop quasi [hums the same melody again] and you invite quasi, because this gets boring fast, you quasi enforce the interaction at some point." (Expert Interview: Expert Four, 13:04 - 13:17 - translated from German by the authors)

The time axis for the loop interval could also be defined by stones, that are dragable from left to right, marking the beginning and the end of the current interval. In this scenario melodies would be always created by the user him- or herself. A possible harmony voice could be generated using a Markov model considering the melody as an input.

11.2.9 Repercussions

All of the following repercussions will be included when creating the requirements and use cases (see Section 11.4 and 11.5).

The suggestions concerning the accompaniment and the chords are taken over for the further course of the work. Now the question arises which chords should be used for the accompaniment. Ideal would be chords that match any user input, avoiding dissonance. Also, adjusting the chords with a knob and displaying the chords' root note will be kept in mind. Also, using three strings of the guitar for the users' melody and three for accompaniment was deemed to be a valid approach by the experts.

Since there were different opinions with the headphones, the authors have to decide on an approach. The choice fell on the use of headphones. For the authors of this work, it sounds understandable that if you can not play an instrument, it quickly deters audible failures from the

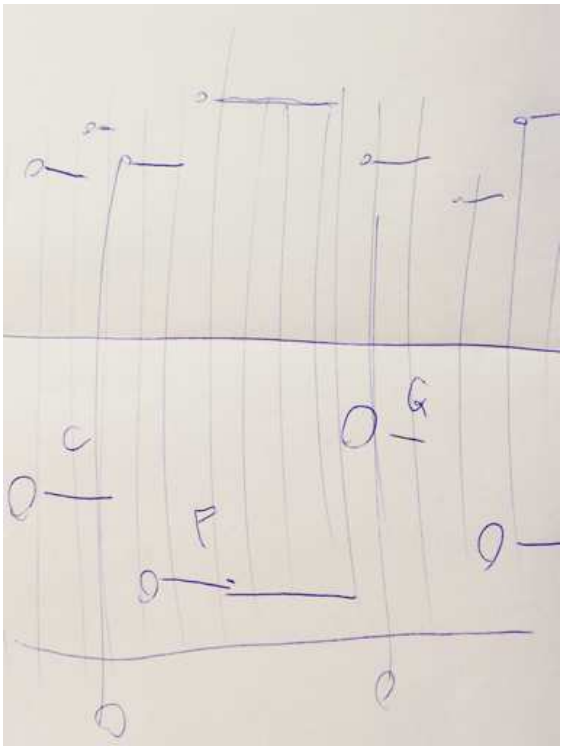


Figure 11.1: Sketch of Expert Four.

environment. Furthermore, distractions from the environment thanks to the headphones are not perceived as strong and the environment is again also not disturbed by the installation.

Regarding the suggestions, the recommended Markov model is noted for later research. However, the question arises, how complex such an implementation is.

The amount of time users spend on installation is between three and fifteen minutes. As a further benchmark for later evaluations, the goal can be set that the user must understand the interface in no more than three minutes. The circumstance, that the interaction time itself can be seen as quite short, doesn't represent an expected problem, as the design attitude playfulness (see Subsection 3.3.2) reduces the time needed to achieve the wanted result. In this case: the creation of a good sounding melody.

Due to the fact that the loop approach was proposed by two of the four experts, the authors will focus on this approach. Regarding the input to adjust the loop range, one of the experts also suggested tokens. A possibility that could possibly be used later on in the work. The chosen form of input through tokens was indirectly confirmed by an expert, as he considered the whole-body interaction to the project aggravating and therefore suboptimal. In another feature of the tokens, the experts also confirmed our assumptions that tokens should be easy and inexpensive to replace.

The feedback of the installation is of course very important. Apart from the output of the user-generated melody, installation errors and possibly suggestions, auditory feedback should be avoided.

Not only the visual feedback but also the chosen colors play an important role in the whole

installation.

11.3 Personas

This section contains the description of personas described in Section 7.1. Since one persona per user type represents the absolute minimum, six personas are described here due to the broad target group. All images seen in this Section have been found on Pexel[3].

Lance Strong

Lance is a 52 year old company owner who, partly because of his profession, is a perfectionist, which stays in direct connection with his workaholic existence. In the few free time he has, Lance does something with his children. After his divorce this means that he and his children usually do something child-friendly on weekends. Mostly in the city to be able to get to the office quickly in case of an company related emergency. Lance has no idea about computers or technology. As soon as he doesn't know his way around something or there happens to be some technical problems, his assistant or the IT department are the ones to whom he turns. Lance doesn't play an instrument but loves classic rock, because it reminds him of his youth.

Jackie Stew

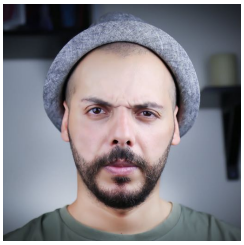
Jackie is a 16 year old student whose greatest hobby is skateboarding. He is a somewhat rebellious young man, who often has too many mischief in his head. He gets bored quite quickly and causes his parents and teachers a lot of trouble. His parents try a lot to open up all possibilities for him, especially since they had a lot less in their childhood. He tries to impress his classmates with his brand clothes and his rebellious appearance. Jackie has, apart from music lessons, little connection to music. In his spare time he mostly listens to rap and uses his skateboard.

Astrid Oakgrem

Astrid Oakgrem is 44 years old and a proud real estate agent, a profession she has only achieved through hard work and a lot of effort. She is very successful in her job. She met her husband at a business dinner and since then they have been running her company together. Astrid attaches great importance to correct behaviour as well as a well-groomed appearance and good behavior. In her spare time she likes to do something with a small circle of friends. She is not interested in music, when she listens to music, she listens to whatever is being broadcast on the radio.

[3] pexel.com

Lewis Ilton



When Lewis performs as a street artist, he plays guitar, sings, or creates artwork. He is usually found in the first district and has already gained some attention in the art scene with his extensive skills. The 33 year old has a very relaxed mentality, is friendly, modest and likes to try out new things with his girlfriend and his family. Especially in the fields of art and music. He goes to many small concerts or other art events where he likes to chat with people and make new friends. Lewis is also color blind, a fact that strongly influences his art style.

Agatha Christinsen

Agatha is a 64 year old retired hairdresser who lives alone in a small apartment on the outskirts of Vienna. Since her retirement, she has enjoyed visiting museums and other quiet and calm public locations. She is a very insecure and hesitant woman, because she doesn't want to make anyone uncomfortable and doesn't want to be a burden. Apart from the museum visits, she loves to sing, even if she sometimes has problems finding the right rhythm in a song. When she sings, it is rather alone. Agatha is very skilled when it comes to fine motor skills.

Christine Bestin

Christine is a bright ten-year-old girl from the country side of Upper Austria. She loves to draw and is very eager and curious to learn new things. At school she convinces with exactly these qualities. Her three brothers all play an instrument, which is why Christine has expressed the wish to learn an instrument as well. At the moment she is still very uncertain and swings between piano and harp.

11.4 Requirements

As already described in the previous chapters, the management of requirements is a continuous process. This section thus reflects the result of determining the requirements, which is the first activity of requirements engineering (see Section 7.2). The result of the second and third activities of *Requirements Engineering* are also display in this section. The result of the fourth activity, which is the continuous management of the specified requirements, will was documented after each method of the practical part and can be seen in the appendix.

The importance of each requirement for the project was defined in three levels. The most important level, also called *Must Have*, is represented by a green dot (●). *Should Have* represents the second level and is visualized by the color yellow (●), whereas *Nice to Have*, which equals the lowest level, is represented by a white dot (○). The full list of requirements, as described in Section 7.2, are listed in the Appendix D.

The functional and user requirements are very vaguely defined at this stage of the project, as these requirements were defined more precisely in the creation process of the sketches and mockups phase (see Sections 11.6 and 11.7).

Business Requirements

1. ● The system should engage the user with music by letting him/her create melodies in an alternative and fun way.
2. ● The development costs of the project must not exceed 1700 Euro
3. ● The installation should attract more customers (e.g. in a museum)
4. ● The maximum development time for the three quarters of a year

System Requirements

1. ● The activities that the system enables the user to perform are not collaborative in nature.
2. ● The input from the user must be tangible (no wearables).
3. ● To support the user in the music creation process, musical accompaniment must be enabled by the system.
4. ● The interface communication between the MUI and the mechanical part must use a music protocol, like MIDI or OSC.
5. ● The generated music must be output via headphones.
6. ● The installation must produce music with analog guitar strings.
7. ● The written software must run on microcontrollers, like the Arduino or Raspberry Pi.
8. ○ The product of this work must be maintainable by non-programmers/ external personnel.
9. ○ The system should contain as much open source software as possible.
10. ○ The guitar strings on the sound creation mechanism must be easy to change.

User Requirements

1. ● The position of tones in the melody and their music parameters (pitch, duration) must be set by the user.
2. ● The position of tones in the melody and their music parameters (pitch, duration) must be changeable by the user.
3. ● The playback speed (in beats per minute) must be changeable by the user.
4. ● The user must have the possibility to change the output volume of the system.
5. ● The user can accept or decline tone suggestions by the system, which try to help him in creating a more harmonious melody.

Functional Requirements

1. ● The system must be able to capture the current position of each token used.
2. ● The system must have a constant connection to the input devices to set the playback speed and volume.

Non-functional Requirements

1. ● Delay between in- and output shall not impair the user.
2. ● The installation must fulfill all attributes for a good usability.
3. ● The interface should have a clearly understandable design by adhering to design knowledge.
4. ● The system must integrate playfulness.
5. ● The installation has to minimize initial interaction fears of the users.
6. ● The installation should be optimized for a interaction time from three to fifteen minutes.
7. ● Mechanical failure must be at an absolute minimum.
8. ● The system should be able to play at melodies with at least 180 bpm.
9. ● The pitches precision must not exceed the auditory threshold of about ±8 cents between two frets.
10. ● The table should not have a round shape
11. ● The tokens must be inexpensive (to replace).
12. ● Colour-blind users have no disadvantage in using the system due to their limitation.
13. ● The skill of the target group regarding music is low or non-existant, but the system should also work for people with musical knowledge.
14. ◒ The size of the user should not be an impediment for him or her.
15. ◒ The system rarely needs to be reconfigured.
16. ○ The system needs to be visually appealing.

After the specification of the requirements now follows the definition of use cases, which, as already mentioned in Subsection 3.5.3, represents an alternative and expanded representation of the user requirements.

11.5 Use Cases

The importance of use cases should not be underestimated at this stage of the project, as they form the basis of the project and highlight specific issues for further development. The interaction between the different use cases can be seen in the form of a UML use case diagram in Figure 11.2. The authors documented questions which arose from the creation process of the use cases, which answers represent design decisions taken in the course of this work. The questions are subdivided in *Software*, *Hardware*, *Input*, and *Feedback*. The few questions which can be answered at the current stage of the project are directly answered in this section. Because of the fact, that the use cases were created while the mechanical part of the thesis was still being produced (see Figure 7.3), only questions which result in a clear answer from the expert interviews, can be answered.

Each of The following subsections consist of the most important use case properties at the moment. The complete use cases with all properties, as explained in Part II, are attached in the appendix (see Appendix E.1).

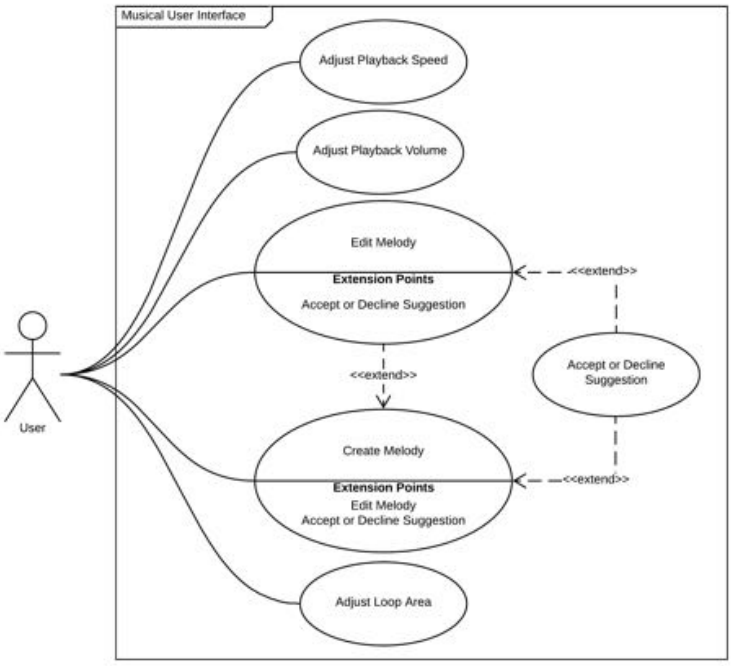


Figure 11.2: UML Use Case Diagram before the sketching process.

11.5.1 Create Melody

Description:

The user begins to manipulate sound parameters through a mixture of exploration and intuitive interaction with tangible input tokens. Concrete connections between changes of the user interface and acoustic feedback (output) through the mechanical construction are recognized by the user. Based on this knowledge the user tries to compose simple melodies. When the music is created, the transmitted information to the user is mainly on a visual level.

Resulting Questions:

- Software
  - Q: What are the software components of the system?
  - Q: How do the software components interact with each other (Interfaces)?
  - Q: What music creation/ editing philosophy is the system based on?
    A: The system uses the loop approach, where a selected area of the available beats are played in a loop. The loop area can also be changed throughout the users interaction.
  - Q: How many bars/ beats do exist?
  - Q: How many tones can be played?
    A: The exact number of playable tones is currently unclear. What is certain at the moment is, that the melody created by the user will be played on three guitar strings (from G, B and e string).

  - Q: How many tones can be played per beat?
  - Q: What principle is the accompaniment based on?
    A: There are currently two possibilities: Either restrict the possibility of chord composition based on the used musical scale or use predefined samples and transpose them if necessary.
  - Q: What predefined samples or chords should be used to accompany the users input?
  - Q: How long does the accompaniment take? Will it be repeated afterwards? How often?

- Hardware
  - Q: Should the accompaniment also be created analogy?
    A: According to the experts being interviewed, this is a valid approach. Three strings will be used for the melody and three strings for the accompaniment.
  - Q: What are the hardware components of the MUI?
  - Q: What hardware does the software run on?
  - Q: What material/ components is the table made of?
  - Q: What height does the table have to be?

- Input
  - Q: How do the tokens look like?
  - Q: How big should one token be?
  - Q: Is there any difference between the tokens? Are there any different token types?
  - Q: How can the tokens help the user to understand the current system status?
  - Q: What is the musical metaphor of a token (single tone, a tune, a chord)?
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?
  - Q: Can the user choose between accompaniment melodies or are they chosen by the system?
  - Q: How can the accompaniment be selected (if the user has the possibility to do so)?
  - Q: Should it be possible to adjust the volume of the accompaniment independently to the overall volume?

- Feedback
  - Q: How can the current system status be displayed comprehensively?
  - Q: What colors should be used for which system status/ property?
  - Q: How can the user be supported without given them the feeling that the system does not do something they don't want to?
  - Q: What visual feedback should be shown to the user?
  - Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?
  - Q: What feedback does the user get concerning the accompaniment?

11.5.2 Edit Melody

Description:

The user edits the previously created music by rearranging the tangible objects. Editing is done according to the similar principle of creating music (see Use Case 1).

Resulting Questions

- Software
  - Q: Can melodies be persisted and loaded?
  - Q: How can the system support the user, so that his/ her short term memory load can be reduced?
- Input
  - Q: How does the user change the already defined tones?
  - Q: How does the user remove/delete already defined tones?

11.5.3 Adjust Playback Speed

Description:

The user lets the current melody play faster or slower by using the associated input device. By using this device, he/ she is able to let increase or decrease the playback speed until a certain limit. The system reacts accordingly and plays the created melody faster/ slower.

Resulting Questions:

- Software
  - Q: What are the minimum and maximum bpm being allowed by the system?
- Hardware
  - Q: Use continuous transition between bpm or stages of bpm?
  - Q: If stages are being used, how big are they?
- Input
  - Q: Where on the installation should the device to adjust the playback speed be placed?
  - Q: Which input device should be used to adjust the playback speed?
- Feedback
  - Q: What is the haptic feedback when turning the input device for the playback speed?
  - Q: What icons can be used as a visual metaphor for *(*fast) and *(*slow)?

11.5.4 Adjust Playback Volume

Description:

The user adjusts the volume output via a physical input device on the installation. The adjusted sound volume then gets output via headphones the user wears. When adjusting the volume, haptic and/or visual feedback is returned.

Resulting Questions:

- Hardware
  - Q: Which device should be used to output the sound?
    A: The user wears headphones (see Section 11.2).
- Input
  - Q: What input device should be chosen to adjust the volume?
  - Q: Where on the installation will the device to adjust the volume be placed?
- Feedback
  - Q: What icons should be used for the visual representation of the sound volume?

11.5.5 Accept or Decline Suggestions

Description:

During the creation of the melody, help for the token(s) to be set gets displayed on the user interface. The user can either accept the suggestion (interact with it) or decline it (actively decline or ignore).

Resulting Questions:

- Software
  - Where is the suggestion displayed?
- Feedback
  - Q: When is the suggestion displayed? What technical parameters trigger them?
  - Q: How does the suggestion visualization look like?
  - Q: What is the right amount of suggestions? The user should not feel that the creation process is being taken away from him.
  - Q: Does the suggestion need to be non-intrusive?
  - Q: How can the suggestion be non-intrusive?
- Input
  - Q: How can the user accept or decline suggestions?

11.5.6 Adjust Loop Area

Description:

While the melody is being played in a loop, the user can adjust the area of tones which will be repeated infinitely.

Resulting Questions:

- Feedback
  - Q: How is the loop area being displayed to the user?
  - Q: What does the feedback look like, when exceptional cases (e.g. loop area get's moved but not set while the music keeps playing) happen?
- Input
  - Q: How does the user change the loop area?

11.6 Sketches

Based on the statements of the expert interviews and the literature research, the authors opted for the loop approach as the basic functionality of the system. With the loop approach the user selects an area within the available number of beats, which is then played repeatedly in a loop. During playback, the selected area can be adjusted.

After the selection of this functionality, the aim of this phase was to design different ways of interaction between the user and the system and to find possible answers to questions which were defined in the previous course of this work (see Section 11.5). The sketches were mainly created on a white board and table with white paper laying atop of it. The authors used brain storming, followed by discussing the resulting possibilities. As it can be seen in this section, the boarder between sketches and wireframes were very blurry but wasn't the main concern of the authors, as it was to create a good approaches. Additionally, post-its were used to visualize additional info, uncertainties, open questions or similar. The focus was to come up with multiple applicable ways of interaction. Some sketches also had physical components in it, which are more likely to be assigned to mockups.

The results of the sketching process resulted in three possibilities on how to design the UI. One of those possibilities had to be discarded (see Subsection 11.6.2) and one had two different approaches for its token types (see Subsection 11.6.1). The following three subsections each explain one of these design possibilities in chronological order to the creation process, whereas the last subsection is dedicated to the impact of the sketch process on the working document of the project.

11.6.1 Blank Approach

At the beginning, the blank approach started under the name *Grid Approach* (see Figure 11.3), but changed its name in the course of its creation. The grid was intended as a visual help for the user to see where new beats and pitches begin. The token at this point where in rectangular form and intended to have three different tone lengths (1/2, 3/4 and 4/4 length). The width of the token mapped the length of the tone. The token can be placed on four different bars with four

beats per bar, therefore four 4/4 bars or 16 beats. The chronological sequence of tones/ beats is displayed on the x-axis and goes, as usual in western ethnography, from left to right.

The y-axis on the other hand displays the pitch of the tones that can be played. It is possible to choose between 23 different pitches, starting with G3 and ending with D5. Each string has its own color which gets displayed underneath the token in the form of a rectangle. The rectangle is slightly bigger than the token and is shown as soon as the token gets laid on the tabletop. This visualizes the current system and token status to the user, which will be important when defining the number of playable tones per beat (see Section 11.7). If the token gets removed, the visualization also vanishes. As already stated, the created melody of the user can occupy three of the six strings from the mechanical part. These melody strings are installed the way it can be seen in Chapter 10, due to the rapid succession of tone changes. Tones which can be played on one of two strings, were assigned to only one of the strings only to achieve consistency in sound generation and to not lead to confusion when the same tone gets played on two different strings.

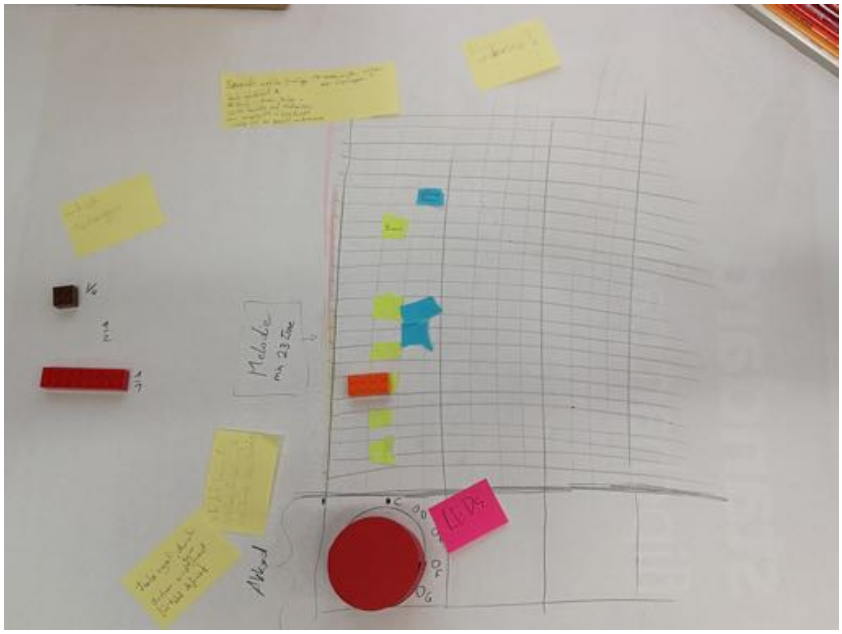


Figure 11.3: Sketch of the *Grid approach*.

The other three strings are, as suggested in the expert interviews, used for the accompaniment. At the beginning of the project it was considered to implement a fully automatic accompaniment. However, as most music creating algorithms can not compose harmonic melodies and the calculation in real time is extremely computationally intensive and complicated to create, the authors decided to use power chords (see Subsection 4.1) instead. A round knob, displayed as the red cylinder in Figure 11.3, is used to set the base-tone of a power chord. This power chord is being played for the duration of one bar. That means, that under each bar one knob for the respective power chord is present. The knob itself can be set to a number of different positions, each position representing one chord. There is no continuous transition between the positions, as they rather snap in position. The top position of the knob deactivates the power chord. Next to the knob, the name of the power chords base-tone is display. The idea came from one of the interviewees of the

expert interviews (see Section 11.2). The expert suggested displaying the name of the base-tone enables the user to recall (as recognition might not be possible for non-musically trained people) the name when trying different chords and he or she wants to got back to a previous tone. The current selection is also highlighted with a LED to easier make out which chord is currently active (apart from a line on top of the knob, not sketched in Figure 11.3).

As already mentioned at the beginning of this section, the loop approach was chosen as the underlying functionality of the system. The initial idea was to enable the user to move the start and end of the loop area via touch. However, since this interaction would differ greatly from the other tangible interaction, would have violated system requirement 2. and would most likely not be understood without any kind of additional information (like a tutorial) this idea was discarded. Instead, it was decided to use two tokens, which are physically limited to the displacement along the x-axis (see lower part of figure 11.3). These tokens have different icons on their top than the tone-tokens to make their purpose more understandable to the user and to be visually distinguishable. A miniature overview is displayed between these loop-tokens where the currently placed tokens are visualized. The height of the overview is reduced in height but not in length. Both loop-tokens also draw a vertical bar (not displayed in Figure 11.3) on the interaction area on which the tokens are placed. Between these bars, a vertical line moves from left to right, causing all the markers passed through to be played while passing. If the vertical line (which will be called *current-location-bar* in the rest of this thesis) passes the right bar visualized above of the loop-token, it starts from the bar above the left loop-token.

After sketching the grid approach the two authors partially reconsidered some of the already defined properties and discussed and argued them anew in a joint exchange. This exchange led to changes regarding the available token tone lengths and the overview. For the tone length, the 1/4 tone was added as the smallest token which can be laid. This change should enable the user in creating a greater variety of melodies on the four 4/4 bars. It must be evaluated in the oncoming evaluation, if the users get overwhelmed by the additional possibilities which might result in frustration and in a shorter interaction time. The reconsideration also had its impact on the miniature overview, which was completely removed from further implementation. This was justified by the fact, that the interaction area does not correspond to any size that would have the user lose the overview. It was also defined, that the loop-tokens are now being used at the bottom of the interaction area, with physical limitation to the x-axis. This area on which the tokens can be moved is reserved exclusively for the loop-tokens, since otherwise problems with tone-tokens being placed at the lower end of the y-axis can occur. The lines of the grid approach were also discarded. The fear was that the displayed lines would be bad for the understanding of the user because of the visually overloaded GUI. This removal is also responsible for the (reconsidered) name of this approach, the *Blank Approach*. The focus of the blank approach is on a clearly structured and not overloaded GUI. However, the absence of the grid can have a negative impact on the users' orientation. Therefore *Snapping* was implemented as an assistance (see Subsection 11.6.4). Snapping refers to the functionality, that the edges of the colored rectangle, displayed beneath the token, are projected to the nearest beat and pitch, formally being displayed by the grid. Snapping therefor enables an invisible grid. With snapping, the user should have no difficulty in setting the rhythm adapted to a 4/4 beat. However, snapping cannot be rejected by the user and is therefore not a non-instrusive aid. On the other hand, snapping can also be seen as a form of error prevention and something that protects the user from frustration (see golden rules of UI design in Subsection 3.4.1). It may also be seen as a *Token and Constraint* approach implemented on the software side of the installation. Because of this different viewpoints, snapping needs to be evaluated throughly later in the design process.

Another draft of a visual aid can be found at the bottom of the image (see Figure 11.4). Not

set and still free bars and beats should be represented somewhat darker in this design. However, this idea was not considered practicable, especially because the darkening of this areas would probably have led to confusion for the users and the darkening of screen areas could be better used for a different purpose (see Section 12.1).

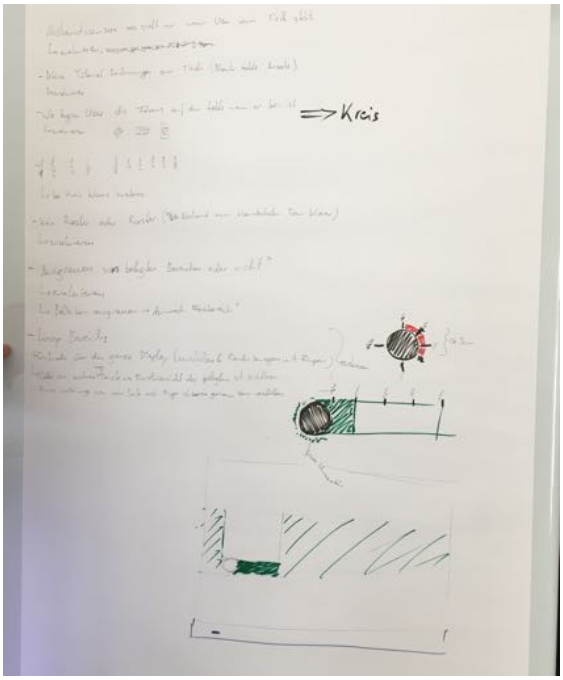


Figure 11.4: Sketch of the *Blank Approach*.

The second possibility for the tone-tokens of the blanks approach can be seen in Figure 11.4 as this tokens have the shape of a cylinder. The idea was to change the length of the token by turning the token to the left or to the right. The current tone length is represented by a bar which gets filled when the token gets turned clockwise and emptied when its being turned anti-clockwise. The color of the bar represents the color of the string on which the tone is being played. Predefined fill levels, visualized by small markings on the top and bottom of the bar, indicate when a certain tone length is set. The bar itself gets filled continuously. A suspected problem concerns the haptic feedback when the token gets rotated. Since the token does not have an anchor point on the table's surface, an unintentional movement during rotation is foreseeable. A possible solution to this problem is the installation of small magnets on the top and bottom of the tokens and in the tabletop itself. The magnets in the table surface would be covered by a semi-transparent foil on which the picture gets projected (see Section 12.1). This would also be an equivalent to a hardware-side implementation of the snapping- aid. Apart from the different tone-tokens, the blank approach with cylindrical token doesn't differ from the other properties.

To help the final decision on to use the cylindrical or rectangular tokens, both possibilities need to be evaluated.

11.6.2 Circle Approaches

In the circle approaches, the interaction's surface was intended to have a different shape than the interaction area of the previous approach (see Figure 11.5). As a result of this idea, two approaches were created, both of which are presented here.

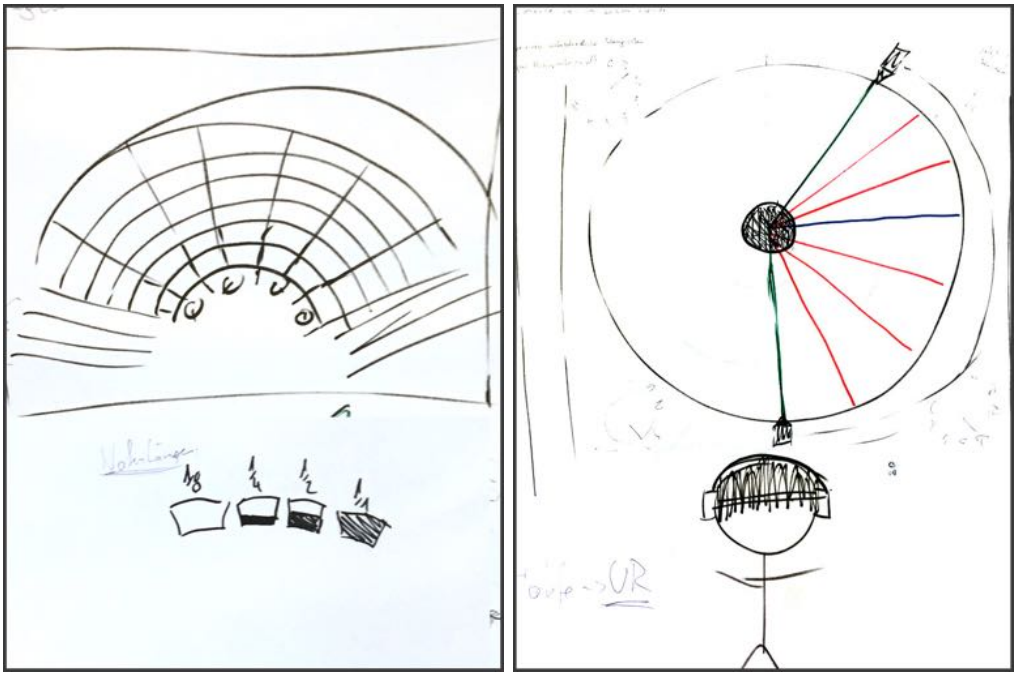
Starting with the similarities between both approaches, the positioning of the mechanical component of the installation is, as with the other approaches, still unclear. The sketches visualize two proposed positions for three strings respectively, as they are drawn on each side of the table top. On the left side for the created melody and on right side for the accompaniment. The effect of this arrangement may hinder the continuous eye contact with the interaction surface advantage of TUIs and thus negatively affect the attention focus of the users' flow state (see Subsection 3.4.2). An uncertainty that must be evaluated. The accompaniment and its adjustment possibilities were based on the same principle of the blank approach.

The full-circle approach also uses two tokens to set the loop area. The selected loop area, its ingoing line (green in Figure 11.5a and the current position in the course of the created melody (in the form of a line) is adopted from the blank approach. The time sequence takes place clockwise, in contrast to from left to right. This in turn meant that not only did the order of the placeable tones changes from left to right to right to left during a full rotation, but the positions of high and low tones at the lower part of the circle are, as opposed to the upper part of the interaction area, reversed. A circumstance that might cause additional difficulties for the users in keeping the overview. The shape of the interaction surface, which strongly resembles the visual output of a radar, would not have such negative influences on the orientation if the user stood in the middle of the circle. Following this possibility, the user loses a lot of overview, because the whole table can never be seen. Additional problems, such as being tangled in the cables of the headphones, and the process of getting in the center of the table demonstrate the impracticality of this approach. Furthermore since the shape of the interaction surface is round, but the shape of the table has to be rectangular (as it would imply collaborativity), it is very likely that this circumstance can lead to confusion.

These impracticabilities led to the design of the half-circle approach (see Figure 11.5b). With this approach the user stands in the middle of the semicircle. The semicircle has the same structure and properties as the grid approach. The, now curved, x-axis represents the time course from left to right and the y-axis represents the pitch height. The accompaniment and the loop area are also like in the previous rectangular approach, only in semicircular form. It was suspected that the half-circular form would solve some, but not all of the encountered problems.

Additional difficulties with the tokens in both approaches came to light. Due to the increasing radius of each pitch line, the tokens must also be able to change their curvature and size or have a size and form, that is regardless of the radius of a circle. One possible solution was to give all tokens the same size and reflecting the actual tone length by the visual feedback of the system. To achieve this there must have be several tokens with different colors or other physical or visual properties. The idea was, to show the different tone length via the *fill level*, displayed on top of the token. The fill level, shown dark in Figure 11.5b, stands out physically from the rest of the token, so that haptic differentiation is also possible.

With the full- and half-circle approach there was also the consideration of attaching the interaction surface to the wall. However, this was rejected for several reasons. For example because of possible problems regarding different heights of the users. This could be solved with an adjustable height of the display area. However, if the interaction area has to be height-adjustable, the whole table has to be. This is because of the beamer, which generates the projected picture, must maintain



(a) Full-circle Sketch (b) Half-circle sketch

Figure 11.5: Sketches of the two differing, created *Circle Approaches*

the same distance from the tabletop so that the displayed image always has the same size. It must therefore be moved together with the interaction surface. This in turn has a negative effect on people with physical impairments, children or people with insufficient physical strength.

As can be seen, this approach faced some diverse problems, where only some of them didn't interfere with a number of requirements or didn't make the interaction between user and system challenging. The circular form in itself does not serve any direct purpose in this draft and does not fulfill any requirement but rather complicates it. Therefore, the circle approaches were discarded shortly after the sketches were made.

11.6.3 Drawing Approach

This approach had the aim of finding a tangible interaction method without using token which constantly lay on the interaction surface. Apart from that this approach has the same properties as the blank approach (see Subsection 11.6.1).

While brainstorming the authors defined two tangible input objects: a pen and an eraser (see Fig11.6). As soon as the tip of the pen touches the table, tones can be created, i.e. *drawn*, by vertical and horizontal movements. Hence the name of this approach. A tone lasts as long as the pen is in contact with the table at a certain pitch. As soon as the pen exceeds the y-axis position which the pitch is adhered to, a new tone starts at the new pitch. This is a fairly simple approach, which can be expanded in many different ways. With continuous tracking of the input device it is for example also possible to create transitions between tones which are many pitches apart from each other. Th created tones can be shifted by a continuous touch of the pen.

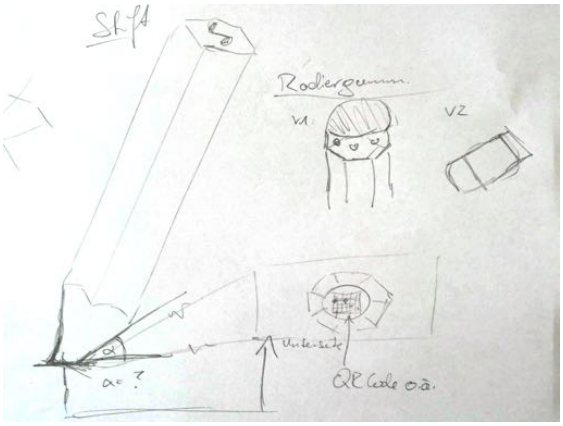


Figure 11.6: Sketch of the *Drawing Approach* showing a pen and eraser tools.

The pen itself is larger than ordinary pens, primarily to convey that it is not a normal pen and that a QR code (or similar) can be clearly recognized by the system at the tip of the pen. Because larger patterns are easier to recognize than smaller ones, and the system behaves less inconsistently, the pen must be larger but easy to use. Direct contact between the pattern and the table is also advantageous, as different lighting conditions can influence image recognition. Therefore, the tip of the pen must be chamfered at an angle that does not negatively affect the interaction. The angle of this bevel is unclear at this point.

Created tones can be deleted by touching the table surface with a second object, the eraser. The eraser, like its counterpart, has a code attached to it that erases the corresponding tone when it touches it visual representation on the table surface. Instead of being an independent tangible object, the eraser can also be attached at the end of the pen.

Since there are no tokens at the table, it is also possible to have four bars several times available, save them and switch between them in order to produce longer melodies. If this was being used, the comeback of the overview would be a logic consequence, as keeping track of several 4/4 bars is not easy. In this case, the miniature overview, mentioned in the creation process of the blank approach, should have a size that allows the loop area to be precisely adjusted even with the 4/4 bar currently displayed on the interaction area.

11.6.4 Repercussions

The biggest change is considered to be the addition of another functional requirement to the project. Since many people who are not familiar with music do not have a good sense of rhythm, it was decided to support the user in determining the right set-in of tones within the created melody. The correct set-in is based on the respective beats of the four-quarter beat used. This is achieved by the use of the previously mentioned snapping. Snapping works as follows: As soon as the user puts a token on the interaction area, thy system recognizes it and responds with visual feedback in form of a colored rectangle which is slightly larger than a token. This rectangle is, as already explained, displayed beneath the token. When the token doesn't move, the center of the rectangle automatically moves horizontally to the nearest beat and stays there. The same applies on the vertical axis but with the nearest pitch. To the user this looks like the rectangle snaps to a position that the system considers valid. When moving the token again, the

rectangle is again displayed directly underneath the token. It is suspected, that snapping can be seen as an unwanted intrusive or confusing help, a circumstance, which must be evaluated in the forthcoming evaluation.

In the course of the sketching process the authors agreed, that the user is able to set the accompaniment by him-/herself. This results in an additional user requirement, a new use case, and an updated use case diagram (see Figure 11.7) as well as in the requirements (see Appendix D.2). Throughout the creation of the current three approaches, previously raised questions from the other use cases (see Section 11.5) could be answered and are described in the following. It should be noted, that all questions from the previous use cases concerning the accompaniment will be shifted to the new accompaniment use case.

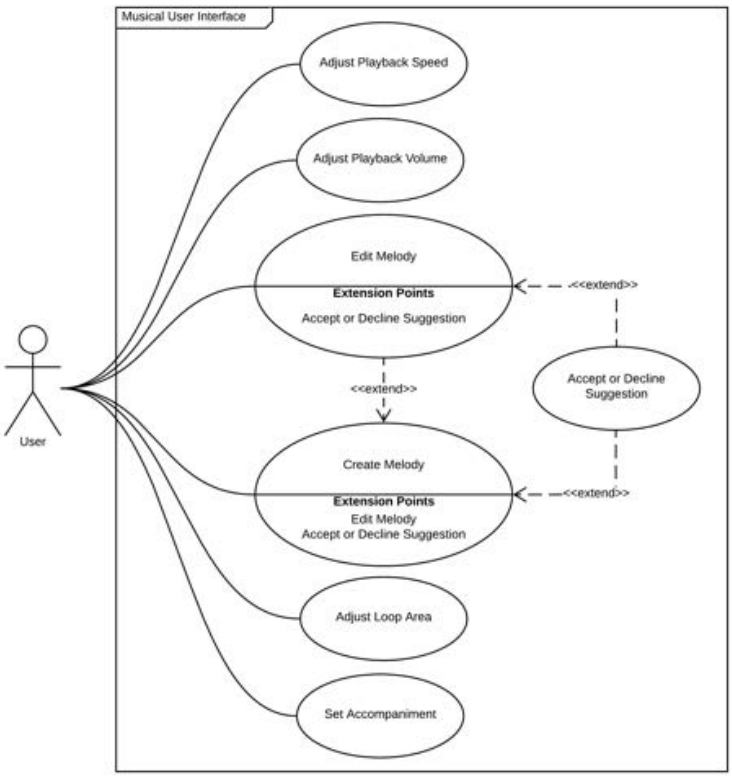


Figure 11.7: UML Use Case Diagram after sketching.

Use Case- Set Accompaniment

Description

The user is able to set and change the accompaniment, in the form of a power chord, for each of the four bars available to him. Each bar has one turning knob beneath the interaction surface. The base tone of the defined power chord is beneath the turning position of the knob and will be highlighted with the help of a glowing LED next to the base tones name.

Resulting Questions

- Software
  - Q: What principle is the accompaniment based on?
    A: There are currently two possibilities: Either restrict the possibility of chord composition based on the used musical scale or use predefined samples and transpose them if necessary.
  - Q: What predefined samples or chords should be used to accompany the users input?
    A: Power chords will be used to accompany the user. It is a simple way of supporting the users melody, needs no runtime computation, fits to every melody and can be played on three strings.
  - Q: How long does the accompaniment take? Will it be repeated afterwards? How often?
    A: One power chords is set for one bar. The power chord starts with the first beat and ends with the last beat of the coherent bar (or loop area, if it is smaller than one bar).
- Hardware
  - Q: Should the accompaniment also be created analogy?
    A: Yes, the accompaniment will be play analogy on the three lower strings of the mechanical part of the installation.
- Input
  - Q: Can the user choose between accompaniment melodies or are they chosen by the system?
    A: The user can choose the power chords base tone for each bar.
- Feedback
  - Q: What feedback does the user get concerning the accompaniment?
    A: The current accompaniment for one bar is highlighted with an active LED at the current settings of the knob, which sets the power chord. Additionally, each possible power chord has it's base tone written next to it.

Answered Questions

- Software
  - Q: How many bars/ beats do exist?
    A: There exist four bars consisting of four beats, therefore sixteen beats in total.
  - Q: How many tones can be played?
    A: The user has the possibility to play 23 different tones (see Section 11.2) on three strings. One tone is always played on one certain string to prevent confusion.
  - Q: Can melodies be persisted and loaded?
    A: No. Melodies can not be persisted or loaded. If the Drawing Approach will be pursued further, this might be a possible addition to the installation.
  - Q: What are the minimum and maximum bpm being allowed by the system?
    A: At least 180 bpm must be possible (see Subsection 9).

- Input
  - Q: How does the user change the loop area?
    A: There is a second kind of token, with which the user is able to change the loop area.
  - Q: How do the tokens look like?
    A: One of the approaches has two different tone- token approaches. In the first approach, the tokens have a rectangular form, whereas in the second approach, the token have a circular form. The loop area token should differ in form and icon atop of the token, there is currently nothing more defined.
  - Q: Are there any different token types? What is the difference between them?
    A: There are currently two types of token. One for the adjustment of the loop area and one acts as a physical metaphor for tones.
  - Q: How can the tokens help the user to understand the current system status?
    A: A token directly can't help displaying the current system status, but the system recognizing the token displays a colored rectangle beneath the token. This token is slightly larger than the token itself and has the color of the corresponding string it will be played on.
  - Q: What is the musical metaphor of a token (a single tone, chord or similar)?
    A: One token resembles a single tone. Tokens differ in width, which reflects the tone length.
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?
    A: The token are not limited in any way.
  - Q: How does the user change already defined tones?
    A: Depending on the approach, there are different ways of changing the length and position of the defined tones. With the rectangular token, the token itself must be replaced with another token to change to the desired length. Changing the tone length with the cylindrical tokens is achieved by turning the token (anti-) clockwise. The Drawing Approach has no intended interaction to change the tone length. It must be therefore deleted and set again to change its length. Changing the tone position in the approaches using token can be achieved by simply moving the token on the interaction surface. The Drawing Approach achieves this by continuously touching an existing tone with the pen and moving the pen over the interaction surface, whereas the visual representation of the tone follows the pen.
  - Q: How does the user delete/ remove already defined tones?
    A: All token using approaches delete existing tones by removing the token from the interaction surface on the tabletop. The Drawing Approach deletes tones by touching them with the eraser, the second tangible interaction device of this approach.
- Feedback
  - Q: How can the current system status be displayed comprehensively?
    A: Apart from the mentioned visual feedback, this question will be answered in the following evaluation.
  - Q: What colors should be used for which system status/ property?
    A: Currently colors are not being used for the display of any property or status.

- Q: How is the loop area being displayed to the user?
  A: As explained in the next answered question, the loop area is being visualized with the help of two vertical bars across the interaction area.
- Q: What visual feedback should be shown to the user?
  A: The visual feedback consists of every recognized tone on the interaction area, a loop start and loop end bar, a line which moves from the start bar to the end bar (and therefore displays the current position in the melody) and the base-tone of the chosen power chords.
- Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?
  A: No, this would be too expensive and would interfere with non-functional requirement number 11..

Sketching provided the development process with several possibilities on how to interact with the system. Some of the provided answers and solutions need to be tested in the evaluation process, to find out if the intended solutions also work for the user. But before that, some of the remaining, unanswered questions need to be addressed, something that the creation of mockups is intended to do.

11.7 Mockups

As already described in the previous section, the main aim of the mockups is to answer the already stated questions that remain unanswered from the previous design methods and to make further improvements, so that the forthcoming user evaluation of this iteration can take place. As with the blank approach, the time was used to reconsider current decisions, to discuss it in a joint exchange and to change it if necessary.

The results of these exchanges are the content of this section. The first subsection describes changes that affect both approaches, the following two subsections contain information regarding the mocking process of the blank and the drawing approach. The last subsection consists of all repercussions of all changes happened in the creation of mockups to the current working document.

11.7.1 General Changes

Since the listed changes in this subsection concern both approaches to almost the same extent, it is not necessary to differentiate these changes according to the respective approaches.

The biggest change of the mockup phase concerns the deletion of a functional requirement. As described above, the total effort of the project was very extensive as why the accompaniment as a whole was removed from the project scope, as it is the use case which can be removed without any further effects on other system properties. Nevertheless, it is still possible and desired to implement an accompaniment in future works (see Chapter 14). In addition to this functional requirement, all relevant questions which are linked to the accompaniment have been removed from the resulted questions.

Furthermore, the suggestions for the user were defined more precisely. The scenario is as follows: the user is in a situation in which he/ she does not find an appropriate tone at the point of his/ her choice in the melody. The system proposes a tone at the specified location, which the user

can then accept or reject (see Section 11.5). The difficulty lies in recognizing the position at which the user might need help, the time at which the help is offered and selecting the correct tone that will be displayed to the user as a suggestion to match his previously defined tones. As a representation of these suggestions, a visual representation in the shape of the token is displayed at a certain location. The difference to the representation beneath the tone-tokens is, that the visualization is not colour filled but only consists of coloured outlines (see Figure 11.8). The position of the suggestion is always one beat after the position on which the last token has been placed. There are two options for selecting the pitch of the suggested tone. The first possibility is based on a Markov model (see Subsection 4.3.3), which selects the most suitable tone based on the previously played tones. The second possibility is an algorithm, which selects a random tone from the pentatonic scale (see Section 4.1.9). When creating the mockups, there was no final decision made, as it was not required for the upcoming evaluation and will be defined in the second design iteration. The suggestions visualization is displayed after the selected loop area hasn't been changed but has been played four times in a row from start to finish. This way of determining the moment of possibly needed help was chosen not to overwhelm the user with suggestions and because of the circumstance, that the elapsed time without any input can be interpreted as a time frame in which the user doesn't know what to do and therefore needs help. The user can either accept the suggestion by laying a token into the displayed suggestions-rectangle or ignore it, which equals declining the proposed suggestion.

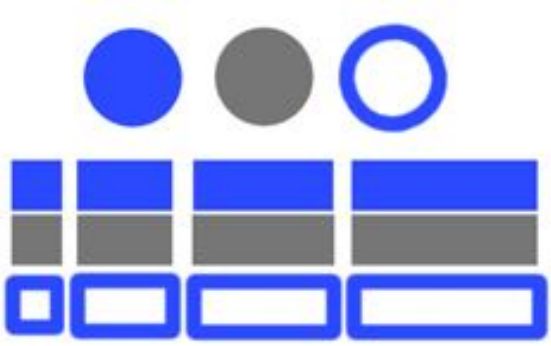


Figure 11.8: All token related visual feedback as used for the mockups.

One question that was also clarified in this phase is the number of playable tones on one beat. As a reminder, it is either possible to allow one tone per beat (on all strings) or to play one tone per beat and string, which would equal the property of a guitar. The authors changed the previous approach of one tone per beat and string and opted for one tone per beat, arguing that users would be exposed to a possibly less frustrating experience. The potential frustration is due to the high number of possible pitches, and thus sound combinations, between two or three tones and the possible discord that may arise in the creation process. This choice also entails another change. The displayed color of the strings, which should make it easier for the user to differentiate which tone is played on which string, are more likely to be the source of confusion, because the system would sometimes visually differentiate between two tones but the system itself would make no difference. A visual differentiation without any comprehensible reason for the user. Accordingly, the colored display of the pitches at the boarder of the interaction surface has been removed and replaced with one color (see Figure 11.8). Whether this decision was the right one must be evaluated in the context of the following user tests.

Furthermore, it must be clarified what happens when the user lays several overlapping tones on the same beat. In this respect, the systems behavior is based on the *First Come First Serve* approach. This means, that the first token to be placed on a certain beat will be played as long as it lays there. If the token gets removed, the token which has been laid there after the now removed token will be played as long as it gets removed and so on. All inactive token are visualized by a gray rectangle underneath the token, whereas the active token's rectangle is colored in blue.

Some changes were also made in context with the playback speed and volume of the system (see Figure 11.9). Rotary knobs were chosen to be used for both input devices which are mounted on the left side under the interaction area of the installation. For the icons of the playback speed, a turtle was chosen as a metaphor for slow playback and a hare as a metaphor for fast playback. Above the playback speed knob, a half-arc is drawn. Three vertical lines can be seen on this half-circle, so three speeds can be set by the user. The turtle is shown on the left, and the rabbit on the right side of the half-circle. Above the knob of the volume is also a half-circle, which gets thicker from left to right. To the left of the arch is a speaker with a visualized sound wave in front of it, to the right another speaker with three sound waves in front of it.

It was also decided that not one, but two pair of headphones should be usable at the final product. This enables the user to share his experience and creations with other people which he or she visits the museum. This is not an important feature for the prototype which is being developed in this thesis, as only single users will be evaluated in its course. Nevertheless, the containment of multiple headphones is recorded in the form of a system requirement.

11.7.2 Blank Approach

During the mocking process, the mockups were used on top of the table which was used for the functional prototype (see Section 12.1), and to test if there might be any problems with the table's height. The table can be seen on many of the following figures (e.g. Figure 11.9).

Figure 11.9 displays several mockups. First, however, it should be mentioned that the semi-transparent, white surface on the tabletop is the, several times mentioned, interaction area. One of the two loop bars can be recognized as the red vertical line on the left side of the interaction area, which resembles the start of the loop area. The end of the loop area is on the right end of the interaction area, also in red. Both consist, apart from the red bar, of a loop-token at the bottom of the interaction screen. Previously used 3D prints have been used as a mockup for these tokens. Additionally, slightly yellow painter's tape has been used to tape on the prints as a physical basis to draw two arrows upon. The arrows, pointing to the left and right respectively, should indicate, that the tokens are only moveable on the x-axis of the interaction area. The physical limitation of the loop bar token has not been mocked. Between the loop bars lies the current-location-bar which indicates the current position of the system in the melody. As mentioned before, this bar loops from left to right between the start and the end of the loop area. Every active tone (represented by a blue rectangle beneath the tone-token) gets played when it gets passed by the current-location-bar.

A few mockups can also be identified in front of the interaction area. The bottom left edge of the tabletop shows the mockups for adjusting the playback speed and the volume. Knobs from available rotary encoders were used as mockups. The icons and semicircles described in the subsection before were printed on a piece of paper, which were placed underneath the knobs. Next to it lie a pair of headphones, the same model which has already been used in the process of creating the mechatronical part of this thesis. On the remaining part of the wood plank in

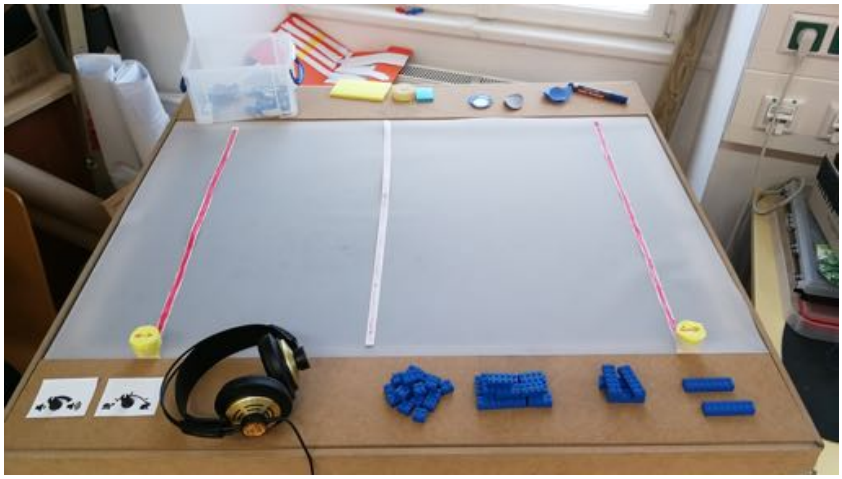


Figure 11.9: Mockup of the *Blank Approach* with rectangular tokens.

front of the interaction area are the rectangular tone-tokens for the blank approach. Lego[4] bricks were used to mockup these tokens. For the 1/4 beat tokens 2x2 bricks were used, 4x2 bricks for the 1/2 tokens, 6x2 bricks for 3/4 tones and 8x2 bricks for the 4/4 tokens. It can be seen, that the authors used fewer tokens for longer tones. This is part of the upcoming evaluation, which should give information on how many tokens should be used for each tone length. The light blue mockups used for the cylindrical token can be seen in Figure 11.10, in front of the interaction area. These discs, with a diameter of about thirty millimeters, are made of plastic and are originally used for the construction of three-dimensional structures.

Figure 11.10: Cylindrical token mockup.

After completing the mockup description of the blank approach, the subsection below describes the mockups of the second approach.

[4]https://www.lego.com/en-sg

11.7.3 Drawing Approach

The drawing approach has much in common with the blank approach, except for creating and editing sounds. That is why only the different tone-tokens are described here. Two simple mockups were used for the drawing approach. For the pen, which is responsible for creating and editing the sounds, a white board marker was used. Its larger appearance is very similar to the future token. For the eraser token, a 2x4 lego piece was used.

The effects of these changes on the design and development documents are, as in the sections before, described in the following subsection.

11.7.4 Repercussions

This subsection starts with a change that is not directly related to the use cases. It is an additional system requirement according to which two pair of headphones should be usable at the installation. The position of the second pair of headphones on the table must be well chosen, because a badly chosen position can have negative effects on the user. The headphones must be installed where is clear to one or multiple people that the installation is intended for one user only. A suitable position for this would be, for example on the right side of table, away from the center in front of the interaction surface. This indicates, that the person in front of the interaction surface is in charge of using the installation.

The second change concerns the accompaniment. The removed accompaniment has several effects on the working document. In addition to the removal of all requirements associated with it, the use case for selecting the accompaniment is also removed. Thus, the use case diagram looks the same as before the sketches were made (see Figure 11.2), which is why a further presentation is omitted here. As in the previous section, all changes on requirements can be seen in the appendix (see Appendix D.3).

Other changes that have occurred in relation to the use cases are (briefly) listed in the following subsection. The full changes on the use cases can be seen in the appendix (see Appendix E.3). The last subsection treats the newly answered questions with the help of the design phase of this section.

Use Cases

Now that the input devices and interaction processes of all approaches are clear, the updated description of every use case is being updated according to its approach and displayed at this part of the thesis. The complete use cases can be looked up in the appendix (see Subsection E.3). The first two use cases contain three separate descriptions because the different input methods vary depending on the approach.

Create Melody

The creation of a melody is solely achieved by using the tokens of the respective approach. Every other interaction with the system through other input devices is covered in the other use cases.

The user has several tokens to choose from to create music with rectangular tokens. They differ in width, but not in height or depth. The width of the token is considered a physical methaphor for tone length. The wider a token is, the longer it is played. There are four token with different widths to choose from. As soon as the user places a token on the interaction surface, a projected rectangle appears directly beneath the token, which provides information about its current state

to the user. If the rectangle is grey, the tone is not played, as it is deemed inactive by the system. If the rectangle is colored, the tone is considered active and will be played by the system. If a token is placed at a position where there is already an active tone, the new token is set to inactive and the old tone is continued to be played.

This also applies to the blank approach with cylindrical tokens. The difference to the rectangular tokens lies, apart from the form of the tokens, in the determination of the tone length. There are no tokens of different width here, because the tone length is determined by rotating the tokens. As soon as the user places a token on the interaction surface, not only a circle appears below it, which gives information about the activity of the token, but also a bar, which reveals the tone length.

A pen is used for the last possibility of an tangible input device. This metaphorical pen is picked up by the user and drawn over the interaction surface with its tip. A new tone ends as soon as the user either lifts the pen or moves beyond the range of the current tone on the y-axis. As soon as this happens, the old tone is stopped and a new tone with the new pitch is started at the respective position.

Edit Melody

Editing music in this case means changing tone lengths and heights using tokens. This means, that a precondition of this use case is, that tokens are already present on the interaction area of the table.

The tone length can be changed with the rectangular tokens by exchanging the tokens. To change the pitch and the position in the melody, a token must be moved on the y- or x-axis of the interaction area. To delete a tone, the user simply removes the token from the interaction area.

Changing the pitch and position with cylindrical tokens is done in the same way as with the rectangular token. To change the tone length, the user must rotate the token. If the token is rotated clockwise, the visually displayed bar under the token grows from left to right, and if it is rotated in the opposite direction, the bar shrinks. The removal of sounds is achieved by the user removing tokens.

With the Drawing Approach, the user can change tones by touching the visual representation of the token on the interaction surface with the tip of the pen and pulling it across the table to the new desired position while the touch is held up. The change in tone length is achieved by deleting and recreating the tone. The sound is erased by touching the visual representation with the eraser.

Adjust Playback Speed

To adjust the playback speed, the user has to turn the according knob on the left side of the table, below the interaction surface. The purpose of the knob is visualized to the user by the use of specific icons (a tortoise and a running rabbit).

Adjust Playback Volume

To adjust the playback volume, the user has to turn the according knob on the left side of the table, below the interaction surface. The purpose of the knob is visualized to the user by the use of specific icons (two speaker with a different number of sound waves emerging from them).

Accept or Decline Suggestions

After going through the current loop area four times, the user is presented with a suggestion. The suggestions is visualized and consists of the outlines of the current token shape. The user

can ignore the suggestions and thus decline it, or accept it by laying a token in the displayed shape. When doing so, the visualization gets replaced with a filled, token shaped visualization.

Set Loop Area

The loop area can be set by moving the loop bar tokens. The tokens can only be moved on the x-Axis, as it represents the time, and the loop itself is a time related setting of the system. The end loop bar token can not be moved to the left of the start bar token, as they can not be moved on the y-Axis.

Answered Questions

The deletion of all questions which stay in direct relation to the removed accompaniment will not be highlighted in this section.

- Software
  - Q: How can the system support the user, so that his/ her short term memory load can be reduced?

    A: The first come first serve functionality does help the user in not having to remember each token and its order in the token placement.
  - Q: How many tones can be played per beat?

    A: The system plays one tone per beat. Multiple tones will be sorted out by the first come first serve principle.
  - Q: Where is the suggestion displayed?

    A: The suggestion is displayed one beat after the last placed token.
- Hardware
  - Q: Should there be more than one pair of headphones?

    A: It is planned to have two pair of headphones at the table. The position of the second pair of headphones is important, as it can imply unintended properties of the system.
- Input
  - Q: How can the user accept or decline suggestions?

    A: The user accepts suggestions by laying a token into the suggestion's visual representation.
  - Q: What input device should be chosen to adjust the volume?

    A: A turnable knob with a minimum and maximum settings will be used.
  - Q: Which input device should be used to adjust the playback speed?

    A: A turnable knob with a minimum and maximum settings will be used.
  - Q: How can the tokens help the user to understand the current system status?

    A: Token itself can't, but the visual feedback of the token and the first come first serve functionality can.
- Feedback

- Q: When is the suggestion displayed? What technical parameters trigger them?
  A: A suggestion is displayed after the current loop area has been run through four times.
- Q: What does the suggestion visualization look like?
  A: The visualization looks like the outlines of the tone-token.
- Q: What icons should be used for the visual representation of the sound volume?
  A: The two icons for the sound volume will be two speakers. One has one vertical half-circle in front of it, resembling the low output volume, and the other has multiple half-circles in front of it, resembling the opposite.
- Q: What icons can be used as a visual metaphor for *fast* and *slow*?
  A: A tortoise is used for the *fast* metaphor and a running rabbit for the *fast* metaphor.

The questions that have not yet been answered at this point in the work will be answered during the first user evaluation. The documentation of this evaluation is the task of the following section.

11.8 Evaluation

As with creating the mockups, the user evaluation used the same table, which will be used later in the project for the functional prototype (see Section 12.1). On this table, mockups were laid upon to simulate the interface and tangible interaction devices like in Figure 11.9. As audio feedback is crucial for this test, but an implementation of a user interface reacting to the participants input had not been done at this stage, the authors decided to mock the audio output simply by playing guitar according to the participants interaction. This meant, that one researcher play the guitar and the other researcher simulated the visual output of the system by placing and moving mockups on their correct positions.

The structure of this section is based on the events in chronological order. The first subsection describes the participants and their assignment to the approaches. The following section deals more closely with the evaluation process, the content and the focus. The third subsection is dedicated to the results of this evaluation and the last subsection treats the impact of the results on the working document.

11.8.1 Participants

As already described in Section 7.4, the participants of the first user evaluation were experts and non-experts alike. Three of the eight participants were experts with knowledge in the field of user interface design, usability and user research methods from the immediate surroundings of the institute on which the installation was built. It should be noted here, that none of the participants stood in directly relation to the thesis itself or its authors. The non-experts were asked to participate via hallway- recruiting. The consent form of which the participants were asked to sign can be seen in the appendix (see Appendix A.2). The participants were between the age of 23 and 50. Except for one person, all of the four male and three female participants were students. Three of them were musicians and three of them weren't. The user interface experts were all male in the age between 34 and 40. The duration of the test sessions were between 18 and 37 minutes, whereby the experts needed much longer, due to their more detailed feedback. Each participant used a maximum of two interaction methods. The breakdown can be seen in Table 11.1.

Participant Nr.	Design Approach 1	Design Approach 2
0	Blocks	-
1	Drawing	Cylinder
2	Blocks	Drawing
3	Blocks	Cylinder
4	Drawing	Blocks
*5	Cylinder	-
*6	Blocks	Drawing
*7	Drawing	Blocks

Table 11.1: Mockup test participants (* were UI experts) and design approaches.

After the participants have been described anonymously, the following subsection will go into more detail about the procedure and the exact focus of this evaluation.

11.8.2 Procedure

After signing the consent forms and getting an explanation, users were left to interact with the system. If a user couldn't find an answer to emerging questions all by himself and it was a fundamental issue, the researchers gave hints and only gave clear answers when the user apparently couldn't find an answer by him-/herself. At the beginning of the evaluation, no specific instructions were given. In case the participants got out of ideas, the researchers asked them to complete a predefined task they hadn't finished yet. The tasks were as follows:

- Create a melody of at least ten tokens
- Create a melody of as many tokens as you like with different tone duration
- Change the tone duration of a token already in use
- Create a pause between two tones
- Change the volume
- Change the tempo
- Accept a recommendation
- Decline a recommendation
- Modify the loop area

It can be seen, that the created tasks have been derived from the current use cases. Apart from the execution of theses tasks, the focus lied upon the fulfillment of design principles and usability characteristics as well as enabling the defined design attitudes (flow state and playfulness) to the user. As soon as some questions, which could give information about the success of one of them, couldn't be clearly answered from the users behavior and thinking aloud, the user was asked to give his/ her personal opinion regarding these (then differently pronounced) questions after the interaction time had ended. The questions themselves were categorized according to the

mentioned design principles and attitudes. The questions can be seen in Section C of the appendix. As a result of this question, longer conversations were held, especially with participating experts.

The results of this user test procedure and the questions are part of the following subsection.

11.8.3 Results

The first result of the test had something to do with the participants themselves. In the previous work, non-musically trained people were referred to as humans who can not play an instrument. Even if that applies to humans, they can assign certain symbols to music. For example, notes are also recognized by non-musically trained people as notes, since almost every Central European got in contact with music notes at school. Thus, certain symbols can also be used for non-musically trained people.

First the blank approach with cylindrical token. Adjusting the tone length using the round tokens was not understood by any of the participants. In retrospect, this is probably related to the poor affordance of the mockups used for the tone-tokens. An appropriate marking on the mockups, as well as a clear, cylindrical shape would have supported the user in a much better way. This assumption is reinforced by the fact that the slightly cylindrical tokens of the loop area were attempted to be rotated by two people, although the markers on the token consisted of two horizontal arrows pointing away from each other.

The same approach with the block token was quickly understood by the participants. Only the behavior of the system in case of vertical oriented tokens has to be more transparent for the user. This is going to be achieved by the visual feedback of the system reflecting no rotation at all, as it indicates how the correct orientation looks like and how the system views the token. One expert advocated a more understandable physical metaphor and association between sound and token.

The drawing approach was the approach which input tokens were very fast and always understood. The separation of pen and eraser doesn't seem to be necessary, because it seems to hinder the user in the creation process and can rather throw her/him out of the flow. Another advantage of merging pen and eraser is that it prevents additional edge treatments by preventing the pen and eraser from being used at the same time. Direct editing of the tone length was suggested by one participant. Deleting and re-creating a tone does not work properly, seems unnecessarily complicated and should therefore be redesigned. Different interactions between pen and sound (holding the pen on the sound for a long time, double-clicking on the sound, etc.) can, for example, trigger different actions. The difficulty behind this is that there is no real affordance for these interactions. On the other hand, most people already know this kind of interaction from the daily use of touch displays. The question is, whether these interactions can also be connected to the installation during the course of use, mainly because there is no usual touch display in use. Older users could be even more affected by the lack of affordance, as these interactions are not so naturally the result of the use of a technical object.

After presenting the results of each approach individually, the approach overlapping results are next being discussed, beginning with the suggestions. The visual presentation of the suggestions was interpreted as a suggestion by only one participant. Also, the behavior of the suggestions when they appeared was not transparent. If the suggestions were perceived, they were perceived as disturbing rather than helpful. One suggestion from a participant was to give feedback when the user ignores or accepts a suggestion.

The orientation of the users on the interaction surface also posed problems. It was unclear at what height a new tone starts where a new beat would begin which made the creation process

unnecessary difficult for some of the participants.

Also, the use of the first come first serve approach in determining the active tones was not ideal and was also only partially understood. As an alternative, the opposite last come last serve approach was suggested by an expert. This is more practicable, since the user has to remember the order of the tokens set-in the first come first serve approach, in the case of oblivion much has to be tried out and this can quickly lead to frustration. First come first serve would thus only be useful by using a visual representation of the tokens order of activity.

The setting of the playback speed and the volume were generally understandable for the participants. Also, the loop area was not a problem, though the loop bar tokens were moved up and down several times at the beginning of the tests on the mockups of the respective loop bars. The number of beats available to the users was neither too much nor too little for any of the users. That the token themselves were metaphors for single tones was never mentioned. A participant was looking for a way to pause the loop's loops. A circumstance that must be observed in the iteration of the second design cycle.

The following additional peculiarities could be recorded in the course of the user tests: A person tried to use touch as an input method, but also only relatively late and not at the beginning of the test where more likely to test the ability to use. Another person initially did not understand the allocation of pitch and time to the respective axes, but was able to solve the problem all by himself after a short time. Only one person tried to create a familiar melody (all my ducklings).

Playfulness and flow state can not be achieved in the current implementation because there are too many inconsistencies and interface design issues. The aim of future evaluation is thus to achieve these design attitudes.

11.8.4 Repercussions

Again, starting with the approaches, the blank approach with cylindrical tokens was not as clear for the users as the same approach with rectangular token. This might, as already be stated, be because of the suboptimal selection of mockups for the cylindrical token. The drawing approach looks promising, but is confronted with types of interaction without possible affordance. Also the mentioned possible problems for older users cannot be completely dismissed at this stage. Furthermore, it is unclear whether the pen in combination with the used (non-touch) display behaves as the users might expect from (e.g.) a touch display. The pen must also be physically attached to the installation to prevent theft. The extent to which this makes interaction more difficult for the user is also unclear. The drawing approach is thus confronted with many uncertainties and risks. Since the blank approach with the rectangular tokens was also very well understood and there were no problems with the interaction, the researchers choose this form of token for the further course of the thesis. This definition is recorded in the use cases and thus the user requirements (see Appendix D.4 and E.4).

The fact that non-musically trained people can also recognize notes as such, even though they cannot read the note values, but can create the association between notes and music, also holds the possibility of using notes as icons. For example on the tone-token of the selected blank approach.

As can be seen from the results of the evaluation, the suggestions have to be changed. At the moment these changes are still unclear, but will be redefined in the following design iteration. The changes are reflected in a use case and thus also in a user requirement.

Also, the first come first serve algorithm for defining the activity of multiple tokens per beat needs to be changed. The choice of this algorithm unfortunately achieved exactly the opposite of what was intended. The users had to remember the order of the tokens or handle the tokens via trial and error. A very frustrating circumstance. The last come last serve algorithm, on the other hand, eliminates these problems by the elimination of the need not memorizing the token's order and giving the user visual feedback immediately. This also makes the system behavior of the installation more understandable and prevents frustration. The use of the last come last serve algorithm is documented as a functional requirement. Since none of the users expressed a positive or negative opinion about the possibility of playing one note per beat, the opposing approach, i.e. one note per beat and string, is implemented and evaluated in the following design iteration, as it may lead to a scalable difficulty for the playfulness.

Another problem that turned out to be frustrating for the user was the orientation on the interaction surface. Here, too, visual feedback in response to the user's input was missing. Due to this lack of feedback, the user feels left in the dark, it seems as if his/ her actions have no effect. A circumstance that must be prevented with the implementation of an orientation aid for the pitch and beat position of the tokens. This change is also documented in the form of a functional requirement.

Confirmed approaches of the first evaluation concern the loop approach including the selected number of bars or beats and pitches.

CHAPTER 12

Second Iteration

Author: Raphael Kamper

After evaluating the first iteration, we built a prototype and conducted a user test to verify or disprove our so far assumptions. This chapter deals with the design of the prototype and the according user test. Based on the findings so far, we decided to follow the *Blank Approach* discussed in the previous chapter. The note suggestion has to be redesigned and instead of a first come first serve system, we decided to implement a last come last serve algorithm regarding multiple tones per string.

12.1 Prototyping

This section describes all components of the MUI, which were either new added to the installation, or have been improved in the course of the second iteration. Because of the fact that the connection between these components is hardly to follow without a visualization, the authors have added a visual representation of all components and their connections between each other (see Figure 12.1).

The MUI prototype consists of four categories, which are the same as the ones being used for categorizing the questions derived from the use cases: *Software*, *Hardware*, *Input* and *Feedback*. The following subsections are named after these categories and contain the description of every component, which can be assigned to the corresponding category.

12.1.1 Software

Based on the decision to develop a tangible user interface with unique tokens and the lecture reports from the previously built table, its wide spread community and cross platform support, we chose to use the reacTIVision framework[1] maintained by Martin Kaltenbrunner. To get a better picture of all components and how they are linked together, a visualization of the connections between the components can be seen in Figure 12.1. The order of this subsections devisions is based on the order of the software components in the pipeline.

[1] See the reacTIVision GitHub repository.

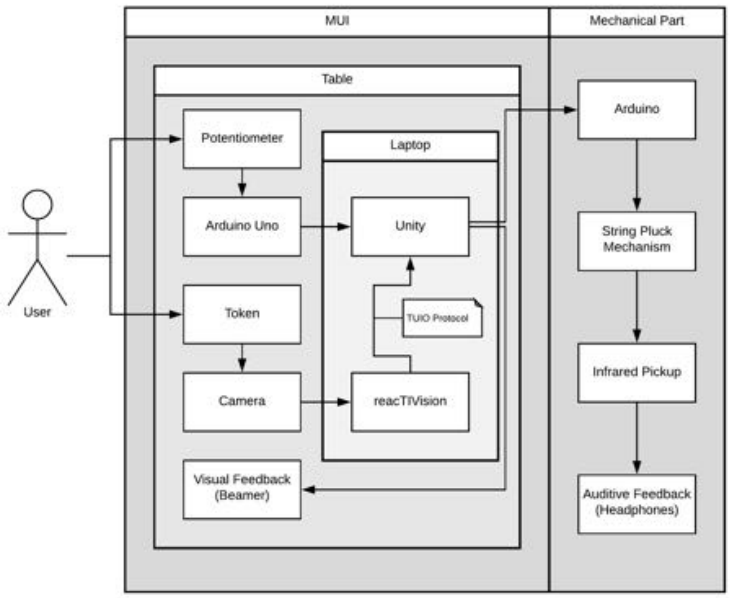


Figure 12.1: Connections between all components of the MUI.

12.1.2 reacTIVision Framework

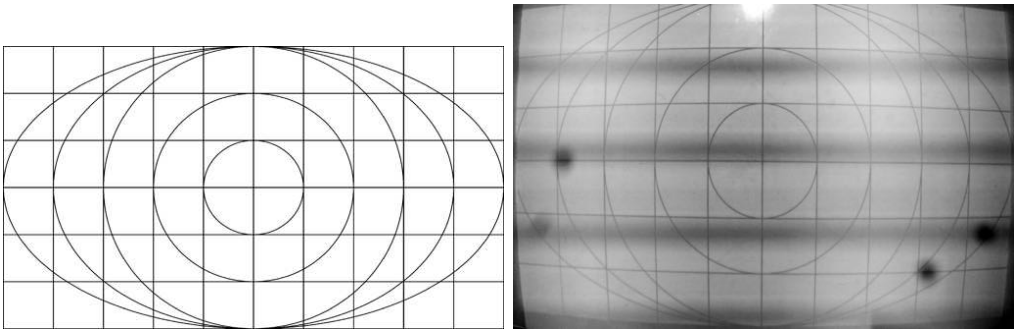
We started the development on both macOS and Windows machines using Xcode and Visual Studio. Therefore, we followed the installation instructions provided on GitHub. We cloned the latest (version 1.6) reacTIVision repository, and migrated to the latest visual studio version (2017), which led to library dependency issues, solved by also updating the included SDL library. On macOS no further adjustments were necessary. When running reacTIVision, a configuration file needs to be created to compensate perspective distortion, set the correct format and image resolution or exposure and contrast values amongst others. Our configuration files are in our GitHub project[2]. To create the calibration file we removed the infrared passing filter (otherwise the reflection cannot be detected) and projected the calibration grid (see Figure 12.2a) on the translucent film. We chose an image resolution of 1920 x 1080 pixel, and therefore, used the provided wide calibration file, which needed to be resized to fit our set resolution.

The resulting camera image is shown in Figure 12.2b and the according adjusted grid correction overlay can be found in Figure 12.3a. The black dots are image artifacts whose origin source we could not find, but this did not result in any tracking problems.

Figure 12.3b shows the image including two markers already placed on the playing field. The following Figure 12.4a shows the image with the infrared passing filter applied, in Figure 12.4b the processed black and white image is shown.

The reacTIVision framework uses a client server architecture using the TUIO protocol[3], described in the following section, and encodes the messages in the Open Sound Control (OSC) format (see Subsection 4.4.3).

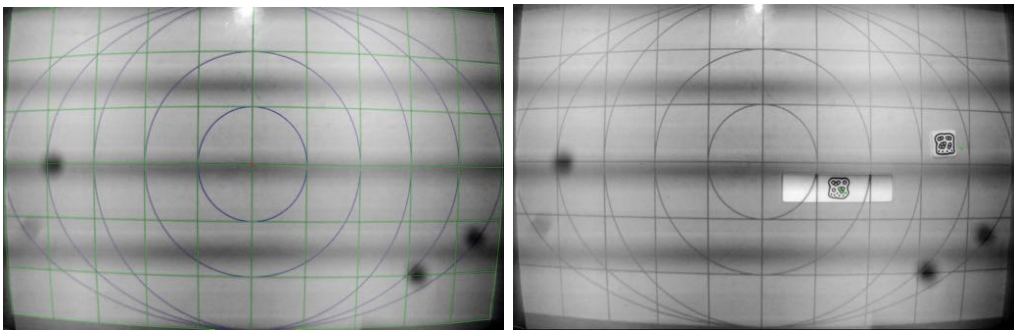
[2]See configurations directory for configuration files and calibration grid.
[3]See project's website.



(a) Wide calibration grid.

(b) Camera image without infrared passing filter.

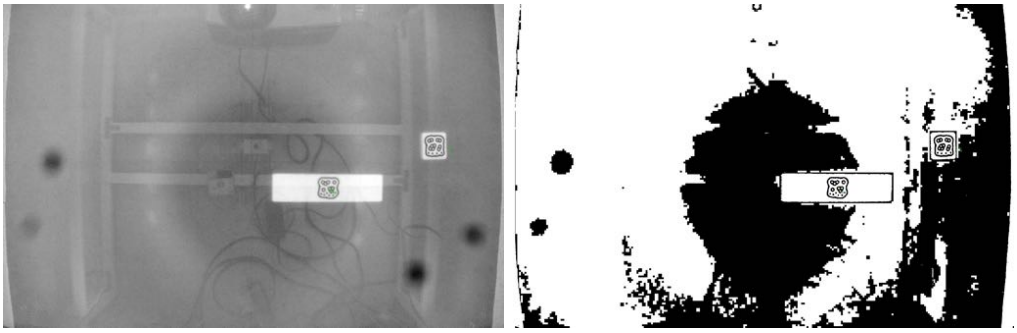
Figure 12.2: Calibration grid file and projected image.



(a) With green grid correction overlay.

(b) With two markers.

Figure 12.3: Camera image of calibration grid without infrared passing filter.



(a) From camera with infrared passing filter.

(b) After procession.

Figure 12.4: Image of calibration grid.

12.1.3 TUIO Protocol

Kaltenbrunner et al. introduced the TUIO Protocol in their conference paper *A Protocol for Table-Top Tangible User Interfaces*[122] based on experiences resulting from the work on the

reactable. Principally, it is designed to support the tracking of finger gestures, blob objects and tagged objects. For our purpose we only used tagged objects, but the option to extend the interaction forms was considered a big advantage for potential following iterations or redesigns. We do not want to go into too much detail about the TUIO protocol. When an object is detected, the protocol supports, amongst other information, a temporary session id, the actual marker id (called class id), position on the (previously calibrated) grid and according rotation and movement vectors per marker.

There are simulators available[4] for testing a client instead of building a fully functional table first and being bound to the table's location. We used the TUIOSimulatorFX[5] (see Figure 12.5a) and decided to use the Unity Game Engine, described in the following subsection, for software prototyping. The input from the TUIOSimulator and the corresponding output from Unity3D can be seen in Figure 12.5.

12.1.4 Unity3D Game Engine

While a Unity3D[6] application seems like an overkill from a software engineering perspective at first hand, but we carefully chose to use it. There are multiple Unity3D-Application frameworks listed on the TUIO's project website, special shader effects such as glowing are easily implementable, and the serial communication with the Arduino boards is easily handleable using custom C# scripts. These scripts enable to write easy and clean code. Additionally, the physical effects of the engine could also be used in this project. For example, the speed of the bar which indicates the current position in the melody, and thus the playback speed, was regulated using velocity (see Code Listing 12.1).

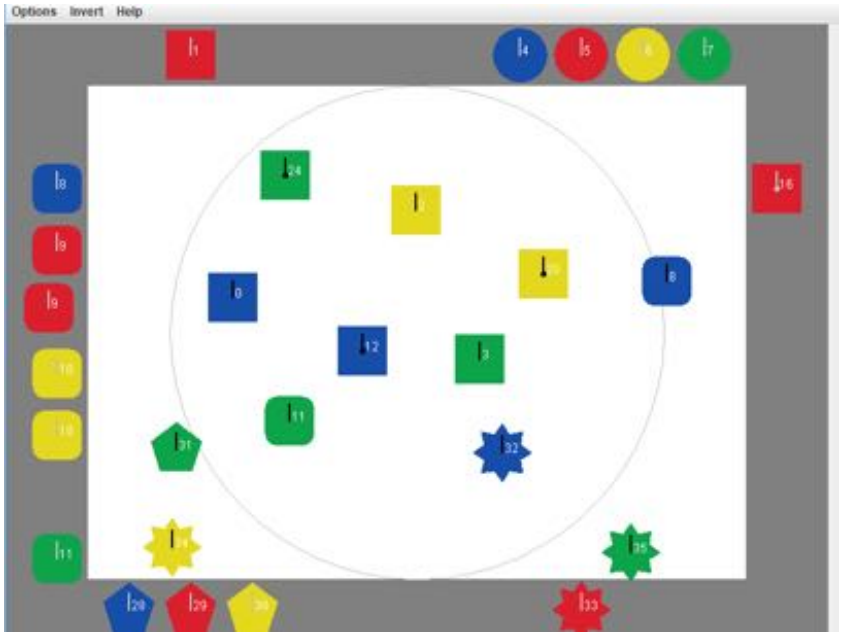
```
void FixedUpdate()
{
    bpm = bpmManager.getBpm();
    m_rigidbody2D.velocity = new Vector2((cellWidth * bpm) / 60, 0);

    //sets position to startBar position if it's x position is below
       the startbar OR if it's x position is above the endbar
    if (this.transform.position.x < startBarPosition.x ||
       this.transform.position.x > endBarPosition.x)
        this.transform.position = new Vector3(startBarPosition.x,
           transform.position.y, transform.position.z);
}
```

Code Listing 12.1: Simple Unity3D C# Method Example

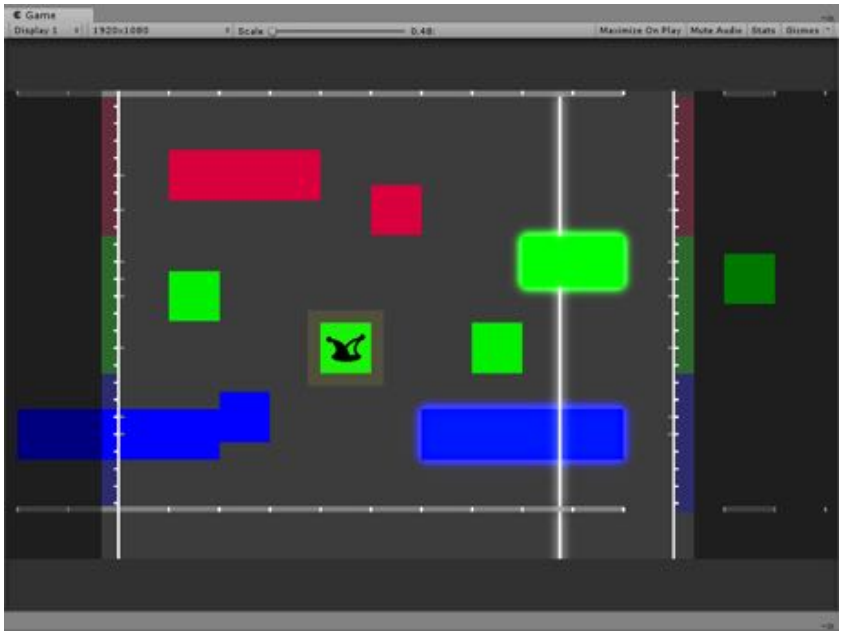
Apart from the fact, that there is a big community using Unity3D, and therefore the existence of countless tutorials for using this game engine, Unity's GUI is very intuitive and enables a fast an efficient work flow. It should also be stated that one of the authors already had experience in using Unity3D in the areas augmented reality and game design.

Two functionalities shall serve as examples for the implementation in Unity3D, *Last Come Last Serve* and *Snapping*. Since last come last serve is based on snapping, snapping will be described

[4] Listed under the tuio.org application frameworks.
[5] See GitHub repository.
[6] See Unity3D website.



(a) TUIO Simulator



(b) The *Play Mode* of Unity3D

Figure 12.5: TUIO Simulator and the corresponding Unity3D Scene

first. Snapping is not applied as long as the token in question moves on the interaction surface. However, if the token lays still, the system calculates the nearest pitch and beat based on the current position. The x-position of the marker is then set so that, depending on the assigned tone

length and thus visual representation, the beginning and end of the visual feedback matches the correct beats. When calculating the y-position, the visual feedback can be left out because the center of the marker must match the calculated y-position. If the token is moved again, snapping is deactivated and the visual representation is again displayed directly underneath the token.

Once snapping is applied, the system remembers the time of snapping, the position, and the string that was snapped on. For each string exists an array with as many positions as there are beats. The visual representations, of the most recent token per beat is then displayed in color, whereas all other visual representations are grayed out. In order to counteract jitter from image recognition, a threshold is implemented that prevents new snapping (and thus setting a new time) without user input.

It should be noted that for both implementations there are many other approaches, such as using colliders[7]. But, as already stated, using Unity3D for the visual output of the system was very convenient. The following subsection covers a description of certain elements of this output.

12.1.5 Hardware

This subsection's purpose is to describe every hardware part of the system. Each part of the MUI has its own subsection, starting with the housing of all the other hardware components: the table.

12.1.6 Table

The table was constructed during Lukas Pichlhöfer's bachelor thesis[8,9] and was the basis of the MUI in this thesis.

Despite the table's wooden frame construction it consists of a 12 mm acrylic plate and a translucent non-reflecting polyester foil. Inside the table a video projector[10] is placed to project its output on the polyester foil. The original construction contained two cameras and used Community Core Vision[11] as tracking software. We wanted to use another framework for the tracking process (see the following Subsection 12.1.2), currently supporting no image stitching, but running multiple instances (one instance per camera with a config file specifying the according camera) seems to be a workaround[12]. The table also includes infrared lightning fixtures for providing a uniform illumination, supported by aluminum foil attached to the wooden walls.

One of the walls can be opened by unlocking the closure system to plug in or out the power supply, and so turn on the projector. It is worth mentioning that the day before the user evaluation took place, we extensively tested the whole installation for hours, and ultimately the beamer overheated and performed a precautious shutdown. During the evaluation day, we put away the removable wall to provide a sufficient air circulation for cooling down the beamer.

[7] See Collider documentation.

[8] The thesis *Dragon's Lair - Ein selbsterklärendes Gesellschaftsspiel auf Basis eine TUIs* is accessible at the Visual Computing and Human Centered Technology institute's library in German.

[9] Instructions to built an alike table are also provided on the reacTIVision project's sourceforge page.

[10] BenQ MW870UST model used in particular.

[11] See CCV website.

[12] According to a forum post by Martin Kaltenbrunner on the reacTIVision project's sourceforge website.

12.1.7 Camera

As stated above, we only wanted to use one camera, therefore, we used a different wide-angle camera[13], centered it within the table, and lowered the mounting point within the table. For easing a spatial camera adjustment, we placed the camera on aluminum rails, allowing corrections on the y-axis (the camera was placed exactly at the center of the x-axis). We 3D-printed a mounting case for the camera, fixed by four screws onto the rails (see Figure 12.6). Under every screw we placed metal springs to allow fine-tuning by adjusting the z-axis enabling a horizontal alignment of the camera. Additionally, we put an infrared passing filter[14] on top of the camera mounting. This way the projector's image is filtered and objects placed on top of the acrylic plate on the polyester foil can be tracked.

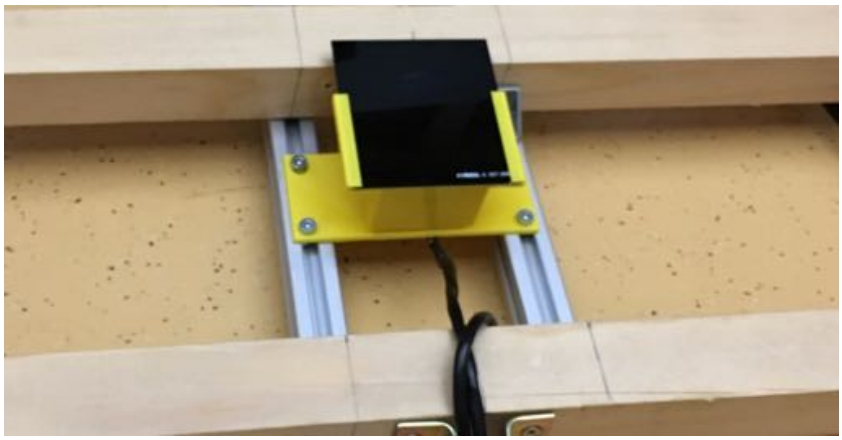


Figure 12.6: Mounted camera with infrared passing filter on top.

12.1.8 Laptop Computer

To run the tracking software, process the camera input and provide the visual output for the projector, a standard laptop[15] is used. The camera is attached via USB port and the video projector via the built in HMDI port. Additionally an Arduino Mega and an Arduino Uno were connected to the Laptop via USB ports. Details describing the used software and communication follow in Subsection 12.1.1.

12.1.9 Input

In this subsection we describe the process of designing the input parts of the tangible musical user interface we built. They consist of note token, joker token, loop bar token and a potentiometer for tempo regulation. The following subsections are arranged in the same order.

[13] We used a 5MP 170° angle USB camera. For further hardware details see the manufacturer's website.
[14] Specifically, a Cokin infrared B89 A007 series filter.
[15] We used an Acer Aspire A515-51G standard version model in particular.

12.1.10 Note Token

After setting up the reacTIVision framework (see Subsection 12.1.2), we started to test different marker forms. ReacTIVision comes with two symbol sets *amoeba* and *yamaarashi* (see Figure 12.7), but also allows self designed symbols following the fiducial design specified by Bencina et al.[123]. 215 amoeba and 299 yamaarashi marker ids are available, and seemed more then sufficient for our purpose.

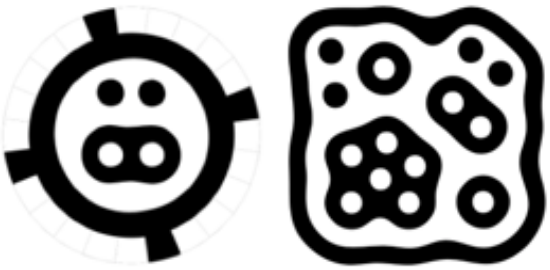


Figure 12.7: Fiducial symbols for maker id 0 in yamaarashi (left) and amoeba (right) representation.

The fiducial markers are created by a genetic algorithm[123] and are interpreted as a rooted tree, whereas, starting from the outside, every color change is represented as a new child (see Figure 12.8). Additionally, the center weight of the black and white leaves are calculated to get an orientation vector (see Figure 12.9).

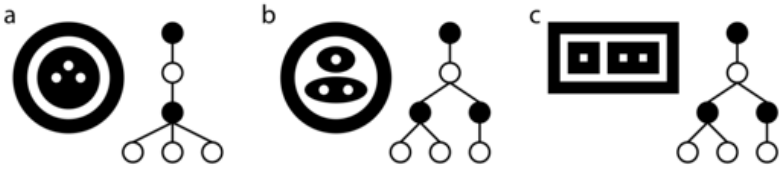


Figure 12.8: Example markers and their tree representation[123, p. 3, Figure 2]

.

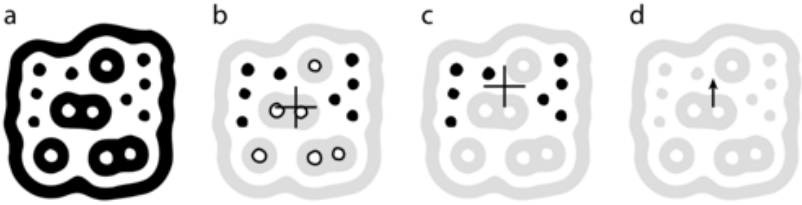


Figure 12.9: Calculation of the orientation vector[123, p. 3, Figure 3].

We printed both the amoeba and yamaarashi symbols on paper in different sizes to find limitations regarding marker size[16]. We limited the maximum marker id to 37 and only test markers within

[16]It is worth mentioning that a camera with a wrongfully adjusted focus might result in perfectly fine

the range of 0 to 37. Within this range we achieved better results using the yamaarashi markers when resizing the markers to 30 x 30 mm[17], but could not find any differences when using the final size of 35 x 35 mm. As the amoeba markers are commonly used, and more experiences are shared online, which could potentially support faster troubleshooting, we decided to use the amoeba markers. To get a first hands on experience, we used wooden blocks of different sizes, 3D printed the markers in different sizes and used paper printouts as shown in Figure 12.10.

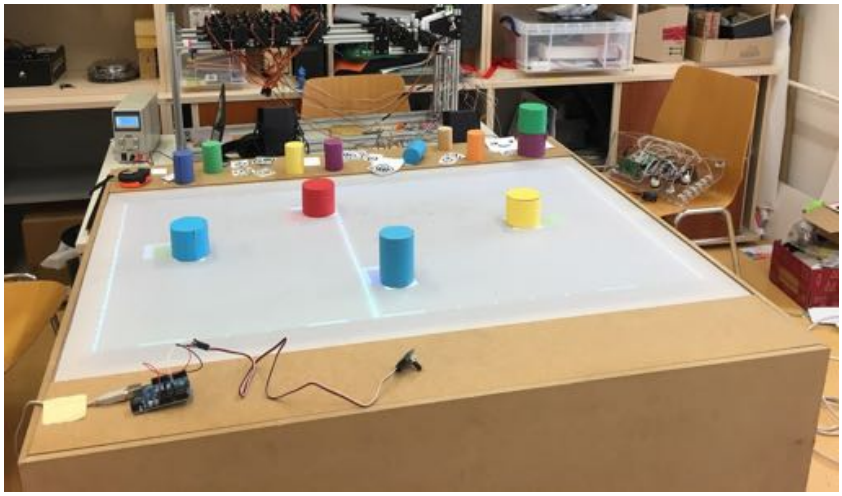


Figure 12.10: Wooden blocks and paper prototypes to test different marker sizes.

We figured out that a size of around 30 mm is quiet handy, but to efficiently use all available space, the size is also determined by the length and height of the playing field and the minimum available beats. The image projection of the beamer on the polyester foil is about 960 x 640 mm. Dividing the length of 960mm by 16 beats, as we designed it in the first iteration's sketches, (see Subsection 11.6.1) results in 60mm per token. As the placed token on the table should be highlighted by a colored marking, the marking's space has to be taken into account. We experimented with 3D prints and finally decided on a marker size of 45 x 45 mm, using about 35 x 35 mm for the fiducial symbol and $\frac{60-45}{2} = 7.5$ mm space on each side for marker highlighting. The height of 640 mm leaves space for $640/60 = \lfloor 10.66 \rfloor$ mm tokens. This is clearly not enough as the mechatronic guitar installation supports ten frets alone per string. Therefore to regulate tone height we made a finer subdivision into 24 (8 frets per string) pitches. In other words we created a two-dimensional array of 16 beats and 24 tones (this plays a role for internally mapping the note positions as described in Subsection 12.1.4). The tradeoff with this approach is that $640/24 = 26.66$ mm won't allow the placement of two half tones above each other. While this is not a problem on one string, as only one fret can be played per string, it means that the highest fret and the lowest fret on the next higher string could not be played at the same time. When supporting at least five frets, using a standard guitar string tuning this is no restriction in the sense of available pitch combinations, as the 5th fret (or the 4th on the B-string) already has the

tracking results if the fiducial markers are big enough, but when resizing the markers, not considering the camera focus (because it "already worked"), leads to seemingly random results and may not be so easy to troubleshoot.

[17]The smallest prints were 25 x 25 mm, but those led to flickering tracking results, especially during marker movement. This is an exclusion criteria when implementing a last-come last-serve algorithm.

same tone height as the higher next open string. It may create the impression of a restriction to users that are, if not guitar players, most likely unaware of this fact.

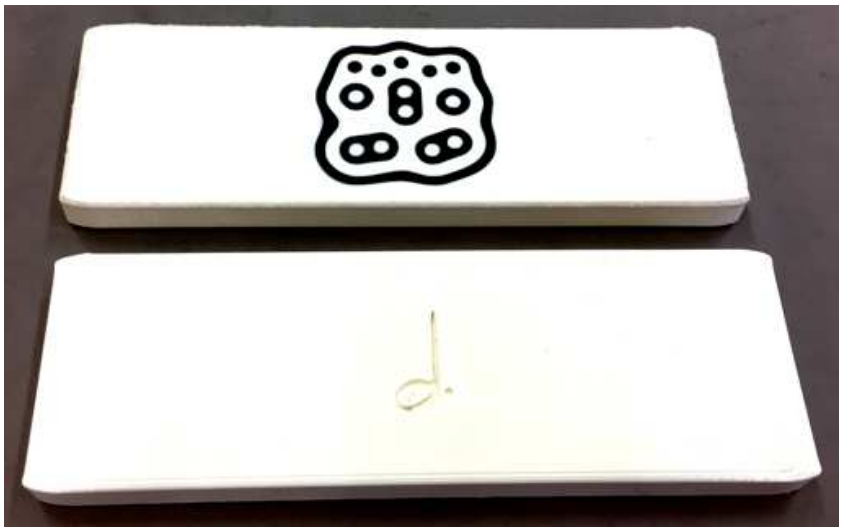


Figure 12.11: 3/4 note marker bottom (top) and top (bottom) view.

In total we designed 38 tokens of three categories (note, joker and loop) with a height of 10 mm per token, 30 note markers in different sizes according to their note values, namely 12 x 1/4 (45 x 4 5mm), 8 x 2/4 (90 x 45 mm), 6 x 3/4 (135 x 45 mm) and 4 x 4/4 (180 x 45 mm). Every note token has a note symbol on top (see Figure 12.11) reflecting its value (based on a result of the evaluation from the first iteration). We 3D printed most of the tokens using a dual extruder printer with black and white standard 2.85 mm PLA. Additionally, we printed some tokens with two other single extruder printer (one from the university, the other form one of the authors) using white material and modeling the fiducial symbols hollow, then coloring it by hand with a black permanent marker[18].

12.1.11 Joker Token

One finding regarding the note recommendation (see Subsection 11.8.3) was that participants were generally confused by the appearing visualization at first. It is hard to determine a good moment for a hint without disrupting the user, and we wanted the suggestions to be non intrusive. Therefore, we redesigned the interaction by leaving this decision to the user. We created joker tokens that differed from note tokens in having a joker symbol on top instead of the note symbol (see Figure 12.12). A user can place the joker at any time s/he wants onto the playing field at an arbitrary position. This results in a note suggestion at a given beat, visualized as a string dependent colored square with a joker symbol in it (see Figure 12.5b).

[18]We were forced to use this method as the dual extruder printer failed catastrophically while printing a token, permanently damaging the print core container. Although we were finally capable of temporary fixing it with duct-tape, it took several hours to remove the hardened PLA material, and therefore, we started using the single extruder printer as a precaution.

Figure 12.12: 1/4 joker token top view.

We decided to implement an algorithm choosing random tones from a pentatonic scale (see Subsection 4.1.9) as this is a very simple approach. Although the resulting tone might not be perfectly suitable to the tones before, if the user places multiple joker tokens in a row, this leads to a somewhat good sounding melody. Multiple jokers per beat are possible, resulting in randomly choosing tones from the remaining strings. Due to the distances and use of the pentatonic scale, it does not result in a dischord, but also leading to an acceptable sound. Out of the total 38 tokens six were joker tokens, specifically 2 x 1/4 (45 x 45 mm), 2 x 2/4 (90 x 45 mm), 1 x 3/4 (135 x 45 mm) and 1 x 4/4 (180 x 45 mm).

12.1.12 Loop Token

During the mockup test it became clear that the loop tokens should not be placed somewhere on the playing field (see Subsection 11.8.3). Nevertheless, we still experimented by simply placing loop tokens anywhere on the field, and in fact implemented this behavior in the musical interface's software. In the end we decided to construct a wooden frame, spatially separating the loop tokens as shown in Figure 12.13. Wood was chosen as a material to reduce the fear of interacting with the elements, as wood is not associated with electronic, conductive parts, and cannot be destroyed by simply touching it.

This should prevent the user from moving the loop token alongside the displayed loop bar, which would not change the system's state or lead to any feedback, and therefore, has the potential of confusing the user. This could be especially the case if the user rotates the token, resulting in the arrows indicating an intended vertical movement. To eliminate such useless interactions with the loop tokens, we created the wooden frame. Out of the total 38 tokens, two were loop tokens with dimensions 45 x 45 mm and a triangle within a cut out circle. The loop-tokens are placed into the wooden frame in a way that the start loop token's triangle top looked to the left and the end loop token's triangle to the right, when standing in front of the table. The wooden frame consists of two 100 cm wooden sticks with a right angle profile. They are simple stuck into a 3D

Figure 12.13: Start loop token within the wooden frame.

printed fixture clipped onto the table walls. The position of the tokens defines the active loop area, within beats are constantly replayed. Elements outside of the active loop area are darkened to indicate inactivity as sketched above in the blank approach (see Subsection 11.6.1).

12.1.13 Potentiometer

For tempo regulation we used a potentiometer. We 3D printed a case and knob with a turtle and a rabbit as iconic labels to indicate the tempo. Minimum tempo is 60 bpm and the maximum tempo is 200 bpm as stated in the mechatronics part introduction (see III). The potentiometer (see Figure 12.14) has a radius of 300° and is connected to an Arduino Uno where the output values are sampled ten times, mapped accordingly to the 60 - 200 bpm range, quicksorted, and the median (to get more stable results) of the ten measurements is sent to the laptop via USB serial connection[19]. The reason for using a second microcontroller, only handling the potentiometer, lies within the logical separation of the musical user interface from the mechatronic installation.

12.1.14 Feedback

Due to the fact that the used feedback, apart from the auditory output of the generated melody, is almost exclusively limited to visual feedback (see Figure 12.1), this subsection focuses on the visual feedback created in Unity3D. The few haptic feedback is discussed at the end of this subsection.

[19]The code can be found on GitHub.

Figure 12.14: 3D-printed case and knob for potentiometer.

First, there are some mentionable facts about the chosen colors that are used as string colors in the interface: the colors were chosen with the help of an online tool[20], which supports the color selection process in consideration of visually impaired people. Although using a 3000 lumen video projector (see Subsection 12.1.6) at daylight and artificial lightning from above, it displays the colors in a slightly distorted way. These colors are used for the rectangular visual representation of active markers and on both loop bars, facing away from the loop area and indicating which part of the y-axis is assigned to which string. The visual feedback of the token differs slightly for joker token. Here an additional light yellow rectangle is added to the representation to distinguish the joker from other tone-tokens when the token lies in the center of the visual feedback (see Figure 12.5b).

In order to give users the opportunity to orient themselves on the interaction surface, so-called orientation lines have been implemented. There are two types of orientation lines, the outer orientation lines and the token orientation lines (see Figure 12.15). The outer orientation lines are held in white and positioned at the loop bars and at the top and bottom of the interaction surface. Due to the positioning on the loop bars, the outer orientation lines are always visible at the edges of the current loop area. Each mark on the loop bars stands for one pitch available to the user. The space between to marks on the top and bottom of the interaction surface resembles exactly one beat. The loop bars snap to the same x-position the white mark is placed, which is why Figure 12.15 does not show a mark at the beginning of the loop bar.

The second kind of lines for orientation are the already mentioned marker lines for orientation. The purposes of these lines are the visual highlighting of currently moving markers (see Figure 12.15a), pitches, and beats on which already a token lies (see Figure 12.15b). The visual highlighting of a token only takes place when the token is moving. When this happens, the token lines for orientation, which mirror the markers x and y position, get twice as thick as when the token lays still.

The is another visual feature attached to the loop bars, which has been mentioned in another context in Section 11.6. This feature is a dark but transparent surface that darkens all interface

[20]The chosen colors on DavidMathLogic.

(a) When moving. (b) When snapped.

Figure 12.15: Lines for Orientation

elements that are outside the loop area, having the aim of drawing the user's attention to the current loop area (see Figure 12.5b).

Furthermore, there are some special cases that can occur either between the current-location-bar and the loop bars, or the current-location-bar and the token. The first special case occurs as soon as a loop token is moved and the current-location-bar would pass the moving loop bar. However, since this can happen between each beat and can interfere with the damping of the notes, for instance, another bar appears at the old loop bar position while it is moved. This slightly different looking bar visualizes the current start or end of the loop range. The other special cases occur when a tone token is placed on, or removed from a beat that the current-location-bar passes at the same time. As the tokens move away, the sound is first played to the end, leaving the visual representation in place until the sound ends. If the token is placed on a beat that the current-location-bar is moving over, the token does not snap until the bar is no longer over the affected beat. This measure prevents problems with the last come last serve algorithm (see Subsection 12.1.4).

The glowing of the rectangles beneath the token and the bar, which resembles the current location in the melody, was achieved by the use of the free MK Free Glow[21] shader. Another visual feedback was the result of a certain 3D printing material (and infill percentage) used to print the tone-tokens. The glowing current-location-bar gets reflected on the inner top of the token, visually highlighting the remaining duration of the tone.

The haptic feedback of the tokens is not very meaningful at the moment. The material itself is light because it is plastic, thus conveying an easily replaceable and inexpensive impression. Since the tokens were printed with a 0.2 mm resolution of the 3D printer, the surface is slightly grippy. This is a circumstance that can be helpful when picking up the tokens from the interaction area. The icons are embedded as negative in the token, which is why they are slightly perceptible. There is no further haptic feedback on the token.

The used potentiometer has a smooth transition between the adjustable minimum and maximum values. The choice was between a potentiometer with a number of fixed values and the one used.

[21] The MK Free Glow Shader.

Since the authors did not want to set only four adjustable values, the other potentiometer was used.

12.1.15 Repercussions

The token design was set to include the note symbol or joker symbol for tokens playing a tone. As the tokens' design cannot support the user directly to understand the system, visual feedback such as highlighting colors and darkened areas were designed. The loop-tokens' design was chosen to contain a triangle within a circle. The triangles edge shows to the left or right side. Hardware components for the musical user interface were chosen with the already existing table including a beamer, infared LED stripes and a USB camera with an external infrared passing filter. The input components consist of the note, joker and loop tokens, as well as the tempo potentiometer. The software tracking the tokens is the reacTIVision framework, the graphical user interface parts are implemented in Unity3D.

The table height is 92 cm. That means it is not accessible to all people, like wheelchair users or children. The color scheme was adapted to support visually impaired people.

The note and joker tokens were placed within a carton box, and in order to highlight the potentiometer, it was put nearer to the center to keep it within the users' reachable area. Bpm values can be set continuous and are not split into several predefined tempo areas.

All changes were adapted in the form of updated use cases and requirements. Both can be seen in the appendix in Section E.5 and D.5 respectively.

12.2 Evaluation

We evaluated the prototype designed during this second iteration the same way as in the first iteration, but additionally audio recorded the users and created a log file.

Twelve Participants took part in the user test. One out of the twelve user tests was aborted, due to a software bug causing tones to be played although the according note token was already removed. We did a rollback to a (more) stable version after this incident. The difference lies in a known bug, where a tone is not played if only one beat is between the start and end loop bar, and the tempo is increased over specific bpm limit. We decided to use this version and interfere in that special case by telling the user this is a software bug. Due to this occurrence we only list eleven participants further on.

12.2.1 Participants

The eleven participants were in the age range of 26 to 34 years, four were female and seven male. Only two of them did not play an instrument at all. Played instruments were piano (5x), guitar (4x), drums (2x), bass (1x), transverse flute (1x) and violin (1x), as shown in Table 12.1. One of the users (participant 05) was a UI expert.

12.2.2 Procedure

The participants were handed a consent form (see Appendix A.3) and given time to carefully read it. After signing the form, we showed them the video cameras and audio recorder, started recording while explaining the thinking aloud method and asked them to verbalize their thoughts

Participant Nr.	Age	Gender	Music Instrument
00	28	m	piano
01	30	m	none
03	29	m	drums
04	28	f	piano, violin
*05	34	m	piano, drums
06	28	m	piano, bass
07	29	f	guitar
08	33	f	transverse flute, guitar, piano
09	26	f	guitar
10	28	m	none
11	27	m	guitar

Table 12.1: Participants (* was UI expert; [user test with Nr. 02 was aborted]).

accordingly. Although the participants were free to ask us questions all the time, we told them that we will not interfere in the first couple of minutes of their interaction time letting them explore the interface on their own. If a participant was seemingly stuck and did not know what to do, we gave hints in the form of tasks like trying to change the tempo, but only if the participant already interacted for a few minutes. There was no exact amount of time we set, because this is up to the user's motivation to solve a self set task, and as all users followed our thinking aloud instructions, we could decide on an interference depending on the current situation.

The first iteration's evaluation, in this case based on the expert interviews (see Section 11.2), suggests the use of headphones for such an installation. While this might lead to an interaction in a museum context and does not distract other visitors, this does certainly not apply to a user test. Therefore, we decided to use common speakers for audio output. This way the output can be recorded without further equipment and the researchers perceive the same auditory feedback as the users, simplifying the communication with them.

Tests were generally open ended and stopped by the users not interacting with the interface anymore. Afterwards all participants asked how the prototype works, mostly interested in the token tracking, and we gave an explanation. Figure 12.16 shows the prototype in the user test environment. The note and joker tokens were sorted by size and placed in two carton boxes. Speakers were placed at the table's end in front of the mechatronic installation. A black blanket was hung up to hide the shelves directly behind the whole installation to prevent distraction when focusing on the mechatronic part. The open lab certainly does not provide an ideal environment for simulating a museum's atmosphere. It seemed to us that users had no difficulties focusing on the user interface, but were distracted when viewing on the surrounding components such as the mechatronic part or the loudspeakers. The potentiometer was set to middle positions resulting in a bpm value of approximately 130 bpm.

Figure 12.16: Prototype test setup.

12.2.3 Results

We are aware of Rosenzweig's [46] statement that at least 20 people are necessary to evaluate quantitative data as we mentioned in Subsection 3.5.2. Nevertheless, we do not evaluate the log files alone to support or disprove our observation based statements, but in combination with the previous qualitative analysis. The presented figures containing data from the log files, are used to illustrate the statements, and we chose representative ones in this section, but all figures can be found in Appendix F - F.0.1. Following subsections present the results of the user test analysis. Each interaction element is described separately, concluding with overall observations.

12.2.4 Note Token

All users understood the mapping regarding time and tone height. The overall approach was to simply put a token onto the playing field and wait for feedback. Usually, as soon as the current-location-bar hits the note token, the x-axis is correctly interpreted as the time axis. The correlation of position on the y-axis and tone height was discovered by either moving the token along the y-axis or by placing an additional token on a different y-position with respect to the already placed token. Some, but not all, users tried to center the note token inside the visual representation. One participant tried to correct a dissonance this way. All users observed the note or joker symbols on top of the markers. Figure 12.17 shows the token position of played tokens by all participants. The high frequency of position eight, that is the first green token, may be caused by exploring the color and tone-height connection, also seen at position 16 for the red area.

The length of the different tokens was also understood by all participants, again following an explorative approach. Especially the non quadratic tokens were rotated or partly placed diagonal,

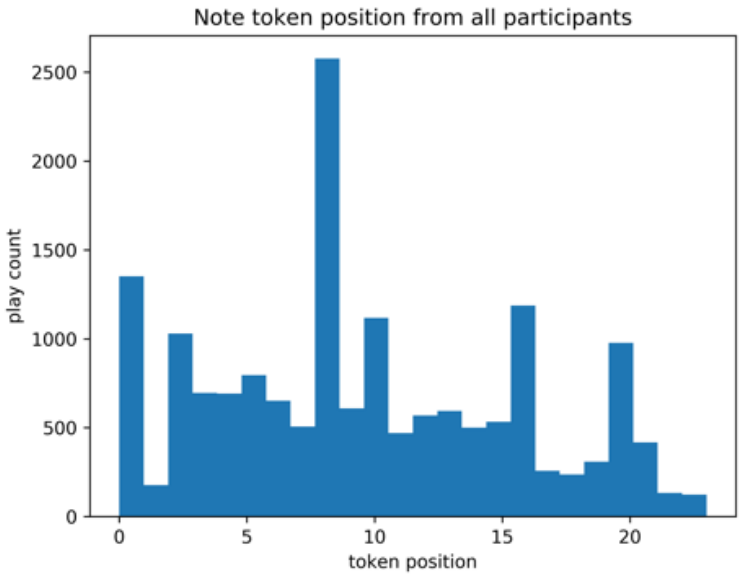


Figure 12.17: Histogram showing note positions of played tones for all participants.

with the expectation to play a melody getting higher or lower. One participant expressed the wish for even smaller tokens to support 1/8 notes.

The last come last serve system, including the darkened disabled tokens, was not registered by all participants from the beginning. Most users questioned the system first but accepted it as a fact soon. Those users noticed the mechatronic part beforehand, and drew the conclusion that this might relate to the strings. The same applies for the colors. While some of the users simply were satisfied with the concept of different colors, meaning different tone heights, those spotting the colors on the mechatronic part first, directly associated the strings with the according colors.

12.2.5 Joker Token

The joker tokens were referred to as "joker" by all participants. Only one participant, the UI expert, correctly described the behavior of the token. Generally, all other participants were confused and demanded an explanation from the researchers afterwards. Following quote sums up our observations:

> "I do have a problem to interpret the joker, I don't get it, I've had a clear expectation of what's happening [...] maybe a magic trick [...]" (Participant 10, translated by the authors from German)

Participants expected the joker token to result in a sophisticated feedback, either acoustic or at least in visual support, suggesting a melody for instance. Joker tokens were usually tried out (see Figure 12.18), then not used anymore, and by some users eventually tried out again.

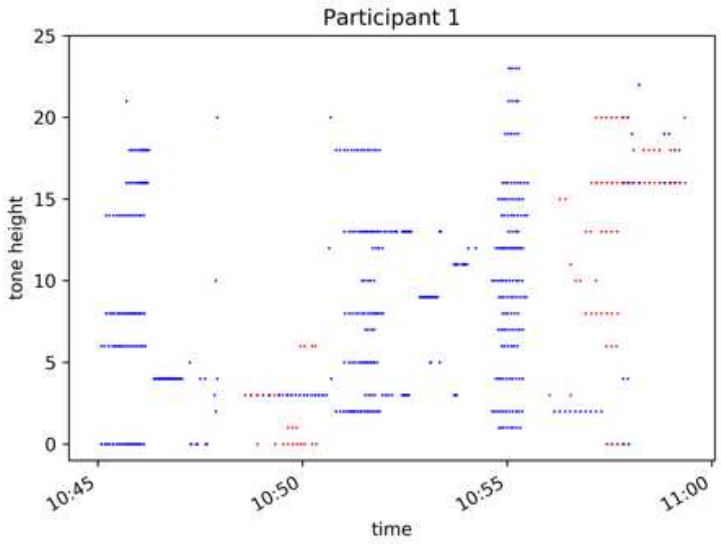


Figure 12.18: Scatter plot with blue note token and red joker token positions over time.

12.2.6 Loop Token

The loop-tokens were not noticed at all by some users. It often took hints from the researches asking a question like: "Are you satisfied with the amount of beats?" One participant interpreted it as forward and backward buttons and tried to press it. One participant placed note tokens within the wooden frame between the loop tokens. Due to a software bug that did not lead to a tone being played, but in servo motor performing the damping action, making a quiet but perceivable mechanic sound (passive auditive feedback as described in Subsection 3.2). That understandably confused the participant, trying to move the note token, replacing it with another token, to get the monotone result of the mechanic sound. One participant used the loop-tokens to create a pause instead of removing all tokens from the table. The average loop size in beats is shown in Table 12.2. Given T is the total interaction time and t the relative time span in which the loop size was x beats, the weighted arithmetic mean loop size $\overline{x}$ is calculated as follows:

$$\overline{x} = \sum_{i=0}^{n} x_i \cdot \frac{t_i}{T} \tag{12.1}$$

12.2.7 Tempo

While the turtle and rabbit icons were interpreted as slow and fast, many users did not notice the potentiometer at all, and only used it after the hints of the researchers. Participant 07 did not change the tempo at all. The log file analysis failed for participant 03 regarding the bpm values. While the participant in fact changed the tempo, there was a serial communication error logging exceptions only instead of bpm values. If a tempo change occurred, it often was changed

from maximum to minimum or vice versa as shown in Figure 12.19. The weighted arithmetic mean bpm value is shown in Table 12.2 and lies within 101 and 144 bpm.

Participant	Time	Loop size	bpm
00	15:13	07	132
01	14:08	12	144
03	10:14	08	no data
04	32:22	07	113
*05	34:38	11	132
06	19:25	12	109
07	29:47	09	124
08	26:44	12	132
09	11:22	11	122
10	25:39	10	138
11	26:52	10	101

Table 12.2: Interaction time and weighted average loop size in beats and weighted average bpm value.

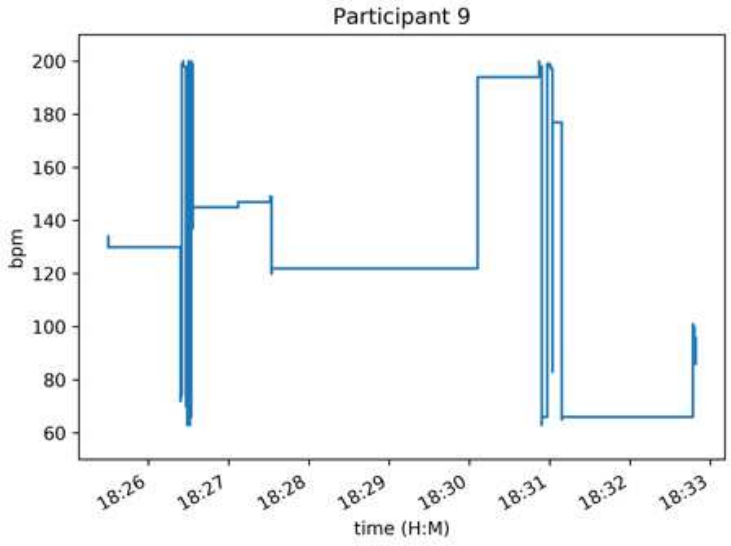


Figure 12.19: Bpm change over time.

12.2.8 Additional Observations

The participants composed melodies as well as chords by trial and error, although chords were not played for a the maximum available number of 4 beats. Three participants referred to their

composition as harmonic melodies and specifically put effort on creating melodies playing, at least partly, multiple strings. The implemented orientation system was not noticed by many participants. The UI expert suggested to visualize lines showing the horizontal and vertical alignment of the token while moving it.

Most participants interacting longer than 15 minutes started composing at the end of their interaction. They were more confident in using the tempo potentiometer, the loop bars, and started shaking their bodies rhythmically.

One participant suggested the use of special loop-tokens. Those tokens should be labeled with a repeat sign and additionally a numeric value such as x2, x3 or x4, with respect to the number an area between those tokens would be looped x times.

It was not obvious, even to any of the guitar playing, participants that frets overlap. Most of them were surprised that the highest tone in the blue area is higher than the lowest tone on the green area on top of the blue area. Some counted the orientation marks, drawing the correct conclusion that this must be the mapping of frets, and one even gesticulated the playing of a specific melody with an imaginary air guitar.

12.2.9 Repercussions

The higher positions in the red colored area were most distant from the user and generally played less often. That is most likely not only a distance issue, but most users started in the blue and green area, making chord creation within the red area way more complex. Participants were generally not noticing the implemented orientation hints, therefore, a better orientation guidance system should be implemented.

Probably biased by the joker in card games that can be really powerful or helpful, simply picking a tone from the pentatonic scale at the present beat was beneath expectations. Another observed problem was the interpretation of the behavior. When placing the joker token at a specific beat and then moving it somewhere else, it does not keep its tone height. That bears the potential to interpret the tone height as completely random. This random concept may again be supported by card game associations, where a joker card could randomly replaces any other card. Clearly the joker interaction has to be redesigned.

The loop size of maximum 16 beats seems to be sufficient as the arithmetic mean loop size is within seven to twelve beats. The suggestion to use additional loop tokens could partly overcome certain limitations, can be implemented easily, fits into the overall ways of interaction and should be considered within a potential redesign.

The interaction time, especially with the participants started composing after approximately 15 minutes, is insofar from interest, as interviewees in our expert interviews stated, that the interaction time in a museum context will not exceed a few minutes. In the testing environments users have a motivation to explore the interface and find all potential interaction mechanisms. In a museum a joker token causing confusion, for example, could frustrate the user leading to an interaction stop. There are many more distractions such as other installations or people waiting in a line to use the prototype. When choosing an explorative interface, it is self-evident that it takes time for the user to give a meaning to the interface. Nevertheless, this timespan should be reduced to support earlier composing.

One issue was, the participants not noticing the mechatronic part of the installation. There needs to be a form of highlighting. The movement of the servo motors can be seen and heard, but

obviously does not attract enough attention to the participants. Coloring the string module in the according interface's string color could be a solution to this problem.

The tokens should not be placed in carton boxes, but maybe in cups worked into the table so it provides a flat surface. The same goes for the loop-tokens and the wooden frame or the tokens should be designed in a way that users cannot place note tokens in between. That design change would also highlight the tempo potentiometer more, reducing the possibility to oversee it.

Part V

Summary and Future Work

CHAPTER 13

Summary

The design process was stopped after three design iterations for the mechanical part and two design iteration of the MUI part of this work. The authors managed to create a working proof of concept within the financial limitations fulfilling many requirements. The following sections provide a summary of the findings so far including errors which have been made during the design process.

13.1 Mechatronical Part

Author: Raphael Kamper

Regarding the development of a mechatronic instrument our previous knowledge and technical experience were strongly limited. We really underestimated the potential challenges coming with both hardware and software design for microcontrollers with an expertise in only software engineering. While most electronic components we used are not expensive, they are easily damageable and wiring mistakes can be extremely hard to find. In retrospective it is clear that it was not a good idea to do so, but testing the optical pickup, for instance, worked. Creating the plucking mechanism with stepper motors worked. We used an external power supply for the optical pickup to avoid power imbalances, but used unshielded wires, and therefore, electromagnetic fields from the stepper motor caused low but steady auditory noise. This was only revealed after we assembled the final prototype. Accidentally short-circuit components and being stuck in crucial design phases while waiting for ordered replacement parts cost a lot of time, especially in the early design phases when budgetary considerations prevented ordering of additional spare parts. Financial issues also influenced the quality of components, especially the servo motors, resulting in precision or speed problems.

On the other hand modeling the aluminum frame and 3D printing worked really well. 3D modeling prevented us from planning faulty constructions, fixes can be easily made and spare parts are available in common DIY stores. Printing hollow models to verify a design enables rapid prototyping, and while printing parts, others can be improved and redesigned already. 3D

printing[1] still has issues or limitations. Multiple attempts of really simple models, such as our note tokens, failed when printing tokens over a long time like nine hours or above. This can be settled by supervising prints but is inconvenient[2].

The iterative design process turned out to be quite efficient, although it was not always clear if an improvement in an early stage, such as the feedback position system, can be considered, not necessarily though, a new iteration already, but we decided otherwise. Also the decision to use a module based system has proven favorable. Each string module could be replaced and the overall system is extensible, allowing future upgrades. That mentioned, we now give an overview of our findings during each iterations.

13.1.1 First Iteration

The underlying physical principles of frequencies and frets for plucked string instruments, but also for stepper motors, were documented and calculation scripts were written. Those were useful throughout all subsequent iterations and used as a reference. The optical pickup was found to be working and not changed anymore from this iteration on. A solenoid for fretting gets too hot, deforming the PLA material and the replacement using a servo turned out to be satisfying solution regarding noise and speed. One of the most important lessons learned, was the necessity of a position feedback system. Principally, the output of this iteration was a functional module, but did not fulfill the requirements regarding speed. To achieve the desired precision we 3D printed frets allowing a position scope of a few millimeters.

13.1.2 Second Iteration

The findings regarding the position feedback system were also applied to the plucking mechanism were not changed in the last iteration. The second iterations approach did not fulfill our requirements and was aborted after it became clear. Retrospectively the approaches using rotary encoders are owed to a lack of research. Of course there are rotary encoders fulfilling our needs exist, but not within our budgetary restrictions, and the ones we used are not built for such high speeds. Potentiometers might be a possible replacement for this particular application. The decision to follow this approach lead to the development of 3D printed gears, a belt tensioner and calculations to create a mathematical model describing the interplay of gears, motors, feedback and positioning. None of the lastly mentioned elements were used for the next iteration or within the prototype. This iteration reflects our lack of experience with electromechanic components in particular. It would have been beneficial to have this in advance when working on such a project.

13.1.3 Third Iteration

The third and last iteration turned out to fulfill our requirements regarding speed and precision. We adapted the fretting mechanism in a way to use one servo motor per fret. This results in a very fast playable bpm value of about 350 bpm and absolute precision if the fret positions are calculated and placed correctly. High speed servo motors could reach an ever better performance. Software running on the Arduino Mega and Uno boards is supported by libraries supporting communication with the servo controller and the stepper motor drivers, the serial communication

[1] We used the Ultimaker 2 Extended+ with single extrusion and the Ultimaker 3 Extended with double extrusion.

[2] Especially as remote control in combination with webcams is pretty common by now for 3D printers, but not setup within the university environment.

with the laptop computer is easy to establish, simplifying the implementation. As mentioned in the above section introduction, stepper motors influence other components by electromagnetic fields, and reduction of such effects has to be considered. The third iteration was showcased at the yearly beginners' day at the TU Wien faculty of informatics. Both authors of this work could not attend the event, and a short email was written giving an explanation of how to put the installation into operation and troubleshoot potential problems. This worked quite well and allows an optimistic assumption that museum staff could also maintain the installation.

13.2 Musical User Interface

Author: Jakob Blattner

This section contains the summary of the definition of all elements and properties of the MUI throughout the design process. All major changes of the installation regarding the user requirements are visualized in Table 13.1, where each row visualizes a user requirement, feature or property of the system and each row symbolizes a conducted research method. Each cell contains a dot, which gives information about the state of the feature at the end of each respective method. There are four different dots which display different statuses. The white dot (○) defines, that the usage of the feature itself was clear, but not how it was going to be implement in all its detail. A green dot (●) either visualizes the first implementation or maintaining the implementation from the previous method. A yellow dot (●) visualizes its need to be redefined, and a red dot (●) defines its deletion. The table is intended to provide a quick overview of the design process of the MUI. Because of the number of methods applied, the table has to use abbreviations, as the table's would otherwise not fit on one page. *LR* stands for literature research, *EI* for expert Interviews, *UCs* for use cases, and *PT* stands for prototyping. A more detailed description has been written in the following subsections, with each design iteration captured in one subsection.

	LR	EI	UCs	Sketches	Mockups	1^{st} Eval.	PT	2^{nd} Eval.
Loop Behavior		○	●	●	●	●	●	●
Bars & Beats		○	○	●	●	●	●	●
Token Type		○	●	●	●	●	●	●
Tone Tokens			○	○	●	●	●	●
Creating Melody	○	○	○	●	●	●	●	●
Editing Melody	○	○	○	●	●	●	●	●
Adjusting Volume			○	●	●	●	●	
Adjusting Speed			○	●	●	●	●	●
Accompaniment	○	○	○	●	●			
Adjust Loop Area		○	●	●	●	●	●	●
Orientation						○	●	●
Snapping				●	●	●	●	●
Suggestions	○	○	○	○	●	●		
Joker							●	●

Table 13.1: In which design phase which interface property was set.

Before the first iteration, some properties of the installation have been predefined to define a specific direction in which the thesis should go. The target users were set to be non-musically trained people, not excluding musicians as potential users. Nevertheless, it was defined, that

the user interface is going to be designed for non-musically trained people. Additionally, the installation was defined to be non-collaborative, as a collaborative system automatically needs a minimum of two users to interact with. A circumstance that is not always given and thus can prevent an interaction in advance, something the authors wanted to avoid. The possibility of combining a collaborative and non-collaborative approach would have exceeded the scope of this work and probably would have produced an unsatisfactory result. The installation itself has to use a tangible input, as this is seen as an intuitive way of interacting and should therefore ease up the user's (maybe first) creation of music. The user should also be supported with suggestions and other, at that time undefined, supporting functionalities. The definition of a string instrument at the beginning of the thesis seems narrowly defined at first glance, but the fact that not only a guitar or bass but also a zither, harp, hurdy-gurdy etc. are defined as string instruments implies a variety of ways to play a string, a different sound and can even lead to very different ways of interaction. All these factors showed a broad variety of possible approaches.

13.2.1 First Iteration

The first method conducted in the first iteration was the literature research. In accordance with its purpose, literature research served to acquire knowledge in the relevant fields of this thesis. The literature research built the basis of this work, leading to the selection of the ISO 9241-210 process model as the underlying design process. Additionally, the playful design attitude was set to be a very important factor in the design of the interface, needing scalable difficulty to bring the user in a flow state. In the course of the literature research, many different MUIs with diverse properties were found. These also had a strong impact on the work through their presentation of possible starting points.

The expert interview defined many outlines of the project. Wearables were deemed to be unsuitable by one expert, as why the input via token became the center of attention. The loop approach, which has been already encountered in the literature research, was defined as the basic system behavior as half of the experts suggested this approach all by themselves. The guitar (or it's generated sound) was chosen to be the instrument of choice. A decision based on the guitar's wide notoriety and popularity. The accompaniment was set to be implemented with chords, being played on the three lower strings of the guitar, whereas the melody composed by the user was going to be played on the three other strings. Headphones were chosen as the device through which the sound should be output, so that the user is undisturbed by his/ her surroundings and vice versa. Apart from the sound generated, the experts recommended the omission of additional auditive feedback, with the exception of suggestions (dis-) appear on the interaction area. One expert suggested, that the interaction time for each user lies between three and fifteen minutes. Regarding the support for the users, one expert, of the concerning field, proposed to use the markov model, based on the previous played tones by the user, for suggestions.

In the course of the use case creation, the authors defined, that the user also must be able to change the playback speed and the playback volume of the installation. The user also has to be able to accept or decline suggestions by the system and can set the accompaniment manually. Many questions regarding the areas hardware, software, feedback and input were documented in the course of the creation of use cases. Questions which were then used to update the already defined requirements.

The sketching process was set out to define the token input and the basic system behavior more precisely. Three interface design approaches were created, one already dismissed in the creation process. One of the two remaining approaches had two possible token designs which differed in its

usage and visual representation. All three input possibilities where tested in the next evaluation. The following properties were the same across all interface design approaches. The x-axis of the interaction area was set to reflect the time, whereas the y-axis defined the pitch tone. Sixteen beats, based on a four quarter rhythm, and 23 pitch heights were defined to be available for the user's music creation process. With the use of the tone-token, the user was able to create single tones (no chords or similar) on said interaction area. Another type of of token was added, whose purpose was to adjust the loop area, in which the created melody gets repeated indefinitely. The chords for the accompaniment were appointed to be power chords, as their sound fits to every combination of tones. The user should also be able to manually set one power chord per bar (of which four are present). Another help, the so called snapping, was defined in the sketching process, simplifying the precise beat and pitch positioning on the interaction area.

Between the sketching and creation of mockups, the accompaniment was, in consultation with the thesis' supervisor, completely removed from the scope of this work, as the overall effort to be put into the thesis was already more than intended and planned. In the creation process of the mockups itself, it was defined to enable one tone per beat and that overlapping tokens would be active according to the first come first serve principle. This means, that the first token to be placed on a certain beat will be played as long as it lays there. If the token gets removed, the token which has been laid there after the now removed token will be played and so on. Four tone-token widths and therefore tone lengths were defined with the possibility of playing 1/4, 1/2, 3/4 and 4/4 tones. Key points of the suggestions were also defined. The visual feedback of a suggestion should be shown after four revolutions of the loop area, without any input of the user. This circumstance can be equated with the circumstance, that the user doesn't know what to do next and needs help. The position of the suggestions were defined to be the beat after the last laid tone-token. The playback speed and volume were also described more precisely. The dedicated input methods were defined as knobs as well as the icons shown above the turnable knobs.

After the evaluation of the mockups via users tests from standard users and experts alike, the researchers chose the blank approach and its rectangular input token because of the overall positive and fast understanding of users and feedback by the participating experts. The mapping of time and tone height on the respective x- and y-axis was also understood without any problems. The first come first serve principle had to be changed, as its behavior was not coherent for users nor experts. The last come last serve principle was mentioned to be the better functionality in this concern. Additionally, all participants needed some kind of orientation, which indicates on which tune and beat the token can be placed. Changes, which were set-in the second design iteration of the MUI.

13.2.2 Second Iteration

The prototyping was set to improve the encountered problems evaluated in the test of the previous iteration and create the first functional prototype of the installation. This meant, that before the improvements can be implemented, the hardware for image recognition and creation of visual feedback and the software components communication and visual feedback had to be chosen and a solution had to be programmed. In the course of the prototyping, suggestions were removed and joker were added. The joker is a new designed form of suggestion, with the difference that the user can change the time, length and beat position of the randomly chosen pentatonic tone which fits to every other input. After this had be done, further definitions were done by the authors. Additionally the final token size of the tone and loop-token were set. The previously defined input device for the playback speed was also created. The input device for the playback

volume was left out, as for the evaluation in the second design iteration, the researchers had to listen to the melodies created by the participants. The input device for the playback sound itself should not pose any problems since the input method (a rotary encoder) and the show icons (two speakers with on and three visualized sound waves respectively) are widely known and used. The functionality to lay one tone per beat was changed to one tone per beat and string to see if the users were overwhelmed by this possibility or would welcome this approach. A visual representation of this was added by giving every string a certain color. This color was displayed on the mechanical part (but only rudimentary), the projected bars on the GUI representing the current start and end of the loop and underneath the token itself. The mentioned orientation support was added at the sides of the interaction area.

The last method in the second iteration was, again, an evaluation. This time, logging also had been added as a source of quantitative data. The results were as follows: the behavior of the joker wasn't understood by every except one participant. The loop-token were partly not recognized by the users, which could have had something to do with the container in which the token were put in, as it slightly blocked the view on and the accessibility of the loop-token. The same applies for the potentiometer to change the playback speed. Not everyone understood the last come last serve functionality immediately, as well as the association between colors and strings. Interestingly some users interacted way longer than fifteen minutes with the system. It must highlighted, that the place where the test was conducted differed widely form the future museal use context. The implemented orientation support, at the border of the interaction area, was also too far from the center of attention to be perceived by the user. There were also some suggestions by the participants, including the possibility to save and load certain parts of a melody, the implementation of a play/pause button, and additional tokens which would repeat a certain area in the loop area for two, three or four times in a row.

CHAPTER 14

Future Work

Author: Raphael Kamper

The most important task for future work is the polishing of the prototype, fixing open issues, and then conduct a user study in a museum context. The findings of the user test we held were quite revealing, but in a museum it is most likely that the participants' interaction behavior would change. Despite this suggestion we have a lot of concrete comments for redesign, mostly relying on the user test evaluation from the second iteration, but also like to address potential issues. Therefore we start with questioning our very basic core concept: the looping behavior.

> "Music exists in the realm of time." [77, p. 29]

The loop approach picks up this fact by providing a (theoretical) infinite amount of time, and therefore, music can be created with the MUI. The disadvantage coming with this approach is that unlimited time meets limited space. This conflict can be overcome by a skilled person able to place tokens very fast, but the current interface does not support such skilled persons and besides it is very unlikely that a museum visitor will gain those skills. While the composing interaction tends to be working, it might be useful to question the loop approach or at least implement advanced looping capacities (see Section 3.4.2).

14.1 Advanced Looping Capacities

One possibility to achieve this could be a play and pause function. Especially in early stages it annoys users to hear the same sequence of tones being played again and again. It never gives the user time to think and constantly pressures the user to interact. Providing a pause mode, where users get feedback only for moved tokens, for instance, would allow an easier composition of chords or harmonic melodies.

Another adaption could be the support of saving particular sequences and replay them later on. Therefore, an overview and detail design approach could be applied by displaying the visualizations of the saved sequences and only using present inputs as potential new sequence being attached to the former composition.

Taking up the suggestion participant of the user test made, special loop-tokens could also easily extend the limited space and in this manner mocking the saving of a sequence. This would most likely lead to a special, probably minimalistic, form of compositions. Looping multiple sequences quickly extends the limited number of 16 beats many times.

14.2 Joker Interaction

The joker interaction definitely needs a complete redesign. One scenario could be that the joker plays a melody, algorithmically composed or hard coded. This would address multiple issues. First of all, the attributed special behavior to the joker could meet the users' expectations. Secondly, this could be designed in a way once the joker is placed, visual representations of the melody are show and played, and the user can rebuild the melody by placing ordinary note tokens on the visual representations of the joker's melody. After removing the joker token, the user has a good sounding melody represented by the note tokens. Placing the joker again would extend the melody. The joker size could indicate the number of tones included in the melody. Thirdly, this could initiate composing by the user herself/himself, because those more sophisticated suggestions give the user a better understanding of what is possible. On the other hand, this could be considered patronizing by the user, resulting in a lack of motivation to compose by himself/herself.

14.3 Orientation

It may be helpful to introduce tempo stages as most users tried out extreme values, but set the potentiometer somewhere near a middle position and leaving there. Stages of different tempi in 20 bpm steps, for instance, could animate the user to find a more suitable tempo for his/her composition. Orientation also needs better support when adjusting the note position. Even if users noticed the orientation marks described in Subsection 3.2, they are only visible at the edges, while the position fine tuning might take place in the middle of the playing field. Therefore, orientation lines connected with the already existing feedback, only shown during token movement, could improve this situation. The loop bars actually take a lot of the limited space. We considered moving the loop bars by finger, which would be supported by the reacTIVision framework. It would save space, but if this is the only interaction not including tokens might not be obvious to the user, because there is no affordance as described in Subsection 3.2.2. As already considered during the first iteration's sketching phase (see Subsection 11.6.4), highlighting the mechatronic part with colored LEDs could make the color mapping obvious to the user. This might result in a faster understanding of string and fret mapping within the interface.

An interviewee in the expert interviews (see Subsection 11.2) advised us against an automatic play mode for the instrument. This bears a too high potential of the user not understanding the interaction, but a user interface expert during the mockup test (see Subsection 11.7) suggested the auto-play of a tutorial. In a museum context this might be very useful. Starting a tutorial, without actually playing anything, but only showing a video of possible interactions, gives the user a hint that interaction is desired, and at the same time teaches basic ways to do so.

14.4 Additional Suggestions

The loop-token frame should be redesigned in a way that other tokens do not fit in between. This only causes confusion and such an interaction should not be possible at all. As mentioned in

Section 12.1, the video projector overheated during long term use. Sufficient ventilation has to be provided when using such a device in a museum.

Special designed chord token could provide the possibility of playing more sophisticated melodies and allow simple transpositions. Consistent with chords, chord accompaniment creation could lead to a completely different composing behavior. We considered an automated chord creation using the first iterations (see Chapter 8) approach of the mechatronic part. An easy we to achieve a good sounding accompaniment would be by the use of power chords as described in Section 4.1.7, using a present note token as a root tone.

List of Figures

List of Tables

Bibliography

[1] M. Bretan and G. Weinberg, "A survey of robotic musicianship," *Communications of the ACM*, vol. 59, pp. 100–109, 2016.

[2] D. Bihanic, *Empowering Users through Design: Interdisciplinary Studies and Combined Approaches for Technological Products and Services*. Springer, 2015.

[3] J. M. Carroll, *Human-computer interaction in the new millennium*. ACM Press, 2001.

[4] S. Johnson, *Interface Culture: How New Technology Transforms the Way We Create and Communicate*. Harper, 2000.

[5] W. O. Galitz, *The Essential Guide to User interface Design: An Introduction to GUI Design Principles and Techniques*. Wiley, 2007.

[6] I. S. MacKenzie, "Historical context," in *Human-Computer Interaction*, pp. 1–26, Morgan Kaufmann, 2013.

[7] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., 1983.

[8] T. Catarci, "Visual interfaces," in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), pp. 3379–3382, Springer US, 2009.

[9] A. F. Blackwell and M. F. Costabile, "Direct manipulation," in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), pp. 847–847, Springer US, 2009.

[10] M. F. Costabile and A. F. Blackwell, "Visual metaphor," in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), pp. 3387–3388, Springer US, 2009.

[11] Y. Ioannidis, "Visual representation," in *Encyclopedia of Database Systems* (L. Liu and M. T. Özsu, eds.), pp. 3405–3410, Springer US, 2009.

[12] I. E. Sutherland, "Sketchpad: A man-machine graphical communication system," in *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference*, pp. 329–346, ACM, 1963.

[13] H. D. Hellige, *Geschichten der Informatik: Visionen, Paradigmen, Leitmotive*. Springer, 2004.

[14] J. Johnson, T. Roberts, W. Verplank, D. Smith, C. Irby, M. Beard, and K. Mackey, "The xerox star: a retrospective," *Computer*, vol. 22, no. 9, pp. 11–26, 1989.

[15] J. Johnson, *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Elsevier Inc., 2014.

[16] H. Ishii, "The tangible user interface and its evolution," *Communications of the ACM*, vol. 51, no. 6, pp. 32–36, 2008.

[17] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre, "Haptic interfaces and devices," *Sensor Review*, vol. 24, no. 1, pp. 16–29, 2004.

[18] O. Shaer and E. Hornecker, "Tangible user interfaces: past, present, and future directions," *Foundations and Trends in Human-Computer Interaction*, vol. 3, no. 1–2, pp. 1–137, 2010.

[19] S. Hinske, M. Langheinrich, and M. Lampe, "Towards guidelines for designing augmented toy environments," in *Proceedings of the 7th ACM Conference on Designing Interactive Systems*, DIS '08, pp. 78–87, ACM, 2008.

[20] O. Zuckerman, S. Arida, and M. Resnick, "Extending tangible interfaces for education," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '05*, p. 859, ACM, 2005.

[21] A. N. Antle, "The cti framework: Informing the design of tangible systems for children," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pp. 195–202, ACM, 2007.

[22] B. Ullmer, H. Ishii, and R. J. K. Jacob, "Token+constraint systems for tangible interaction with digital information," *ACM Transactions on Computer-Human Interaction*, vol. 12, no. 1, pp. 81–118, 2005.

[23] L. Garber, "Tangible user interfaces: Technology you can touch," *Computer*, vol. 45, no. 6, pp. 15–18, 2012.

[24] O. Zuckerman, "To tui or not to tui: Evaluating performance and preference in tangible vs. graphical user interfaces," *International Journal of Human-Computer Studies*, vol. 71, no. 7-8, pp. 803–820, 2013.

[25] B. Schneider, P. Jermann, G. Zufferey, and P. Dillenbourg, "Benefits of a tangible interface for collaborative learning and interaction," *IEEE Transactions on Learning Technologies*, vol. 4, no. 3, pp. 222–232, 2011.

[26] M. Sturmlechner, *Prototypische Entwicklung eines Tangible Musical Interfaces für Kinder im Vorschulalter*. PhD thesis, TU Wien, 2008.

[27] X. Xiao, P. Puentes, E. Ackermann, and H. Ishii, "Andantino: Teaching children piano with projected animated characters," in *Proceedings of the The 15th International Conference on Interaction Design and Children*, IDC '16, pp. 37–45, ACM, 2016.

[28] J. T. Sello, "The hexenkessel: A hybrid musical instrument for multimedia performances," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '16, pp. 122–131, 2016.

[29] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, "The reactable: exploring the synergy between live music performance and tabletop tangible interfaces," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pp. 139–146, ACM, 2007.

[30] B. Smus and M. D. Gross, "Ubiquitous drums: A tangible, wearable musical interface," in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pp. 4009–4014, ACM, 2010.

[31] H. Newton-Dunn, H. Nakano, and J. Gibson, "Block jam: a tangible interface for interactive music," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '03, pp. 170–177, 2003.

[32] T. Fischer and W. Lau, "Marble track music sequencers for children," in *Proceedings of the 2006 Conference on Interaction Design and Children*, IDC '06, pp. 141–144, ACM, 2006.

[33] J. Patten, B. Recht, and H. Ishii, "Audiopad: a tag-based interface for musical performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '02, pp. 148–153, 2002.

[34] J. Van Der Linden, E. Schoonderwaldt, J. Bird, and R. Johnson, "Musicjacket—combining motion capture and vibrotactile feedback to teach violin bowing," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 104–113, 2011.

[35] D. Becking, C. Steinmeier, and P. Kroos, "Drum-dance-music-machine: Construction of a technical toolset for low-threshold access to collaborative musical performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '16, pp. 112–117, 2016.

[36] D. Ogawa, K. Tanabe, V. Yem, T. Hachisu, and H. Kajimto, "Haptone: Haptic instrument for enriched musical play," in *ACM SIGGRAPH 2016 Emerging Technologies*, SIGGRAPH '16, pp. 12:1–12:2, ACM, 2016.

[37] H. Kanke, T. Terada, and M. Tsukamoto, "A percussion learning system by rhythm internalization using haptic indication," in *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology*, ACE '15, pp. 14:1–14:5, ACM, 2015.

[38] F. Zamorano, "Simpletones: A collaborative sound controller system for non-musicians," in *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pp. 3155–3158, ACM, 2013.

[39] A. Knörig, B. Müller, and R. Wettach, "Articulated paint: Musical expression for non-musicians," in *Proceedings of the international conference on new interfaces for musical expression*, NIME '07, pp. 384–385, ACM, 2007.

[40] E. Costanza, S. B. Shelley, and J. Robinson, "Introducing audio d-touch: A tangible user interface for music composition and performance," in *Proceedings of the 2003 International Conference on Digital Audio Effects*, DAF '03, 2003.

[41] O. Vallis and A. Kapur, "Community-based design: The democratization of musical interface construction," *Leonardo Music Journal*, vol. 21, pp. 29–34, 2011.

[42] P. Bennett and S. O'Modhrain, "The beatbearing: a tangible rhythm sequencer," in *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges*, NordiCHI '08, ACM, 2008.

[43] M. B. Alonso and D. V. Keyson, "Musiccube: making digital music tangible," in *CHI'05 extended abstracts on Human factors in computing systems*, CHI EA '05, pp. 1176–1179, ACM, 2005.

[44] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pp. 206–213, ACM, 1993.

[45] C. M. Macdonald, *Understanding Usefulness in Human-computer Interaction to Enhance User Experience Evaluation.* PhD thesis, Drexel University, 2012.

[46] E. Rosenzweig, *Successful User Experience: Strategies and Roadmaps.* Elsevier Inc., 2015.

[47] S. Gelineck and D. Overholt, "Haptic and visual feedback in 3d audio mixing interfaces," in *Proceedings of the Audio Mostly 2015 on Interaction With Sound*, AM '15, pp. 14:1–14:6, ACM, 2015.

[48] J. Larsen and H. Knoche, "Hear you later alligator: How delayed auditory feedback affects non-musically trained people's strumming," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '17, pp. 456–459, 2017.

[49] S. Jordà, "Interactive music systems for everyone: exploring visual feedback as a way for creating more intuitive, efficient and learnable instruments," in *Proceedings of the Stockholm Music Acoustics Conference*, SMAC '03, (Stockholm, Sweden), p. 44, 2003.

[50] J. J. Gibson, "Observations on active touch," *Psychological Review*, vol. 69, no. 6, pp. 477–491, 1962.

[51] E. Groen, M. Oosterbeek, A. Toet, and I. Hooge, "Discrimination of concurrent vibrotactile stimuli," in *Haptics: Perception, Devices and Scenarios* (M. Ferre, ed.), pp. 23–32, Springer Berlin Heidelberg, 2008.

[52] C. D. Gilbert, "The constructive nature of visual processing," in *Principles of Neural Science* (E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. J. Hudspeth, eds.), ch. 25, pp. 556–576, McGraw-Hill Education, fifth edit ed., 2012.

[53] D. A. Norman, "Affordance, conventions, and design," *Interactions*, vol. 6, no. 3, pp. 38–43, 1999.

[54] J. Barbosa, M. M. Wanderley, and S. Huot, "Exploring playfulness in nime design: The case of live looping tools," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '17, pp. 87–92, 2017.

[55] D. Arfib, J.-M. Couturier, and L. Kessous, "Expressiveness and digital musical instrument design," *Journal of New Music Research*, vol. 34, pp. 125–136, 2005.

[56] N. B. Ruparelia and N. B., "Software development lifecycle models," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, p. 8, 2010.

[57] M. Richter and M. Flückiger, *User-Centred Engineering.* Springer, 2014.

[58] R. Kneuper, "Sixty years of software development life cycle models," *IEEE Annals of the History of Computing*, vol. 39, no. 3, pp. 41–54, 2017.

[59] C. Larman and V. R. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.

[60] D. Stone, C. Jarrett, M. Woodroffe, and S. Minocha, *User Interface Design and Evaluation.* Interactive Technologies, Elsevier Science, 2005.

[61] D. Hix and H. R. Hartson, *Developing User Interfaces: Ensuring Usability through Product & Process*. John Wiley & Sons, Inc., 1993.

[62] M. F. Costabile, "Usability in the software life cycle," *Handbook of Software Engineering and Knowledge Engineering*, vol. 1, pp. 179–192, 2001.

[63] D. J. Mayhew and D. J., "The usability engineering lifecycle," in *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '99, p. 147, ACM, 1999.

[64] A. Cooper, R. Reimann, D. Cronin, and A. Cooper, *About face 3: The Essentials of Interaction Design*. Wiley Publishing, Inc., 2007.

[65] B. Shneiderman and C. Plaisant, *Designing the User Interface: Strategies for Effective Human-computer Interaction*. Addison-Wesley, 2010.

[66] A. Nijholt, *Playful user interfaces: Interfaces that invite social and physical interaction*. Gaming Media and Social Effects, Springer Singapore, 2014.

[67] D. Robson, "Play!: Sound toys for non-musicians," *Computer Music Journal*, vol. 26, no. 3, pp. 50–61, 2002.

[68] W. Thompson, R. Fleming, S. Creem-Regehr, and J. K. Stefanucci, *Visual Perception from a Computer Graphics Perspective*. CRC Press, 2016.

[69] K. Baxter, C. Courage, K. Caine, K. Baxter, C. Courage, and K. Caine, *Understanding Your Users: A Practical Guide to User Research Methods*. Elsevier Inc., 2015.

[70] C. Moser, *User Experience Design: mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern*. Springer, 2012.

[71] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pp. 152–158, ACM, 1994.

[72] C. M. Barnum, *Usability testing and research*. Allyn & Bacon, Inc., 2001.

[73] K. E. Wiegers, *Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycl*. Microsoft Press, 2003.

[74] G. Weinberg and S. Driscoll, "Toward robotic musicianship," *Computer Music Journal*, vol. 30, no. 4, pp. 28–45, 2006.

[75] R. Rowe, *Machine Musicianship*. The MIT Press, 2001.

[76] K. Haider, *Einführung in die Musiktheorie*. Lang, 2000.

[77] H. Owen, *Music Theory Resource Book*. Oxford University Press, 2000.

[78] C. von Ehrenfels, *Über "Gestaltqualitäten"*. Reisland, 1890.

[79] E. Mach, *Beiträge zur Analyse der Empfindungen*. G. Fischer, 1886.

[80] V. S. Ramachandran and E. M. Hubbard, "Synaesthesia–a window into perception, thought and language," *Journal of Consciousness Studies*, vol. 8, no. 12, pp. 3–34, 2001.

[81] D. Maurer, T. Pathman, and C. J. Mondloch, "The shape of boubas: Sound–shape correspondences in toddlers and adults," *Developmental Science*, vol. 9, no. 3, pp. 316–322, 2006.

[82] W. Köhler, *Psychologische Probleme*. Springer-Verlag Berlin Heidelberg, 1933.

[83] G. Nierhaus, *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer, 2009.

[84] O. Hödl, *The design of technology-mediated audience participation in live music*. PhD thesis, TU Wien, 2016.

[85] H. Young, "A categorial grammar for music and its use in automatic melody generation," in *Proceedings of the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design*, FARM '17, pp. 1–9, ACM, 2017.

[86] S. D. You and P.-S. Liu, "Automatic chord generation system using basic music theory and genetic algorithm," in *2016 IEEE International Conference on Consumer Electronics-Taiwan*, pp. 1–2, 2016.

[87] A. Freed and A. Schmeder, "Features and future of open found control version 1.1 for nime," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '09, pp. 116–120, 2009.

[88] S. Barton, E. Prihar, and P. Carvalho, "Cyther: a human-playable, self-tuning robotic zither," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '17, pp. 319–324, 2017.

[89] E. Singer, K. Larke, and D. Bianciardi, "Lemur guitarbot: Midi robotic string instrument," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '03, pp. 188–191, 2003.

[90] J. McVay, D. A. Carnegie, J. W. Murphy, and A. Kapur, "Mechbass: A systems overview of a new four-stringed robotic bass guitar," in *Proceedings of the 19th Electronics New Zealand Conference*, 2012.

[91] M. J. Kartomi, *On Concepts and Classifications of Musical Instruments*. Chicago Studies in Ethnomusicology, University of Chicago Press, 1990.

[92] S. A. Bedini, "The role of automata in the history of technology," *Technology and Culture*, vol. 5, no. 1, pp. 24–42, 1964.

[93] W. Schmidt, *Herons von Alexandria Druckwerke und Automatentheater*. B. G. Teubner, 1899.

[94] A. W. J. G. Ord-Hume, "Cogs and crotchets: A view of mechanical music," *Early Music*, vol. 11, no. 2, pp. 167–171, 1983.

[95] A. Kapur, "A history of robotic musical instruments," in *Proceedings of the 2005 International Computer Music Conference*, 2005.

[96] A. Buchner, *Vom Glockenspiel zum Pianola*. Artia, 1959.

[97] B. Schmuhl, *Maschinen und Mechanismen in der Musik: 31. wissenschaftliche Arbeitstagung, Michaelstein, 9. bis 11. Mai 2003*. Michaelsteiner Konferenzberichte, Wißner, 2006.

[98] J. Murphy, J. McVay, P. Mathews, D. A. Carnegie, and A. Kapur, "Expressive robotic guitars: Developments in musical robotics for chordophones," *Computer Music Journal*, vol. 39, pp. 59–73, 2015.

[99] C. V. Raman, "Experiments with mechanically-played violins," in *Proceedings of the Indian Association for the Cultivation of Science*, vol. 6, 1920.

[100] A. Satz, "Music of its own accord," *Leonardo Music Journal*, vol. 20, pp. 73–78, 2010.

[101] J. W. Murphy, P. Mathews, A. Kapur, and D. A. Carnegie, "Robot: Tune yourself! automatic tuning for musical robotics," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '14, pp. 565–568, 2014.

[102] Y. Pan, M.-G. Kim, and K. Suzuki, "A robot musician interacting with a human partner through initiative exchange," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '10, pp. 166–169, 2010.

[103] G. Hoffman and G. Weinberg, "Interactive improvisation with a robotic marimba player," *Autonomous Robots*, vol. 31, no. 2-3, pp. 133–153, 2011.

[104] G. Hoffman and G. Weinberg, "Gesture-based human-robot jazz improvisation," in *2010 IEEE International Conference on Robotics and Automation*, pp. 582–587, IEEE, 2010.

[105] M. Kajitani, "Development of musician robots in japan," in *Proceedings of the Australian Conference on Robotics and Automation*, 1999.

[106] L. Flanigan and A. Doro, "Plink jet," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '08, (Genoa, Italy), pp. 349–351, 2008.

[107] R. Vindriis and D. A. Carnegie, "Strumbot—an overview of a strumming guitar robot," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '16, 2016.

[108] T. Ogata and G. Weinberg, "Robotically augmented electric guitar for shared control," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '17, pp. 487–488, 2017.

[109] S. Sugano and I. Kato, "Wabot-2: Autonomous robot with dexterous finger-arm–finger-arm coordination control in keyboard performance," in *1987 IEEE International Conference on Robotics and Automation Proceedings*, vol. 4, pp. 90–97, IEEE, 1987.

[110] A. Zhang, M. Malhotra, and Y. Matsuoka, "Musical piano performance by the act hand," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3536–3541, IEEE, 2011.

[111] C. Crick, M. Munz, T. Nad, and B. Scassellati, "Synchronization in social tasks: Robotic drumming," in *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 97–102, IEEE, 2006.

[112] J. Long, A. Kapur, and D. A. Carnegie, "The closed-loop robotic glockenspiel: Improving musical robots with embedded musical information retrieval," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, NIME '16, pp. 2–7, 2016.

[113] J. H. van der Meer, *Musikinstrumente: von der Antike bis zur Gegenwart*. Prestel, 1983.

[114] R. Dannenberg, H. Ben Brown, and R. Lupish, "Mcblare: A robotic bagpipe player," in *Musical Robots and Interactive Multimodal Systems* (J. Solis and K. Ng, eds.), pp. 165–178, Springer Berlin Heidelberg, 2011.

[115] J. Solis, K. Taniguchi, T. Ninomiya, T. Yamamoto, and A. Takanishi, "Development of waseda flutist robot wf-4riv: Implementation of auditory feedback system," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3654–3659, IEEE, 2008.

[116] J. Solis, K. Ozawa, K. Petersen, and A. Takanishi, "Design and development of a new biologically-inspired mouth mechanism and musical performance evaluation of the wf-4rvi," in *2013 IEEE Workshop on Advanced Robotics and its Social Impacts*, pp. 200–205, IEEE, 2013.

[117] M. Maguire, "Methods to support human-centred design," *International Journal of Human-Computer Studies*, vol. 55, no. 4, pp. 587–634, 2001.

[118] H. Topi and A. Tucker, *Computing Handbook: Information Systems and Information Technology*. CRC Press, 2014.

[119] K. Pohl and C. Rupp, *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. Dpunkt Verlag, 2015.

[120] K. Bittner and I. Spence, *Use Case Modeling*. Addison-Wesley, 2003.

[121] J. A. Jacko, *Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*. CRC Press, 2012.

[122] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza, "Tuio - a protocol for table based tangible user interfaces," in *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, 2005.

[123] R. Bencina, M. Kaltenbrunner, and U. P. Fabra, "The design and evolution of fiducials for the reactivision system," in *Proceedings of the 3rd International Conference on Generative Systems in the Electronic Arts*, p. 10, 2005.

APPENDIX A

Consent Forms

A.1 Expert Interviews

Master Thesis Advisor:
Associate Prof. Dipl.-Ing. Dr.techn. Hilda Tellioğlu
hilda.tellioglu@tuwien.ac.at

Master Thesis Assistant:
Univ.Ass. Dipl.-Ing. Peter Fikar, Bakk.techn.
peter.fikar@tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

Researchers:
Jakob Blattner
e1026117@student.tuwien.ac.at

Raphael Kamper
e1125579@student.tuwien.ac.at

Consent Form

Goal of Research

Goal of this research is to gain insight and a deeper understanding of music composition with a special focus on algorithmic composition, application of music theory for string instruments and user research as well as audience participation within a musical context. The research is done as a part of the researchers' master thesis *Design of a haptic music Interface to support Performance Art on String Instruments* (tentative title). The thesis is advised by Hilda Tellioğlu and Peter Fikar.

Researchers

The researchers: Jakob Blattner and Raphael Kamper are all students of the TU Vienna and currently working on their master thesis as mentioned above.

Participation and Withdrawal

We would like to do an interview with you that will take around 30 minutes. Your participation is entirely voluntary. You can decline to answer any question. You may withdraw at any point or afterwards.

Use of research / Confidentiality

The collected data during the interview will be used for the researchers' master thesis in anonymized form. Information will not be shared with anyone else without your permission.

Permissions

You grant us the right to takes notes during the interview and to audio record this interview.

Feel free to ask questions and tell us your concerns at any time.

Participant

_____	_____	_____
Name	Place, DD.MM.YYYY	Signature

Researchers

_____	_____	_____
Name	Place, DD.MM.YYYY	Signature
_____	_____	_____
Name	Place, DD.MM.YYYY	Signature

A.2 Mockup Tests

Master Thesis Advisor:
Associate Prof. Dipl.-Ing. Dr.techn. Hilda Tellioğlu
hilda.tellioglu@tuwien.ac.at

Master Thesis Assistant:
Univ.Ass. Dipl.-Ing. Peter Fikar, Bakk.techn.
peter.fikar@tuwien.ac.at

Researchers:
Jakob Blattner
e1026117@student.tuwien.ac.at

Raphael Kamper
e1125579@student.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

Consent Form

Goal of Research

Goal of this research is to gain insight and a better understanding of the interactions taking place while using a haptic music interface. The research is done as a part of the researchers' master thesis *Design of a haptic music Interface to support Performance Art on String Instruments* (tentative title). The thesis is advised by Hilda Tellioğlu and Peter Fikar.

Researchers

The researchers: Jakob Blattner and Raphael Kamper are both students at the TU Wien and currently working on their master thesis as mentioned above.

Participation and Withdrawal

We would like to do a mockup test with you that will take around 15 minutes. Your participation is entirely voluntary. You can decline to answer any question. You may withdraw at any point or afterwards.

Use of research / Confidentiality

The collected data during the interview will be used for the researchers' master thesis in anonymized form. Information will not be shared with anyone else without your permission.

Permissions

You grant us the right to takes notes during the test and to video record you.

Feel free to ask questions and tell us your concerns at any time.

Participant

Name	Place, DD.MM.YYYY	Signature

Researchers

Name	Place, DD.MM.YYYY	Signature
Name	Place, DD.MM.YYYY	Signature

A.3 User Tests

Master Thesis Advisor:
Associate Prof. Dipl.-Ing. Dr.techn. Hilda Tellioğlu
hilda.tellioglu@tuwien.ac.at

Master Thesis Assistant:
Univ. Ass. Dipl.-Ing. Peter Fikar, Bakk.techn.
peter.fikar@tuwien.ac.at

Researchers:
Jakob Blattner
e1026117@student.tuwien.ac.at

Raphael Kamper
e1125579@student.tuwien.ac.at

Consent Form

Goal of Research

Goal of this research is to gain insight and a better understanding of the interactions taking place while using a haptic music interface. The research is done as a part of the researchers' master thesis *Design of a haptic music Interface to support Performance Art on String Instruments* (tentative title). The thesis is advised by Hilda Tellioğlu and Peter Fikar.

Researchers

The researchers: Jakob Blattner and Raphael Kamper are both students at the TU Wien and currently working on their master thesis as mentioned above.

Participation and Withdrawal

We would like to do a user test with you that will take around 15 minutes. Your participation is entirely voluntary. You can decline to answer any question. You may withdraw at any point or afterwards.

Use of research / Confidentiality

The collected data during the user test will be used for the researchers' master thesis in anonymized form. Information will not be shared with anyone else without your permission.

Permissions

You grant us the right to takes notes during the test and to video record you.

Feel free to ask questions and tell us your concerns at any time.

Participant

_____	Wien, 25.09.2018	_____
Name	Place, DD.MM.YYYY	Signature

Researchers

Jakob Blattner	Wien, 25.09.2018	_____
Name	Place, DD.MM.YYYY	Signature
Raphael Kamper	Wien, 25.09.2018	_____
Name	Place, DD.MM.YYYY	Signature

APPENDIX B

Interview Guideline

23.01.2018

Komposition

Interviewpartner: [redacted]
Erwartungen: Ansätze zur Musikerzeugung. Möglichkeiten live zu komponieren bzw. zu improvisieren. Im Idealfall kann ein allgemeingültiger Ansatz (innerhalb eines gewissen Kontexts, Genres, usw.) implementiert werden. Vor allem Hinweise zur Musiktheorie und Algorithmen.

Speziell an [redacted]: Vorgehensweise zur Festlegung der Interaktionsmöglichkeiten bei Trombosonic.

Fragensammlung- Komposition:

Musiktheorie:

- Welche Schemata gibt es mit dem die Eingabe des Benutzers vereinfacht wird, sodass das Erzeugnis harmonisch ist?
  - Begrenzung der "erlaubten" Töne wie z.B. Pentatonik?
  - Akkordabfolgemuster bei Begleitstimmen?
  - Welche dieser Schemata eignen sich für das Gitarrenspiel?

Algorithmen:

- Welche Möglichkeiten gibt es zur automatischen Musikerzeugung bzw. Komposition?
  - Algorithmisch und Improvisationstechniken auf Gitarre?
- Welche Faktoren beeinflussen die Wahl des automatischen Musikerzeugunsalgorithmus?
- Welche Möglichkeiten gibt es jeweils für Begleitstimmen oder Melodie?
  - Worin liegen die Unterschiede?
  - Kann die Begleitung zur Laufzeit algorithmisch erzeugt werden?
  - Kann sich die Begleitung positiv auf das Empfinden des Benutzers auswirken (klingt besser → größeres Erfolgsgefühl)?
- Welcher Algorithmus ist somit zu empfehlen?
- Welche Anwendungsgebiete (also z. B. Genre, Fähigkeiten, live- Improvisation vs. Komponieren) haben diese Ansätze (in unserem Rahmen, also zeitliche kurze "Echtzeit"-anwendung)?
- Welche Vor- und Nachteile ergeben sich durch die jeweiligen Ansätze?

Instrument:

- Welche Auswirkungen sind durch die Wahl des Instruments zu erwarten?
  - Aufforderung zur Interaktion von gewissen Instrumenten größer als von anderen (s.u.)?
- Sieht der Interviewpartner die Verwendung von analog gespielten Saiten als Pro oder Contra? Was spricht dafür/ dagegen?
  - Gibt es erwartbare oder feststellbare Unterschiede in der Interaktion mit einem analogen Musikinstrument im Vergleich zu rein digitaler Musikerzeugung?

1

- Wie kann ein geeignetes Mapping zwischen Eingabe und dem zu verändernden Tonparameter (Lautstärke, Tonhöhe, Dauer) gefunden werden?
    - Gibt es hier bekannte Synästhesien, also z.B. tiefer Ton korreliert mit großem Objekt oder beispielsweise mit Farben?

Benutzer:

- Mit welchen Problemen haben die meisten Nichtmusiker zu kämpfen wenn sie eine Lehrveranstaltung beim Interviewpartner ([redacted]) absolvieren bzw. daran teilnehmen?
- Wird der Einsatz im musealen Kontext als sinnvoll erachtet?
    - Gibt es weitere/ ander mögliche Kontexte in denen die Installation verwendet werden kann?
    - Können sich bestimmte Elemente als für die Interaktion förderlich erweisen?
- Unter welchen Voraussetzungen kann die initiale Interaktion gefördert werden?
    - Wie verhält sich die Interaktion mit dem Publikum?
    - Audioausgabe von Bedeutung (Kopfhörer vs. “offene” Lautsprecher)?

Interface

Interviewpartner: [redacted]
Erwartungen: Neue Einsichten von Experten mit Erfahrung auf dem jeweiligen Gebiet. [redacted] hat keine (uns bekannte) direkte Erfahrung mit MUIs, jedoch mit TUIs und UIs im Allgemeinen. Demnach können Fehler, Probleme mit unterschiedlichen Eingabemethoden, bei Usability Evaluation etc. in Erfahrung gebracht werden. Auch kann [redacted] zum Design Prozess Tipps geben, neue Designansätze liefern etc.

[redacted] wiederum ist auf dem Gebiet der MUIs sehr erfahren. Bei ihm liegen die Erwartungen an das Interview im speziellen auf MUIs und nicht UIs im generellen.

Da beide Interviewpartner einen anderen Ausgangspunkts bzgl. MUIs/ TUIs haben, bleiben die Fragen bei beiden im Großen und Ganzen gleich. Somit sollen unterschiedliche Lösungsansätze und Vorschläge in Erfahrung gebracht und im Designprozess berücksichtigt werden.

Fragensammlung- Interface:

Usability:

- Auf was muss bei der Usability Evaluierung mit Personen ohne Kenntnis im Bereich in dem das Interface angewendet wird beachtet werden?
- Welche Usability Evaluationsmethoden werden auf der TU meistens eingesetzt? Mit welchen Usability Evaluationsmethoden gibt es meistens die größten Probleme?

Playful Interfaces:

- Ist der “Playful Interface” Ansatz sinnvoll für unser Projekt?
- Soll bei einem Playful Interface Ansatz auch ein gewisser Schwierigkeitsgrad (--> no fun without a challenge) beachtet werden?

Input Methoden:

- Welche Input Methoden waren bei TUIs, die auf der TU evaluiert oder erstellt wurden, am innovativsten?
- Gibt es tangible Input Methoden die in den letzten Jahren auf den Markt gekommen sind?
- Input mit Controllern empfehlenswert (mögliche Vor- und Nachteile)?
- Sind Tokens in öffentlichen Einsatz zu empfehlen? → Probleme damit? z.B. Diebstahl o.ä.
- Erfahrung des Interviewpartners mit Applikationen die Whole-Body-Interaktion verwend(et)en.
- Kann die Art des Inputs (z.B. WBI) die Initialinteraktion/ den Aufforderungscharakter beeinflussen?
- Welche Interfaces gibt es die (Takt-) Schläge bzw. Rhythmus erkennen können?
    - Bzw. welche rapid prototyping Ansätze gibt es? → Güldenpfennig

Benutzer:

- Auf welche Benutzergruppen soll nicht vergessen werden (Farbenblinde Menschen etc.)?
- Kann es sein, dass weniger Leute mit einer Applikation interagieren wenn sie von mehreren Leuten beobachtet werden?

Feedback:

- Sieht der Interviewpartner die Verwendung von analog gespielten Saiten als Pro oder Contra? Was spricht dafür/ dagegen?
- Wie kann zusätzliches Feedback im Mechanismus verbaut werden?
- Wann soll kein Feedback durch den Mechanismus wiedergegeben werden?

MUIs/ TUIs:

- Was soll beim Bau von großen TUIs beachten werden? Gab es Probleme mit in Vergangenheit erstellten TUIs?
- Gabe es musikalische TUIs die dem Interviewpartner gerade in den Sinn kommen?
- Welche guten/ erfolgreichen Musikspiele (ggf. für Nicht-Musiker) fallen dem Interviewpartner ein?
- Sketch(es) mit Interviewpartner besprechen → Brainstorming! Vorschläge, etc.
    - Inputmethoden für Töne (Tonhöhe, Lautstärke, Akkorde, Rhytmen, etc.)

3

APPENDIX C

Mockup Test

Design Principles

- Is it clear, what each control is for (principle of visibility)?
- Is it clear, how the controls are being used (principle of affordance)?
- Does the user miss something (principle of containment)?
- Is the current system status clear (enough or too much feedback)?
- Does the user have the feeling, that he/she has control over the interface (support internal locus of control)?
- Can any errors from the user side be prevented by the interface (prevent errors)?
- Are any interactions easily (and understandably) reversible (permit easy reversal of actions)?

Design Attitudes:

- Flow State
  - Did you have the feeling of being in control over the interface?
    - Did you have the feeling of correctly bringing the interface to do the thing you wanted it to do?
    - Was it hard to achieve the task you set for yourself at any given point?
  - Did something from the interface distract you from your task?
    - What was it?
  - Did you try to create new music after each task you finished?
- Playfulness
  - Did you have fun?
  - What was challenging?

- What did you try to achieve?
  - Was it too easy to achieve?
  - Was it too hard to achieve?
- Did you understand the interface in a comfortable timeframe? Or did you have the feeling of needing too long?

APPENDIX D

Requirements

D.1 Pre- Sketches

Business Requirements

1. ● The system should engage the user with music by letting him/her create melodies in an alternative and fun way.
   Relates to:
   - System requirements 1., 2., and 3.
   - All user requirements
   - Non-functional requirements 1., 2., 3., 4., 5., 10., and 14.
2. ● The development costs of the project must not exceed 1700 Euro
3. ○ The installation should attract more customers (e.g. in a museum)
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 2., 3., 5., 12., 13., and 14.
4. ○ The maximum development time for the three quarters of a year

D.1.1 System Requirements

1. ● The activities that the system enables the user to perform are not collaborative in nature.
   Relates to:
   - All user requirements
   - System requirement 2.
   - Non-functional requirement 8.

2. ● The input from the user must be tangible (no wearables).
   Relates to:
   - All user requirements
   - Non-functional requirements 4., 12., and 14.
3. ● To support the user in the music creation process, musical accompaniment must be enabled by the system.
   Relates to:
   - Business requirement 1.
   - User requirements 1. and 2.
4. ● For the interface communication between the MUI and the analog musical output (the mechatronical part of this thesis), a music protocol, like MIDI or OSC, must be used.
   Relates to:
   - System requirements 7. and 9.
5. ● The generated music must be output via headphones.
   Relates to:
   - Non-functional requirement 5.
6. ● The installation must produce music with analog guitar strings.
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 15. and 16.
7. ● The written software must run on microcontrollers, like the Arduino or Raspberry Pi.
   Relates to:
   - System requirements 4. and 9.
8. ○ The product of this work must be maintainable by non-programmers/external personnel.
   Relates to:
   - System requirement 6.
   - Non-functional requirements 7. and 15.
9. ○ The system should contain as much open source software as possible.
   Relates to:
   - Business requirement 2.
   - System requirements 4. and 7.
10. ○ The guitar strings on the sound creation mechanism must be easy to change.
    Relates to:
    - System requirements 6. and 8.
    - Non-functional requirements 7. and 15.

D.1.2 User Requirements

1. ● The position of tones in the melody and their music parameters (pitch, duration) must be set by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.
2. ● The position of tones in the melody and their music parameters (pitch, duration) must be changeable by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.
3. ● The playback speed (in beats per minute) must be changeable by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.
4. ● The user must have the possibility to change the output volume of the system.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.
5. ● The user can accept or decline tone suggestions by the system, which try to help him in creating a more harmonious melody.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.

D.1.3 Functional Requirements

1. ● The system must be able to capture the current position of each token used.
   Relates to:
   - User requirement 1. and 2.
2. ● The system must have a constant connection to the input devices to set the playback speed and volume.
   Relates to:
   - User requirement 3. and 4.

D.1.4 Non-functional Requirements

1. ● Delay between in- and output shall not impair the user.
   Relates to:
   - Business requirement 1.
   - All user requirements
2. ● The installation must fulfill all attributes for a good usability.
   Relates to:
   - Business requirement 1.
   - All user requirements
3. ● The interface should have a clearly understandable design by adhering to design knowledge.
   Relates to:
   - Business requirement 1.
   - All user requirements
4. ● The system must integrate playfulness.
   Relates to:
   - Business requirement 1.
   - All user requirements
   - System requirement 2.
   - Non-functional requirements 1., 2., and 3.
5. ● The installation has to minimize initial interaction fears of the users.
   Relates to:
   - System requirement 5.
   - Non-functional requirements 14. and 16.
6. ● The installation should be optimized for a interaction time from three to fifteen minutes.
   Relates to:
   - All user requirements
   - Non-functional requirements 1., 2., 3. and 4.
7. ● Mechanical failure must be at an absolute minimum.
   Relates to:
   - System requirements 2., 7., 8. and 10.
8. ● The system should be able to play at melodies with at least 180 bpm.
   Relates to:
   - User requirement 3.

9. ● The pitches precision must not exceed the auditory threshold of about ±8 cents between two frets.

   Relates to:

   – User requirements 1. and 2.
   – System requirement 6.

10. ● The table should not have a round shape

    Relates to:

    – System requirement 1.

11. ● The tokens must be inexpensive (to replace).

    Relates to:

    – Business requirement 2.
    – System requirement 2.

12. ● Colour-blind users have no disadvantage in using the system due to their limitation.

    Relates to:

    – All user requirements
    – System requirement 2.

13. ● The skill of the target group regarding music is low or non-existant, but the system should also work for people with musical knowledge.

    Relates to:

    – All user requirements

14. ● The size of the user should not be an impediment for him or her.

    Relates to:

    – All user requirements
    – System requirement 2.

15. ● The system rarely needs to be reconfigured.

    Relates to:

    – System requirement 8.

16. ○ The system needs to be visually appealing.

    Relates to:

    – Business requirements 1. and 3.
    – System requirements 2. and 6.
    – Non-functional requirements 5. and 8.

D.2 Pre- Mockups

D.2.1 Business Requirements

1. ● The system should engage the user with music by letting him/her create melodies in an alternative and fun way.
   Relates to:
   - System requirements 1. and 2.
   - All user requirements
   - Non-functional requirements 1., 2., 3., 4., 5., 10., and 14.
2. ● The development costs of the project must not exceed 1700 Euro
3. ○ The installation should attract more customers (e.g. in a museum)
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 2., 3., 5., 12., 13., and 14.
4. ○ The maximum development time for the three quarters of a year

D.2.2 System Requirements

1. ● The activities that the system enables the user to perform are not collaborative in nature.
   Relates to:
   - All user requirements
   - System requirement 2.
   - Non-functional requirement 8.
2. ● The input from the user must be tangible (no wearables).
   Relates to:
   - All user requirements
   - Non-functional requirements 4., 12., and 14.
3. ● For the interface communication between the MUI and the analog musical output (the mechatronical part of this thesis), a music protocol, like MIDI or OSC, must be used. (v2)
   Relates to:
   - System requirements 6 and 8
4. ● The generated music must be output via headphones.
   Relates to:
   - Non-functional requirement 5.
5. ● The installation must produce music with analog guitar strings.
   Relates to:

- Business requirement 1.
- Non-functional requirements 15. and 16.

6. ● The written software must run on microcontrollers, like the Arduino or Raspberry Pi. (v2)
   Relates to:
   - System requirements 3 and 8
7. ○ The product of this work must be maintainable by non-programmers/external personnel. (v2)
   Relates to:
   - System requirement 5
   - Non-functional requirements 7. and 15.
8. ○ The system should contain as much open source software as possible. (v2)
   Relates to:
   - Business requirement 2.
   - System requirements 3 and 6
9. ○ The guitar strings on the sound creation mechanism must be easy to change. (v2)
   Relates to:
   - System requirements 5 and 7
   - Non-functional requirements 7. and 15.

D.2.3 User Requirements

1. ● The position of tones in the melody and their music parameters (pitch, duration) must be set by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.
2. ● The position of tones in the melody and their music parameters (pitch, duration) must be changeable by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.
3. ● The playback speed (in beats per minute) must be changeable by the user.
   Relates to:

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 15., and 16.

4. ● The user must have the possibility to change the output volume of the system.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.
5. ● The user can accept or decline tone suggestions by the system, which try to help him in creating a more harmonious melody.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.
6. ● The user can adjust the loop area, in which the set tones will be repeated infinitely.
   Relates to:
   - System requirement 2.
7. ● The user is able to set/change the pre-defined accompaniment for each bar.
   Relates to:
   - System requirement 2.

Functional Requirements

1. ● The system must be able to capture the current position of each token used.
   Relates to:
   - User requirement 1. and 2.
2. ● The system must have a constant connection to the input devices to set the playback speed and volume.
   Relates to:
   - User requirement 3. and 4.
3. ● The visual representation/ feedback of an active tone token on the GUI should snap to the nearest beat and pitch, so that every tone starts at an exact beat and it can also be assigned to exactly one pitch.
   Relates to:
   - User requirements 1. and 2.

D.2.4 Non-functional Requirements

1. ● Delay between in- and output shall not impair the user.
   Relates to:
   - Business requirement 1.
   - All user requirements
2. ● The installation must fulfill all attributes for a good usability.
   Relates to:
   - Business requirement 1.
   - All user requirements
3. ● The interface should have a clearly understandable design by adhering to design knowledge.
   Relates to:
   - Business requirement 1.
   - All user requirements
4. ● The system must integrate playfulness.
   Relates to:
   - Business requirement 1.
   - All user requirements
   - System requirement 2.
   - Non-functional requirements 1., 2., and 3.
5. ● The installation has to minimize initial interaction fears of the users. (v2)
   Relates to:
   - System requirement 4
   - Non-functional requirements 14. and 16.
6. ● The installation should be optimized for a interaction time from three to fifteen minutes.
   Relates to:
   - All user requirements
   - Non-functional requirements 1., 2., 3. and 4.
7. ● Mechanical failure must be at an absolute minimum. (v2)
   Relates to:
   - System requirements 2., 6, 7 and 9
8. ● The system should be able to play at melodies with at least 180 bpm.
   Relates to:
   - User requirement 3.

9. ● The pitches precision must not exceed the auditory threshold of about ±8 cents between two frets. (v2)

   Relates to:

   – User requirements 1. and 2.
   – System requirement 5

10. ● The table should not have a round shape

    Relates to:

    – System requirement 1.

11. ● The tokens must be inexpensive (to replace).

    Relates to:

    – Business requirement 2.
    – System requirement 2.

12. ● Colour-blind users have no disadvantage in using the system due to their limitation.

    Relates to:

    – All user requirements
    – System requirement 2.

13. ● The skill of the target group regarding music is low or non-existant, but the system should also work for people with musical knowledge.

    Relates to:

    – All user requirements

14. ● The size of the user should not be an impediment for him or her.

    Relates to:

    – All user requirements
    – System requirement 2.

15. ● The system rarely needs to be reconfigured. (v2)

    Relates to:

    – System requirement 7

16. ○ The system needs to be visually appealing. (v2)

    Relates to:

    – Business requirements 1. and 3.
    – System requirements 2. and 5
    – Non-functional requirements 5. and 8.

D.3 Pre- First Evaluation

D.3.1 Business Requirements

1. ● The system should engage the user with music by letting him/her create melodies in an alternative and fun way.
   Relates to:
   - System requirements 1. and 2.
   - All user requirements
   - Non-functional requirements 1., 2., 3., 4., 5., 10., and 14.
2. ● The development costs of the project must not exceed 1700 Euro
3. ○ The installation should attract more customers (e.g. in a museum)
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 2., 3., 5., 12., 13., and 14.
4. ○ The maximum development time for the three quarters of a year

D.3.2 System Requirements

1. ● The activities that the system enables the user to perform are not collaborative in nature.
   Relates to:
   - All user requirements
   - System requirement 2.
   - Non-functional requirement 8.
2. ● The input from the user must be tangible (no wearables).
   Relates to:
   - All user requirements
   - Non-functional requirements 4., 12., and 14.
3. ● For the interface communication between the MUI and the analog musical output (the mechatronical part of this thesis), a music protocol, like MIDI or OSC, must be used. (v2)
   Relates to:
   - System requirements 6 and 8
4. ● The generated music must be output via headphones.
   Relates to:
   - Non-functional requirement 5.
5. ● The installation must produce music with analog guitar strings.
   Relates to:

- Business requirement 1.
- Non-functional requirements 15. and 16.

6. ◐ The written software must run on microcontrollers, like the Arduino or Raspberry Pi. (v2)

   Relates to:

   - System requirements 3 and 8

7. ○ The product of this work must be maintainable by non-programmers/external personnel. (v2)

   Relates to:

   - System requirement 5
   - Non-functional requirements 7. and 15.

8. ○ The system should contain as much open source software as possible. (v2)

   Relates to:

   - Business requirement 2.
   - System requirements 3 and 6

9. ○ The guitar strings on the sound creation mechanism must be easy to change. (v2)

   Relates to:

   - System requirements 5 and 7
   - Non-functional requirements 7. and 15.

10. ○ A second headset on the installation allows a second person to listen to the melody while the user creates it.

    Relates to:

    - System requirements 1. and 4
    - Non-functional requirement 10.

D.3.3 User Requirements

1. ● The position of tones in the melody and their music parameters (pitch, duration) must be set by the user.

   Relates to:

   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.

2. ● The position of tones in the melody and their music parameters (pitch, duration) must be changeable by the user.

   Relates to:

  - Business requirement 1.
  - System requirement 2.
  - Non-functional requirements 1., 2., 3., 4., and 16.

3. ● The playback speed (in beats per minute) must be changeable by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.

4. ● The user must have the possibility to change the output volume of the system.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.

5. ● The user can accept or decline tone suggestions by the system, which try to help him in creating a more harmonious melody.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.

6. ● The user can adjust the loop area, in which the set tones will be repeated infinitely.
   Relates to:
   - System requirement 2.

D.3.4 Functional Requirements

1. ● The system must be able to capture the current position of each token used.
   Relates to:
   - User requirement 1. and 2.

2. ● The system must have a constant connection to the input devices to set the playback speed and volume.
   Relates to:
   - User requirement 3. and 4.

3. ● The visual representation/ feedback of an active tone token on the GUI should snap to the nearest beat and pitch, so that every tone starts at an exact beat and it can also be assigned to exactly one pitch.
   Relates to:

- User requirements 1. and 2.

4. ● When four repetitions of the current loop area passed and the user didn't change any tone parameters, visualize a suggestion tone from the pentatonic scale at the position after the last set tone by the user.

   Relates to:

   - User requirement 5.

5. ● On every beat, only one tone can be active. In order to visualize this, the first tone token that has been put on the table's surface should be displayed as active by the system. This is achieved by displaying the visual representation on the GUI in blue, all other visual representations should be kept in gray. Once the position of the first token is changed, the second token is active, and so on (first come first serve logic).

   Relates to:

   - System requirement 2.
   - User requirements 1. and 2.
   - Non-functional requirements 1., 2., and 3.

D.3.5 Non-functional Requirements

1. ● Delay between in- and output shall not impair the user.

   Relates to:

   - Business requirement 1.
   - All user requirements

2. ● The installation must fulfill all attributes for a good usability.

   Relates to:

   - Business requirement 1.
   - All user requirements

3. ● The interface should have a clearly understandable design by adhering to design knowledge.

   Relates to:

   - Business requirement 1.
   - All user requirements

4. ● The system must integrate playfulness.

   Relates to:

   - Business requirement 1.
   - All user requirements
   - System requirement 2.
   - Non-functional requirements 1., 2., and 3.

5. ● The installation has to minimize initial interaction fears of the users. (v2)
   Relates to:
   - System requirement 4
   - Non-functional requirements 14. and 16.
6. ● The installation should be optimized for a interaction time from three to fifteen minutes.
   Relates to:
   - All user requirements
   - Non-functional requirements 1., 2., 3. and 4.
7. ● Mechanical failure must be at an absolute minimum. (v2)
   Relates to:
   - System requirements 2., 6, 7 and 9
8. ● The system should be able to play at melodies with at least 180 bpm.
   Relates to:
   - User requirement 3.
9. ● The pitches precision must not exceed the auditory threshold of about ±8 cents between two frets. (v2)
   Relates to:
   - User requirements 1. and 2.
   - System requirement 5
10. ● The table should not have a round shape
    Relates to:
    - System requirement 1.
11. ● The tokens must be inexpensive (to replace).
    Relates to:
    - Business requirement 2.
    - System requirement 2.
12. ● Colour-blind users have no disadvantage in using the system due to their limitation.
    Relates to:
    - All user requirements
    - System requirement 2.
13. ● The skill of the target group regarding music is low or non-existant, but the system should also work for people with musical knowledge.
    Relates to:
    - All user requirements

14. ○ The size of the user should not be an impediment for him or her.
    Relates to:
    - All user requirements
    - System requirement 2.
15. ○ The system rarely needs to be reconfigured. (v2)
    Relates to:
    - System requirement 7
16. ○ The system needs to be visually appealing. (v2)
    Relates to:
    - Business requirements 1. and 3.
    - System requirements 2. and 5
    - Non-functional requirements 5. and 8.

D.4 Pre- Prototyping

D.4.1 Business Requirements

1. ● The system should engage the user with music by letting him/her create melodies in an alternative and fun way.
   Relates to:
   - System requirements 1. and 2.
   - All user requirements
   - Non-functional requirements 1., 2., 3., 4., 5., 10., and 14.
2. ● The development costs of the project must not exceed 1700 Euro
3. ○ The installation should attract more customers (e.g. in a museum)
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 2., 3., 5., 12., 13., and 14.
4. ○ The maximum development time for the three quarters of a year

System Requirements

1. ● The activities that the system enables the user to perform are not collaborative in nature.
   Relates to:
   - All user requirements
   - System requirement 2.
   - Non-functional requirement 8.

2. ● The input from the user must be tangible (no wearables).
   Relates to:
   - All user requirements
   - Non-functional requirements 4., 12., and 14.
3. ● For the interface communication between the MUI and the analog musical output (the mechatronical part of this thesis), a music protocol, like MIDI or OSC, must be used. (v3)
   Relates to:
   - System requirements 7 and 9
4. ● The generated music must be output via headphones.
   Relates to:
   - Non-functional requirement 5.
5. ● The installation must produce music with analog guitar strings.
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 15. and 16.
6. ● The system needs to visualize some kind of orientation for the user, so that he has fewer problems distinguishing the different possible beat and pitch positions for the tone tokens.
   Relates to:
   - User requirements 1. and 2.
7. ● The written software must run on microcontrollers, like the Arduino or Raspberry Pi. (v3)
   Relates to:
   - System requirements 3 and 9
8. ○ The product of this work must be maintainable by non-programmers/external personnel. (v2)
   Relates to:
   - System requirement 5
   - Non-functional requirements 7. and 15.
9. ○ The system should contain as much open source software as possible. (v3)
   Relates to:
   - Business requirement 2.
   - System requirements 3 and 7
10. ○ The guitar strings on the sound creation mechanism must be easy to change. (v3)
    Relates to:
    - System requirements 5 and 8

– Non-functional requirements 7. and 15.

11. ○ A second headset on the installation allows a second person to listen to the melody while the user creates it.

    Relates to:

    – System requirements 1. and 4
    – Non-functional requirement 10.

D.4.2 User Requirements

1. ● The position of tones in the melody and their music parameters (pitch, duration) must be set by the user.

   Relates to:

   – Business requirement 1.
   – System requirement 2.
   – Non-functional requirements 1., 2., 3., 4., and 16.

2. ● The position of tones in the melody and their music parameters (pitch, duration) must be changeable by the user.

   Relates to:

   – Business requirement 1.
   – System requirement 2.
   – Non-functional requirements 1., 2., 3., 4., and 16.

3. ● The playback speed (in beats per minute) must be changeable by the user.

   Relates to:

   – Business requirement 1.
   – System requirement 2.
   – Non-functional requirements 1., 2., 3., 15., and 16.

4. ● The user must have the possibility to change the output volume of the system.

   Relates to:

   – Business requirement 1.
   – System requirement 2.
   – Non-functional requirements 1., 2., 3., 15., and 16.

5. ● The user can accept or decline tone suggestions by the system, which try to help him in creating a more harmonious melody.

   Relates to:

   – Business requirement 1.
   – System requirement 2.

  - Non-functional requirements 1., 2., 3., 15., and 16.

6. ● The user can adjust the loop area, in which the set tones will be repeated infinitely.
   Relates to:
   - System requirement 2.

D.4.3 Functional Requirements

1. ● The system must be able to capture the current position of each token used.
   Relates to:
   - User requirement 1. and 2.
2. ● The system must have a constant connection to the input devices to set the playback speed and volume.
   Relates to:
   - User requirement 3. and 4.
3. ● The visual representation/ feedback of an active tone token on the GUI should snap to the nearest beat and pitch, so that every tone starts at an exact beat and it can also be assigned to exactly one pitch.
   Relates to:
   - User requirements 1. and 2.
4. ● On every beat, one tone per string can be active. In order to visualize this, the last tone token that has been put on the table's surface should be displayed as active by the system. This is achieved by displaying the visual representation in color of the coherent string, all other visual representations should be kept in gray. Once the position of the last token is changed, the second to last token is active, and so on (last come last serve logic). (v2)
   Relates to:
   - System requirement 2.
   - User requirements 1. and 2.
   - Non-functional requirements 1., 2., and 3.

D.4.4 Non-functional Requirements

1. ● Delay between in- and output shall not impair the user.
   Relates to:
   - Business requirement 1.
   - All user requirements
2. ● The installation must fulfill all attributes for a good usability.
   Relates to:
   - Business requirement 1.

- All user requirements

3. ● The interface should have a clearly understandable design by adhering to design knowledge.
   Relates to:
   - Business requirement 1.
   - All user requirements
4. ● The system must integrate playfulness.
   Relates to:
   - Business requirement 1.
   - All user requirements
   - System requirement 2.
   - Non-functional requirements 1., 2., and 3.
5. ● The installation has to minimize initial interaction fears of the users. (v2)
   Relates to:
   - System requirement 4
   - Non-functional requirements 14. and 16.
6. ● The installation should be optimized for a interaction time from three to fifteen minutes.
   Relates to:
   - All user requirements
   - Non-functional requirements 1., 2., 3. and 4.
7. ● Mechanical failure must be at an absolute minimum. (v3)
   Relates to:
   - System requirements 2., 7, 8 and 10
8. ● The system should be able to play at melodies with at least 180 bpm.
   Relates to:
   - User requirement 3.
9. ● The pitches precision must not exceed the auditory threshold of about ±8 cents between two frets. (v2)
   Relates to:
   - User requirements 1. and 2.
   - System requirement 5
10. ● The table should not have a round shape
    Relates to:
    - System requirement 1.

11. ● The tokens must be inexpensive (to replace).
    Relates to:
    - Business requirement 2.
    - System requirement 2.
12. ● Colour-blind users have no disadvantage in using the system due to their limitation.
    Relates to:
    - All user requirements
    - System requirement 2.
13. ● The skill of the target group regarding music is low or non-existant, but the system should also work for people with musical knowledge.
    Relates to:
    - All user requirements
14. ● The size of the user should not be an impediment for him or her.
    Relates to:
    - All user requirements
    - System requirement 2.
15. ● The system rarely needs to be reconfigured. (v3)
    Relates to:
    - System requirement 8
16. ○ The system needs to be visually appealing. (v2)
    Relates to:
    - Business requirements 1. and 3.
    - System requirements 2. and 5
    - Non-functional requirements 5. and 8.

D.5 Pre- Second Evaluation

D.5.1 Business Requirements

1. ● The system should engage the user with music by letting him/her create melodies in an alternative and fun way.
   Relates to:
   - System requirements 1. and 2.
   - All user requirements
   - Non-functional requirements 1., 2., 3., 4., 5., 10., and 14.
2. ● The development costs of the project must not exceed 1700 Euro

3. ○ The installation should attract more customers (e.g. in a museum)
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 2., 3., 5., 12., 13., and 14.
4. ○ The maximum development time for the three quarters of a year

D.5.2 System Requirements

1. ● The activities that the system enables the user to perform are not collaborative in nature.
   Relates to:
   - All user requirements
   - System requirement 2.
   - Non-functional requirement 8.
2. ● The input from the user must be tangible (no wearables).
   Relates to:
   - All user requirements
   - Non-functional requirements 4., 12., and 14.
3. ● For the interface communication between the MUI and the analog musical output (the mechatronical part of this thesis), a music protocol, like MIDI or OSC, must be used. (v3)
   Relates to:
   - System requirements 7 and 9
4. ● The generated music must be output via headphones.
   Relates to:
   - Non-functional requirement 5.
5. ● The installation must produce music with analog guitar strings.
   Relates to:
   - Business requirement 1.
   - Non-functional requirements 15. and 16.
6. ● The system needs to visualize some kind of orientation for the user, so that he has fewer problems distinguishing the different possible beat and pitch positions for the tone tokens.
   Relates to:
   - User requirements 1. and 2.
   - Functional requirement 10
7. ○ The written software must run on microcontrollers, like the Arduino or Raspberry Pi. (v3)
   Relates to:

  - System requirements 3 and 9

8. ○ The product of this work must be maintainable by non-programmers/external personnel. (v2)
   Relates to:
   - System requirement 5
   - Non-functional requirements 7. and 15.
9. ○ The system should contain as much open source software as possible. (v3)
   Relates to:
   - Business requirement 2.
   - System requirements 3 and 7
10. ○ The guitar strings on the sound creation mechanism must be easy to change. (v3)
    Relates to:
    - System requirements 5 and 8
    - Non-functional requirements 7. and 15.
11. ○ A second headset on the installation allows a second person to listen to the melody while the user creates it.
    Relates to:
    - System requirements 1. and 4
    - Non-functional requirement 10.

D.5.3 User Requirements

1. ● The position of tones in the melody and their music parameters (pitch, duration) must be set by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.
2. ● The position of tones in the melody and their music parameters (pitch, duration) must be changeable by the user.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 4., and 16.
3. ● The playback speed (in beats per minute) must be changeable by the user.
   Relates to:

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 15., and 16.

4. ● The user must have the possibility to change the output volume of the system.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.
5. ● The user can accept or decline tone suggestions by the system, which try to help him in creating a more harmonious melody.
   Relates to:
   - Business requirement 1.
   - System requirement 2.
   - Non-functional requirements 1., 2., 3., 15., and 16.
6. ● The user can adjust the loop area, in which the set tones will be repeated infinitely.
   Relates to:
   - System requirement 2.

D.5.4 Functional Requirements

1. ● The system must be able to capture the current position of each token used.
   Relates to:
   - User requirement 1. and 2.
2. ● The system must have a constant connection to the input devices to set the playback speed and volume.
   Relates to:
   - User requirement 3. and 4.
3. ● The visual representation/ feedback of an active tone token on the GUI should snap to the nearest beat and pitch, so that every tone starts at an exact beat and it can also be assigned to exactly one pitch.
   Relates to:
   - User requirements 1. and 2.
4. ● The joker token is a type of tokens with different length. When being put on the tables surface, a random tune is being selected and displayed on the GUI, with the same length as the token itself.
   Relates to:
   - User requirement 1. and 2.

  - System requirement 2.

5. ● On every beat, one tone per string can be active. In order to visualize this, the last tone token that has been put on the table's surface should be displayed as active by the system. This is achieved by displaying the visual representation in color of the coherent string, all other visual representations should be kept in gray. Once the position of the last token is changed, the second to last token is active, and so on (last come last serve logic). (v3)

   Relates to:

   - System requirement 2.
   - User requirements 1. and 2.
   - Functional requirement 5
   - Non-functional requirements 1., 2., and 3.

6. ● If one part of a tone token overlaps on the same beat and string with another tone token, the later layed token gets deactivated as a whole.

   Relates to:

   - User requirements 1. and 2.
   - Functional requirement 5

7. ● If a tone token's full length is interrupted by a loop bar, the tone is being played until, or from the beginning of, the loop token.

   Relates to:

   - User requirements 1., 2., and 6
   - System requirement 2.

8. ● If a tone token gets moved while its being played, the visual representation of the tone on the GUI stays at it's position until the tone is finished.

   Relates to:

   - User requirements 1. and 2.
   - System requirement 2.

9. ● If a tone token gets put on a beat which is being played at the moment, the token does not get played until none of the beats are being played.

   Relates to:

   - User requirements 1. and 2.
   - System requirement 2.

10. ● If the user moves a tone token, lines at the horizontal borders of the interaction area and at the loop bars need to be visualized at the same height and with the same width as the token itself. A scale, also located at the loop bars and the horizontal borders of the interaction area, thus gives information about the current position and width of the token in dependence of all possible positions/ pitches.

    Relates to:

    - User requirements 1. and 2.
    - System requirement 2.

D.5.5 Non-functional Requirements

1. ● Delay between in- and output shall not impair the user.
   Relates to:
   - Business requirement 1.
   - All user requirements
2. ● The installation must fulfill all attributes for a good usability.
   Relates to:
   - Business requirement 1.
   - All user requirements
3. ● The interface should have a clearly understandable design by adhering to design knowledge.
   Relates to:
   - Business requirement 1.
   - All user requirements
4. ● The system must integrate playfulness.
   Relates to:
   - Business requirement 1.
   - All user requirements
   - System requirement 2.
   - Non-functional requirements 1., 2., and 3.
5. ● The installation has to minimize initial interaction fears of the users. (v2)
   Relates to:
   - System requirement 4
   - Non-functional requirements 14. and 16.
6. ● The installation should be optimized for a interaction time from three to fifteen minutes.
   Relates to:
   - All user requirements
   - Non-functional requirements 1., 2., 3. and 4.
7. ● Mechanical failure must be at an absolute minimum. (v3)
   Relates to:
   - System requirements 2., 7, 8 and 10
8. ● The system should be able to play at melodies with at least 180 bpm.
   Relates to:
   - User requirement 3.

9. ● The pitches precision must not exceed the auditory threshold of about ±8 cents between two frets. (v2)
   Relates to:
   - User requirements 1. and 2.
   - System requirement 5
10. ● The table should not have a round shape
    Relates to:
    - System requirement 1.
11. ● The tokens must be inexpensive (to replace).
    Relates to:
    - Business requirement 2.
    - System requirement 2.
12. ● Colour-blind users have no disadvantage in using the system due to their limitation.
    Relates to:
    - All user requirements
    - System requirement 2.
13. ● The skill of the target group regarding music is low or non-existant, but the system should also work for people with musical knowledge.
    Relates to:
    - All user requirements
14. ● The size of the user should not be an impediment for him or her.
    Relates to:
    - All user requirements
    - System requirement 2.
15. ● The system rarely needs to be reconfigured. (v3)
    Relates to:
    - System requirement 8
16. ○ The system needs to be visually appealing. (v2)
    Relates to:
    - Business requirements 1. and 3.
    - System requirements 2. and 5
    - Non-functional requirements 5. and 8.

APPENDIX E

Use Cases

E.1 Pre- Sketches

E.1.1 Create Melody

Description

The user begins to manipulate sound parameters through a mixture of exploration and intuitive interaction with tangible input tokens. Concrete connections between changes of the user interface and acoustic feedback (output) through the mechanical construction are recognized by the user. Based on this knowledge the user tries to compose simple melodies. When the music is created, the transmitted information to the user is mainly on a visual level.

Flow of Events

Unknown at the current point of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation and put the headphones on.

Postconditions

None.

Extension Points

The user can change the input (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the software components of the system?
  - Q: How do the software components interact with each other (Interfaces)?
  - Q: What music creation/ editing philosophy is the system based on?

    A: The system uses the loop approach, where a selected area of the available beats are played in a loop. The loop area can also be changed throughout the users interaction.
  - Q: How many bars/ beats do exist?
  - Q: How many tones can be played?

    A: The exact number of playable tones is currently unclear. What is certain at the moment is, that the melody created by the user will be played on three guitar strings (from G, B and e string).
  - Q: How many tones can be played per beat?
  - Q: What principle is the accompaniment based on?

    A: There are currently two possibilities: Either restrict the possibility of chord composition based on the used musical scale or use predefined samples and transpose them if necessary.
  - Q: What predefined samples or chords should be used to accompany the users input?
  - Q: How long does the accompaniment take? Will it be repeated afterwards? How often?
- Hardware
  - Q: Should the accompaniment also be created analogy?

    A: According to the experts being interviewed, this is a valid approach. Three strings will be used for the melody and three strings for the accompaniment.
  - Q: What are the hardware components of the MUI?
  - Q: What hardware does the software run on?
  - Q: What material/ components is the table made of?
  - Q: What height does the table have to be?
- Input
  - Q: How do the tokens look like?
  - Q: How big should one token be?
  - Q: Is there any difference between the tokens? Are there any different token types?
  - Q: How can the tokens help the user to understand the current system status?
  - Q: What is the musical metaphor of a token (single tone, a tune, a chord)?
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?
  - Q: Can the user choose between accompaniment melodies or are they chosen by the system?

  - Q: How can the accompaniment be selected (if the user has the possibility to do so)?
  - Q: Should it be possible to adjust the volume of the accompaniment independently to the overall volume?

- Feedback
  - Q: How can the current system status be displayed comprehensively?
  - Q: What colors should be used for which system status/ property?
  - Q: How can the user be supported without given them the feeling that the system does not do something they don't want to?
  - Q: What visual feedback should be shown to the user?
  - Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?
  - Q: What feedback does the user get concerning the accompaniment?

E.1.2 Edit Melody

Description

The user edits the previously created music by rearranging the tangible objects. Editing is done according to the similar principle of creating music (see Use Case 1).

Flow of Events

Not known at current state of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: Can melodies be persisted and loaded?
  - Q: How can the system support the user, so that his/ her short term memory load can be reduced?

- Input
  - Q: How does the user change the already defined tones?
  - Q: How does the user remove/delete already defined tones?

E.1.3 Adjust Playback Speed

Description

The user lets the current melody play faster or slower by using the associated input device. By using this device, he/ she is able to let increase or decrease the playback speed until a certain limit. The system reacts accordingly and plays the created melody faster/ slower.

Flow of Events

The current position of the knob gets recognized by the hardware and sent to the software which let's the visual feedback reflect the current playback speed and tells the mechatronical part of the system to pluck the strings accordingly to the set speed.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the minimum and maximum bpm being allowed by the system?
- Hardware
  - Q: Use continuous transition between bpm or stages of bpm?
  - Q: If stages are being used, how big are they?
- Input
  - Q: Where on the installation should the device to adjust the playback speed be placed?

  - Q: Which input device should be used to adjust the playback speed?

- Feedback
  - Q: What is the haptic feedback when turning the input device for the playback speed?
  - Q: What icons can be used as a visual metaphor for *(*fast) and *(*slow)?

E.1.4 Adjust Playback Volume

Description

The user adjusts the volume output via a physical input device on the installation. The adjusted sound volume then gets output via headphones the user wears. When adjusting the volume, haptic and/or visual feedback is returned.

Flow of Events

The current position of the knob (or similar input device) gets read by the hardware and sent to the mechatronical part of the system which in- or decrease the volume accordingly to the set position/value of the knob.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Hardware
  - Q: Which device should be used to output the sound?
    A: The user wears headphones (see Section 11.2).
  - Q: Should there be more than one pair of headphones?
- Input
  - Q: What input device should be chosen to adjust the volume?
  - Q: Where on the installation will the device to adjust the volume be placed?
- Feedback
  - Q: What icons should be used for the visual representation of the sound volume?

E.1.5 Accept or Decline Suggestions

Description

During the creation of the melody, help for the token(s) to be set gets displayed on the user interface. The user can either accept the suggestion (interact with it) or decline it (actively decline or ignore).

Flow of Events

Due to many ambiguities not yet predictable at the moment.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2), change the volume (see Use Case 4) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Software
  - Q: Where is the suggestion displayed?
- Feedback
  - Q: When is the suggestion displayed? What technical parameters trigger them?
  - Q: How does the suggestion visualization look like?
  - Q: What is the right amount of suggestions? The user should not feel that the creation process is being taken away from him.
  - Q: Does the suggestion need to be non-intrusive?
  - Q: How can the suggestion be non-intrusive?
- Input
  - Q: How can the user accept or decline suggestions?

E.2 Pre- Mockups

E.2.1 Create Melody

Description

The user begins to manipulate sound parameters through a mixture of exploration and intuitive interaction with tangible input tokens. Concrete connections between changes of the user interface and acoustic feedback (output) through the mechanical construction are recognized by the user. Based on this knowledge the user tries to compose simple melodies. When the music is created, the transmitted information to the user is mainly on a visual level.

Flow of Events

Unknown at the current point of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation and put the headphones on.

Postconditions

None.

Extension Points

The user can change the input (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the software components of the system?
  - Q: How do the software components interact with each other (Interfaces)?
  - Q: What music creation/ editing philosophy is the system based on?
    A: The system uses the loop approach, where a selected area of the available beats are played in a loop. The loop area can also be changed throughout the users interaction.
  - Q: How many bars/ beats do exist?
    A: There exist four bars consisting of four beats, therefore sixteen beats in total.
  - Q: How many tones can be played?
    A: The user has the possibility to play 23 different tones (see Section 11.2) on three strings. One tone is always played on one certain string to prevent confusion.
  - Q: How many tones can be played per beat?

- Hardware
  - Q: What are the hardware components of the MUI?
  - Q: What hardware does the software run on?
  - Q: What material/ components is the table made of?
  - Q: What height does the table have to be?
- Input
  - Q: How do the tokens look like?

    A: One of the approaches has two different tone- token approaches. In the first approach, the tokens have a rectangular form, whereas in the second approach, the token have a circular form. The loop area token should differ in form and icon atop of the token, there is currently nothing more defined.
  - Q: How big should one token be?
  - Q: Is there any difference between the tokens? Are there any different token types?

    A: There are currently two types of token. One for the adjustment of the loop area and one acts as a physical metaphor for tones.
  - Q: How can the tokens help the user to understand the current system status?

    A: A token directly can't help displaying the current system status, but the system recognizing the token displays a colored rectangle beneath the token. This token is slightly larger than the token itself and has the color of the corresponding string it will be played on.
  - Q: What is the musical metaphor of a token (single tone, a tune, a chord)?

    A: One token resembles a single tone. Tokens differ in width, which reflects the tone length.
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?

    A: The token are not limited in any way
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?
- Feedback
  - Q: How can the current system status be displayed comprehensively?

    A: Apart from the mentioned visual feedback, this question will be answered in the following evaluation
  - Q: What colors should be used for which system status/ property?

    A: Currently colors are not being used for the display of any property or status.
  - Q: How can the user be supported without given them the feeling that the system does not do something they don't want to?

    A: The user can be supported in determining the right set-in of tones when creating a melody. This is achieved by the use of the so-called snapping. Snapping works as follows: As soon as the user puts a token on the interaction area, thy system recognizes it and responds with visual feedback in form of a colored rectangle which is slightly larger than a token. This rectangle is, as already explained, displayed beneath the token. When the token doesn't move, the center of the rectangle automatically

moves horizontally to the nearest beat and stays there. The same applies on the vertical axis but with the nearest pitch. To the user this looks like the rectangle snaps to a position that the system considers valid. When moving the token again, the rectangle is again displayed directly underneath the token.

- Q: What visual feedback should be shown to the user?

  A: The visual feedback consists of every recognized tone on the interaction area, a loop start and loop end bar, a line which moves from the start bar to the end bar (and therefore displays the current position in the melody) and the base-tone of the chosen power chords.

- Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?

  A: No, this would be too expensive and would interfere with non-functional requirement number 11..

E.2.2 Edit Melody

Description

The user edits the previously created music by rearranging the tangible objects. Editing is done according to the similar principle of creating music (see Use Case 1).

Flow of Events

Not known at current state of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: Can melodies be persisted and loaded?

    A: No. Melodies can not be persisted or loaded. If the Drawing Approach will be pursued further, this might be a possible addition to the installation.

  - Q: How can the system support the user, so that his/ her short term memory load can be reduced?

- Input
  - Q: How does the user change the already defined tones?

    A: Depending on the approach, there are different ways of changing the length and position of the defined tones. With the rectangular token, the token itself must be replaced with another token to change to the desired length. Changing the tone length with the cylindrical tokens is achieved by turning the token (anti-) clockwise. The Drawing Approach has no intended interaction to change the tone length. It must be therefore deleted and set again to change its length. Changing the tone position in the approaches using token can be achieved by simply moving the token on the interaction surface. The Drawing Approach achieves this by continuously touching an existing tone with the pen and moving the pen over the interaction surface, whereas the visual representation of the tone follows the pen.
  - Q: How does the user remove/delete already defined tones?

    A: All token using approaches delete existing tones by removing the token from the interaction surface on the tabletop. The Drawing Approach deletes tones by touching them with the eraser, the second tangible interaction device of this approach.

E.2.3 Adjust Playback Speed

Description

The user lets the current melody play faster or slower by using the associated input device. By using this device, he/ she is able to let increase or decrease the playback speed until a certain limit. The system reacts accordingly and plays the created melody faster/ slower.

Flow of Events

The current position of the knob gets recognized by the hardware and sent to the software which let's the visual feedback reflect the current playback speed and tells the mechatronical part of the system to pluck the strings accordingly to the set speed.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the minimum and maximum bpm being allowed by the system?
    A: At least 180 bpm must be possible (see Subsection 9).
- Hardware
  - Q: Use continuous transition between bpm or stages of bpm?
  - Q: If stages are being used, how big are they?
- Input
  - Q: Where on the installation should the device to adjust the playback speed be placed?
  - Q: Which input device should be used to adjust the playback speed?
- Feedback
  - Q: What is the haptic feedback when turning the input device for the playback speed?
  - Q: What icons can be used as a visual metaphor for *(*fast) and *(*slow)?

E.2.4 Adjust Playback Volume

Description

The user adjusts the volume output via a physical input device on the installation. The adjusted sound volume then gets output via headphones the user wears. When adjusting the volume, haptic and/or visual feedback is returned.

Flow of Events

The current position of the knob (or similar input device) gets read by the hardware and sent to the mechatronical part of the system which in- or decrease the volume accordingly to the set position/value of the knob.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Hardware
  - Q: Which device should be used to output the sound?
    A: The user wears headphones (see Section 11.2).
  - Q: Should there be more than one pair of headphones?
- Input
  - Q: What input device should be chosen to adjust the volume?
  - Q: Where on the installation will the device to adjust the volume be placed?
- Feedback
  - Q: What icons should be used for the visual representation of the sound volume?

E.2.5 Accept or Decline Suggestions

Description

During the creation of the melody, help for the token(s) to be set gets displayed on the user interface. The user can either accept the suggestion (interact with it) or decline it (actively decline or ignore).

Flow of Events

Due to many ambiguities not yet predictable at the moment.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2), change the volume (see Use Case 4) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Software
  - Q: Where is the suggestion displayed?
- Feedback
  - Q: When is the suggestion displayed? What technical parameters trigger them?
  - Q: How does the suggestion visualization look like?
  - Q: What is the right amount of suggestions? The user should not feel that the creation process is being taken away from him.
  - Q: Does the suggestion need to be non-intrusive?
  - Q: How can the suggestion be non-intrusive?
- Input
  - Q: How can the user accept or decline suggestions?

E.2.6 Adjust Loop Area

Description

The so called loop area describes a part on the interaction area which is set by the user. This area indefinitely repeats every active tone in it.

Special Requirements

- System requirement 2.

Flow of Events

Currently not know.

Preconditions

The user approached the installation, put the headphones on and created a melody.

Postconditions

None.

Extension Points

The user can edit the created music (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) or change the accompaniment (see Use Case 7) at any given point of the interaction.

Resulting Questions

- Input
  - Q: How does the user change the loop area?
    A: There is a second kind of token, with which the user is able to change the loop area.

- Feedback
  - Q: How is the loop area being displayed to the user?
    A: The loop area is being visualized with the help of two vertical bars across the interaction area.

E.2.7 Set Accompaniment

Description

The user is able to set and change the accompaniment, in the form of a power chord, for each of the four bars available to him. Each bar has one turning knob beneath the interaction surface. The base tone of the defined power chord is beneath the turning position of the knob and will be highlighted with the help of a glowing LED next to the base tones name.

Flow of Events

Currently unknown.

Preconditions

The user approached the installation, put the headphones on and created a melody.

Postconditions

None.

Extension Points

The user can edit the created music (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) or change the loop area (see Use Case 6) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What principle is the accompaniment based on?
    A: There are currently two possibilities: Either restrict the possibility of chord composition based on the used musical scale or use predefined samples and transpose them if necessary.
  - Q: What predefined samples or chords should be used to accompany the users input?
    A: Power chords will be used to accompany the user. It is a simple way of supporting the users melody, needs no runtime computation, fits to every melody and can be played on three strings.
  - Q: How long does the accompaniment take? Will it be repeated afterwards? How often?
    A: One power chords is set for one bar. The power chord starts with the first beat and ends with the last beat of the coherent bar (or loop area, if it is smaller than one bar).
  - Q: Should it be possible to adjust the volume of the accompaniment independently to the overall volume?

- Hardware
  - Q: Should the accompaniment also be created analogy?
    A: Yes, the accompaniment will be play analogy on the three lower strings of the mechanical part of the installation.
- Input
  - Q: Can the user choose between accompaniment melodies or are they chosen by the system?
    A: The user can choose the power chords base tone for each bar.
- Feedback
  - Q: What feedback does the user get concerning the accompaniment?
    A: The current accompaniment for one bar is highlighted with an active LED at the current settings of the knob, which sets the power chord. Additionally, each possible power chord has it's base tone written next to it.

E.3 Pre- First Evaluation

E.3.1 Create Melody

Description

The creation of a melody is solely achieved by using the tokens of the respective approach. Every other interaction with the system through other input devices is covered in the other use cases. The user has several tokens to choose from to create music with rectangular tokens. They differ in width, but not in height or depth. The width of the token is considered a physical methaphor for tone length. The wider a token is, the longer it is played. There are four token with different widths to choose from. As soon as the user places a token on the interaction surface, a projected rectangle appears directly beneath the token, which provides information about its current state to the user. If the rectangle is grey, the tone is not played, as it is deemed inactive by the system. If the rectangle is colored, the tone is considered active and will be played by the system. If a token is placed at a position where there is already an active tone, the new token is set to inactive and the old tone is continued to be played.

This also applies to the blank approach with cylindrical tokens. The difference to the rectangular tokens lies, apart from the form of the tokens, in the determination of the tone length. There are no tokens of different width here, because the tone length is determined by rotating the tokens. As soon as the user places a token on the interaction surface, not only a circle appears below it, which gives information about the activity of the token, but also a bar, which reveals the tone length.

A pen is used for the last possibility of an tangible input device. This metaphorical pen is picked up by the user and drawn over the interaction surface with its tip. A new tone ends as soon as the user either lifts the pen or moves beyond the range of the current tone on the y-axis. As soon as this happens, the old tone is stopped and a new tone with the new pitch is started at the respective position.

Flow of Events

Unknown at the current point of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation and put the headphones on.

Postconditions

None.

Extension Points

The user can change the input (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the software components of the system?
  - Q: How do the software components interact with each other?
  - Q: What music creation/ editing philosophy is the system based on?
    A: The system uses the loop approach, where a selected area of the available beats are played in a loop. The loop area can also be changed throughout the users interaction.
  - Q: How many bars/ beats do exist?
    A: There exist four bars consisting of four beats, therefore sixteen beats in total.
  - Q: How many tones can be played?
    A: The user has the possibility to play 23 different tones (see Section 11.2) on three strings. One tone is always played on one certain string to prevent confusion.
  - Q: How many tones can be played per beat?
    A: The system plays one tone per beat. Multiple tones will be sorted out by the first come first serve principle.
  - Q: What happens when multiple tones overlap?
    A: The first token to be placed on a certain beat will be played as long as it lays there. If the token gets removed, the token which has been laid there after the now removed token will be played as long as it gets removed and so on. All inactive token are visualized by a gray rectangle underneath the token, whereas the active token's rectangle is colored in blue.
- Hardware
  - Q: What are the hardware components of the MUI?
  - Q: What hardware does the software run on?

  - Q: What material/ components is the table made of?
  - Q: What height does the table have to be?

- Input
  - Q: How do the tokens look like?

    A: One of the approaches has two different tone- token approaches. In the first approach, the tokens have a rectangular form, whereas in the second approach, the token have a circular form. The loop area token should differ in form and icon atop of the token, there is currently nothing more defined.
  - Q: How big should one token be?
  - Q: Is there any difference between the tokens? Are there any different token types?

    A: There are currently two types of token. One for the adjustment of the loop area and one acts as a physical metaphor for tones.
  - Q: How can the tokens help the user to understand the current system status?

    A: Token itself can't, but the visual feedback of the token and the first come first serve functionality can
  - Q: What is the musical metaphor of a token (single tone, a tune, a chord)?

    A: One token resembles a single tone. Tokens differ in width, which reflects the tone length.
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?

    A: The token are not limited in any way
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?

    A: The system's behaviour on special input is covered in the form of functional requirements in the further course of this thesis.

- Feedback
  - Q: How can the current system status be displayed comprehensively?

    A: Apart from the mentioned visual feedback, this question will be answered in the following evaluation
  - Q: What colors should be used for which system status/ property?

    A: Currently colors are not being used for the display of any property or status.
  - Q: How can the user be supported without given them the feeling that the system does not do something they don't want to?

    A: The user can be supported in determining the right set-in of tones when creating a melody. This is achieved by the use of the so-called snapping. Snapping works as follows: As soon as the user puts a token on the interaction area, thy system recognizes it and responds with visual feedback in form of a colored rectangle which is slightly larger than a token. This rectangle is, as already explained, displayed beneath the token. When the token doesn't move, the center of the rectangle automatically moves horizontally to the nearest beat and stays there. The same applies on the vertical axis but with the nearest pitch. To the user this looks like the rectangle snaps to a position that the system considers valid. When moving the token again, the rectangle is again displayed directly underneath the token.

– Q: What visual feedback should be shown to the user?
  A: The visual feedback consists of every recognized tone on the interaction area, a loop start and loop end bar, a line which moves from the start bar to the end bar (and therefore displays the current position in the melody) and the base-tone of the chosen power chords.
– Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?
  A: No, this would be too expensive and would interfere with non-functional requirement number 11..

E.3.2 Edit Melody

Description

Editing music in this case means changing tone lengths and heights using tokens. This means, that a precondition of this use case is, that tokens are already present on the interaction area of the table.

The tone length can be changed with the rectangular tokens by exchanging the tokens. To change the pitch and the position in the melody, a token must be moved on the y- or x-axis of the interaction area. To delete a tone, the user simply removes the token from the interaction area. Changing the pitch and position with cylindrical tokens is done in the same way as with the rectangular token. To change the tone length, the user must rotate the token. If the token is rotated clockwise, the visually displayed bar under the token grows from left to right, and if it is rotated in the opposite direction, the bar shrinks. The removal of sounds is achieved by the user removing tokens.

With the Drawing Approach, the user can change tones by touching the visual representation of the token on the interaction surface with the tip of the pen and pulling it across the table to the new desired position while the touch is held up. The change in tone length is achieved by deleting and recreating the tone. The sound is erased by touching the visual representation with the eraser.

Flow of Events

Not known at current state of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: Can melodies be persisted and loaded?

    A: No. Melodies can not be persisted or loaded. If the Drawing Approach will be pursued further, this might be a possible addition to the installation.
  - Q: How can the system support the user, so that his/ her short term memory load can be reduced?

    A: The first come first serve functionality does help the user in not having to remember each token and its order in the token placement.
- Input
  - Q: How does the user change the already defined tones?

    A: Depending on the approach, there are different ways of changing the length and position of the defined tones. With the rectangular token, the token itself must be replaced with another token to change to the desired length. Changing the tone length with the cylindrical tokens is achieved by turning the token (anti-) clockwise. The Drawing Approach has no intended interaction to change the tone length. It must be therefore deleted and set again to change its length. Changing the tone position in the approaches using token can be achieved by simply moving the token on the interaction surface. The Drawing Approach achieves this by continuously touching an existing tone with the pen and moving the pen over the interaction surface, whereas the visual representation of the tone follows the pen.
  - Q: How does the user remove/delete already defined tones?

    A: All token using approaches delete existing tones by removing the token from the interaction surface on the tabletop. The Drawing Approach deletes tones by touching them with the eraser, the second tangible interaction device of this approach.

E.3.3 Adjust Playback Speed

Description

To adjust the playback speed, the user has to turn the according knob on the left side of the table, below the interaction surface. The purpose of the knob is visualized to the user by the use of specific icons (a tortoise and a running rabbit).

Flow of Events

The current position of the knob gets recognized by the hardware and sent to the software which let's the visual feedback reflect the current playback speed and tells the mechatronical part of the system to pluck the strings accordingly to the set speed.

Special Requirements

- Business requirement 1.

- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the minimum and maximum bpm being allowed by the system?
    A: At least 180 bpm must be possible (see Subsection 9).
- Hardware
  - Q: Use continuous transition between bpm or stages of bpm?
  - Q: If stages are being used, how big are they?
- Input
  - Q: Where on the installation should the device to adjust the playback speed be placed?
  - Q: Which input device should be used to adjust the playback speed?
    A: A turnable knob with a minimum and maximum settings will be used.
- Feedback
  - Q: What is the haptic feedback when turning the input device for the playback speed?
  - Q: What icons can be used as a visual metaphor for *fast* and *slow*?
    A: A turtle was chosen as a metaphor for slow playback and a hare as a metaphor for fast playback.

E.3.4 Adjust Playback Volume

Description

To adjust the playback volume, the user has to turn the according knob on the left side of the table, below the interaction surface. The purpose of the knob is visualized to the user by the use of specific icons (two speaker with a different number of sound waves emerging from them).

Flow of Events

The current position of the knob (or similar input device) gets read by the hardware and sent to the mechatronical part of the system which in- or decrease the volume accordingly to the set position/value of the knob.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Hardware
  - Q: Which device should be used to output the sound?
    A: The user wears headphones (see Section 11.2).
  - Q: Should there be more than one pair of headphones?
    A: It is planned to have two pair of headphones at the table. The position of the second pair of headphones is important, as it can imply unintended properties of the system.
- Input
  - Q: What input device should be chosen to adjust the volume?
    A: A turnable knob with a minimum and maximum settings will be used.
  - Q: Where on the installation will the device to adjust the volume be placed?
- Feedback
  - Q: What icons should be used for the visual representation of the sound volume?
    A: The two icons for the sound volume will be two speakers. One has one vertical half-circle in front of it, resembling the low output volume, and the other has multiple half-circles in front of it, resembling the opposite.

E.3.5 Accept or Decline Suggestions

Description

After going through the current loop area four times, the user is presented with a suggestion. The suggestions is visualized and consists of the outlines of the current token shape. The user can ignore the suggestions and thus decline it, or accept it by laying a token in the displayed shape. When doing so, the visualization gets replaced with a filled, token shaped visualization.

Flow of Events

Due to many ambiguities not yet predictable at the moment.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2), change the volume (see Use Case 4) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Software
  - Q: Where is the suggestion displayed?
    A: The suggestion is displayed one beat after the last placed token.
  - Q: How is the suggestion selected?
    A: There are currently two possibilities: The first possibility is based on a Markov model (see Subsection 4.3.3), which selects the most suitable tone based on the previously played tones. The second possibility is an algorithm, which selects a random tone from the pentatonic scale (see Section 4.1.9).
- Feedback
  - Q: When is the suggestion displayed? What technical parameters trigger them?
    A: A suggestion is displayed after the current loop area has been run through four times.
  - Q: How does the suggestion visualization look like?
    A: The suggestion looks like the outlines of the visual representation of the approaches tone token (see Figure 11.8).

  - Q: What is the right amount of suggestions? The user should not feel that the creation process is being taken away from him.
  - Q: Does the suggestion need to be non-intrusive?
  - Q: How can the suggestion be non-intrusive?
    A: The user can ingnore, and therefore decline, the suggestion. Because of this, it is not intrusive.

- Input
  - Q: How can the user accept or decline suggestions?
    A: The user accepts suggestions by laying a token into the suggestion's visual representation.

E.3.6 Adjust Loop Area

Description

The loop area can be set by moving the loop bar tokens. The tokens can only be moved on the x-Axis, as it represents the time, and the loop itself is a time related setting of the system. The end loop bar token can not be moved to the left of the start bar token, as they can not be moved on the y-Axis.

Special Requirements

- System requirement 2.

Flow of Events

Currently not know.

Preconditions

The user approached the installation, put the headphones on and created a melody.

Postconditions

None.

Extension Points

The user can edit the created music (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Input
  - Q: How does the user change the loop area?
    A: There is a second kind of token, with which the user is able to change the loop area.
- Feedback
  - Q: How is the loop area being displayed to the user?
    A: The loop area is being visualized with the help of two vertical bars across the interaction area.

E.4 Pre- Prototyping

E.4.1 Create Melody

Description

The user begins to manipulate sound parameters through a mixture of exploration and intuitive interaction with tangible input tokens. Concrete connections between changes of the user interface and acoustic feedback (output) through the mechanical construction are recognized by the user. Based on this knowledge the user tries to compose simple melodies. When the music is created, the transmitted information to the user is mainly on a visual level.

Flow of Events

Unknown at the current point of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation and put the headphones on.

Postconditions

None.

Extension Points

The user can change the input (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the software components of the system?
  - Q: How do the software components interact with each other?
  - Q: What music creation/ editing philosophy is the system based on?
    A: The system uses the loop approach, where a selected area of the available beats are played in a loop. The loop area can also be changed throughout the users interaction.
  - Q: How many bars/ beats do exist?
    A: There exist sixteen beats in total.
  - Q: How many tones can be played?
    A: The user has the possibility to play 23 different tones (see Section 11.2) on three strings. One tone is always played on one certain string to prevent confusion.
  - Q: How many tones can be played per beat?
    A: The system plays one beat per string, resulting in a maximum of three tones per beat.

  - Q: What happens when multiple tones overlap?

    A: The last token to be placed on a certain beat will be played as long as it lays there. If the token gets removed, the token which has been laid there before the now removed token will be played as long as it gets removed and so on. All inactive token are visualized by a gray rectangle underneath the token, whereas the active token's rectangle are colored.

- Hardware
  - Q: What are the hardware components of the MUI?
  - Q: What hardware does the software run on?
  - Q: What material/ components is the table made of?
  - Q: What height does the table have to be?
- Input
  - Q: How do the tokens look like?
  - Q: How big should one token be?
  - Q: Is there any difference between the tokens? Are there any different token types?

    A: There are currently two types of token. One for the adjustment of the loop area and one acts as a physical metaphor for tones.
  - Q: How can the tokens help the user to understand the current system status?

    A: Token itself can't, but the visual feedback of the token and the first come first serve functionality can
  - Q: What is the musical metaphor of a token (single tone, a tune, a chord)?

    A: One token resembles a single tone. Tokens differ in width, which reflects the tone length.
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?

    A: The token are not limited in any way
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?

    A: The system's behaviour on special input is covered in the form of functional requirements in the further course of this thesis.
- Feedback
  - Q: How can the current system status be displayed comprehensively?

    A: Apart from the mentioned visual feedback, this question will be answered in the following evaluation
  - Q: What colors should be used for which system status/ property?

    A: The strings are represented by colors red, blue and green.
  - Q: How can the user be supported without given them the feeling that the system does not do something they don't want to?

    A: The user can be supported in determining the right set-in of tones when creating a melody. This is achieved by the use of the so-called snapping. Snapping works as follows: As soon as the user puts a token on the interaction area, thy system

recognizes it and responds with visual feedback in form of a colored rectangle which is slightly larger than a token. This rectangle is, as already explained, displayed beneath the token. When the token doesn't move, the center of the rectangle automatically moves horizontally to the nearest beat and stays there. The same applies on the vertical axis but with the nearest pitch. To the user this looks like the rectangle snaps to a position that the system considers valid. When moving the token again, the rectangle is again displayed directly underneath the token.

– Q: What visual feedback should be shown to the user?

  A: The visual feedback consists of every recognized tone on the interaction area, a loop start and loop end bar, a line which moves from the start bar to the end bar (and therefore displays the current position in the melody)

– Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?

  A: No, this would be too expensive and would interfere with non-functional requirement number 11..

E.4.2 Edit Melody

Description

The user edits the previously created music by rearranging the note tokens. Editing is done according to the similar principle of creating music (see Use Case 1).

Flow of Events

Not known at current state of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

E.4.3 Adjust Playback Speed

Description

The user lets the current melody play faster or slower by using the associated input device. By using this device, he/ she is able to let increase or decrease the playback speed until a certain limit. The system reacts accordingly and plays the created melody faster/ slower.

Flow of Events

The current position of the knob gets recognized by the hardware and sent to the software which let's the visual feedback reflect the current playback speed and tells the mechatronical part of the system to pluck the strings accordingly to the set speed.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the minimum and maximum bpm being allowed by the system?
    A: 60 - 200 bpm.
- Hardware
  - Q: Use continuous transition between bpm or stages of bpm?
    A: Continuous transition.
  - Q: If stages are being used, how big are they?
    A: There are no stages used.
- Input
  - Q: Where on the installation should the device to adjust the playback speed be placed?
    A: Somewhere near the user, to easily reach it by hand.

  - Q: Which input device should be used to adjust the playback speed?
    A: A potentiometer.

- Feedback
  - Q: What is the haptic feedback when turning the input device for the playback speed?
  - Q: What icons can be used as a visual metaphor for ⟨fast⟩ and ⟨slow⟩?

E.4.4 Adjust Playback Volume

Description

The user adjusts the volume output via a physical input device on the installation. The adjusted sound volume then gets output via headphones the user wears. When adjusting the volume, haptic and/or visual feedback is returned.

Flow of Events

The current position of the knob (or similar input device) gets read by the hardware and sent to the mechatronical part of the system which in- or decrease the volume accordingly to the set position/value of the knob.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Hardware
  - Q: Which device should be used to output the sound?
    A: The user wears headphones (see Section 11.2).
  - Q: Should there be more than one pair of headphones?
- Input
  - Q: What input device should be chosen to adjust the volume?
  - Q: Where on the installation will the device to adjust the volume be placed?
- Feedback
  - Q: What icons should be used for the visual representation of the sound volume?

E.4.5 Accept or Decline Suggestions

Description

This use case needs to be completely redesigned and therefore, will not be covered in this subsection.

E.4.6 Adjust Loop Area

Description

The so called loop area describes a part on the interaction area which is set by the user. This area indefinitely repeats every active tone in it.

Special Requirements

- System requirement 2.

Flow of Events

Currently not know.

Preconditions

The user approached the installation, put the headphones on and created a melody.

Postconditions

None.

Extension Points

The user can edit the created music (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) or change the accompaniment (see Use Case 7) at any given point of the interaction.

Resulting Questions

- Input
  - Q: How does the user change the loop area?
    A: There is a second kind of token, with which the user is able to change the loop area.
- Feedback
  - Q: How is the loop area being displayed to the user?
    A: The loop area is being visualized with the help of two vertical bars across the interaction area.

E.5 Pre- Second Evaluation

E.5.1 Create Melody

Description

The user begins to manipulate sound parameters through a mixture of exploration and intuitive interaction with tangible input tokens. Concrete connections between changes of the user interface and acoustic feedback (output) through the mechanical construction are recognized by the user. Based on this knowledge the user tries to compose simple melodies. When the music is created, the transmitted information to the user is mainly on a visual level.

Flow of Events

The user puts the note tokens onto the playing field and eventually readjusts or removes them.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., 16., 9., and 10.

Preconditions

The user approached the installation and put the headphones on.

Postconditions

None.

Extension Points

The user can change the input (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the software components of the system?
    A: ReacTIVision framework, Unity3D and the .NET Framework.
  - Q: How do the software components interact with each other?
    A: By using the OSC protocol or via serial communication.
  - Q: What music creation/ editing philosophy is the system based on?
    A: The system uses the loop approach, where a selected area of the available beats are played in a loop. The loop area can also be changed throughout the users interaction.
  - Q: How many bars/ beats do exist?
    A: There exist sixteen beats in total.
  - Q: How many tones can be played?
    A: The user has the possibility to play 23 different tones (see Section 11.2) on three strings. One tone is always played on one certain string to prevent confusion.
  - Q: How many tones can be played per beat?
    A: The system plays one tone per string, resulting in a maximum of three tones per beat.

  - Q: What happens when multiple tones overlap?
    A: The last token to be placed on a certain beat will be played as long as it lays there. If the token gets removed, the token which has been laid there before the now removed token will be played as long as it gets removed and so on. All inactive token are visualized by a gray rectangle underneath the token, whereas the active token's rectangle are colored.

- Hardware
  - Q: What are the hardware components of the MUI?
    A: Table, video projector, camera and a laptop computer.
  - Q: What hardware does the software run on?
    A: Acer Aspire A515-51G standard version model.
  - Q: What material/ components is the table made of?
    A: The table is made of wood, the tokens of PLA.
  - Q: What height does the table have to be?
    A: Table height is 92cm.

- Input
  - Q: How can the user be supported in keeping orientation?
    A: Colored lines, according to the are where manipulation occurs, provide orientation for vertical and horizontal positioning.
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?
    A: The system's behaviour on special input is covered in the form of functional requirements in the further course of this thesis.
  - Q: How do the tokens look like?
    A: There are note tokens with a note symbol on top,
  - Q: How big should one token be?
    A: 45 x 45 x 10 mm. The length is adjusted by 45mm according to the note value.
  - Q: Is there any difference between the tokens? Are there any different token types?
    A: There are currently three types of token. One for the adjustment of the loop area, one acts as a physical metaphor for tones and one for the joker token.
  - Q: How can the tokens help the user to understand the current system status?
    A: Token itself can't, but the visual feedback of the token and the first come first serve functionality can
  - Q: What is the musical metaphor of a token (single tone, a tune, a chord)?
    A: One token resembles a single tone. Tokens differ in width, which reflects the tone length.
  - Q: Are the token limited to a certain set of tones/ tunes/ chords?
    A: The token are not limited in any way
  - Q: How can users use the input not as intended? How can that be prevented? How should the System react?
    A: The system's behaviour on special input is covered in the form of functional requirements in the further course of this thesis.

- Feedback
  - Q: How can the current system status be displayed comprehensively?
    A: Apart from the mentioned visual feedback, this question will be answered in the following evaluation
  - Q: What colors should be used for which system status/ property?
    A: The strings are represented by colors red, blue and green, using a color scheme to support visually impaired people.
  - Q: How can the user be supported without given them the feeling that the system does not do something they don't want to?
    A: The user can be supported in determining the right set-in of tones when creating a melody. This is achieved by the use of the so-called snapping. Snapping works as follows: As soon as the user puts a token on the interaction area, thy system recognizes it and responds with visual feedback in form of a colored rectangle which is slightly larger than a token. This rectangle is, as already explained, displayed beneath the token. When the token doesn't move, the center of the rectangle automatically moves horizontally to the nearest beat and stays there. The same applies on the vertical axis but with the nearest pitch. To the user this looks like the rectangle snaps to a position that the system considers valid. When moving the token again, the rectangle is again displayed directly underneath the token.
  - Q: What visual feedback should be shown to the user?
    A: The visual feedback consists of every recognized tone on the interaction area, a loop start and loop end bar, a line which moves from the start bar to the end bar (and therefore displays the current position in the melody)
  - Q: Is there a way to use direct haptic feedback (e.g. rumbling motors)?
    A: No, this would be too expensive and would interfere with non-functional requirement number 11..

E.5.2 Edit Melody

Description

The user edits the previously created music by rearranging the note tokens. Editing is done according to the similar principle of creating music (see Use Case 1).

Flow of Events

Not known at current state of the project.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can adjust the playback speed (see Use Case 3) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

E.5.3 Adjust Playback Speed

Description

The user lets the current melody play faster or slower by using the associated input device. By using this device, he/ she is able to let increase or decrease the playback speed until a certain limit. The system reacts accordingly and plays the created melody faster/ slower.

Flow of Events

The current position of the knob gets recognized by the hardware and sent to the software which let's the visual feedback reflect the current playback speed and tells the mechatronical part of the system to pluck the strings accordingly to the set speed.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and volume (see Use Case 4) at any given point of the interaction.

Resulting Questions

- Software
  - Q: What are the minimum and maximum bpm being allowed by the system?
    A: 60 - 200 bpm.
- Hardware
  - Q: Use continuous transition between bpm or stages of bpm?
    A: Continuous transition.

  - Q: If stages are being used, how big are they?
    A: There are no stages used.

- Input
  - Q: Where on the installation should the device to adjust the playback speed be placed?
    A: Somewhere near the user, to easily reach it by hand.
  - Q: Which input device should be used to adjust the playback speed?
    A: A potentiometer.

- Feedback
  - Q: What is the haptic feedback when turning the input device for the playback speed?
  - Q: What icons can be used as a visual metaphor for (fast) and (slow)?

E.5.4 Adjust Playback Volume

Description

The user adjusts the volume output via a physical input device on the installation. The adjusted sound volume then gets output via headphones the user wears. When adjusting the volume, haptic and/or visual feedback is returned.

Flow of Events

The current position of the knob (or similar input device) gets read by the hardware and sent to the mechatronical part of the system which in- or decrease the volume accordingly to the set position/value of the knob.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

- Hardware

  - Q: Which device should be used to output the sound?
    A: The user wears headphones (see Section 11.2).
  - Q: Should there be more than one pair of headphones?

- Input
  - Q: What input device should be chosen to adjust the volume?
  - Q: Where on the installation will the device to adjust the volume be placed?

- Feedback
  - Q: What icons should be used for the visual representation of the sound volume?

E.5.5 Adjust Loop Area

Description

The so called loop area describes a part on the interaction area which is set by the user. This area indefinitely repeats every active tone in it.

Special Requirements

- System requirement 2.

Flow of Events

User rearranges one or both loop tokens. This causes corresponding visualization changes in darkening the areas that are no longer within the loop area.

Preconditions

The user approached the installation, put the headphones on and created a melody.

Postconditions

None.

Extension Points

The user can edit the created music (see Use Case 2), adjust the playback speed (see Use Case 3) and volume (see Use Case 4) or change the accompaniment (see Use Case 7) at any given point of the interaction.

Resulting Questions

- Input
  - Q: How does the user change the loop area?
    A: There is a second kind of token, with which the user is able to change the loop area. The tokens are embedded in a wooden frame, allowing horizontal movement only.

- Feedback
  - Q: How is the loop area being displayed to the user?
    A: The loop area is being visualized with the help of two vertical bars across the interaction area.

E.5.6 Use Joker

Description

During the creation of the melody, a joker token can be set by the user. Based on a pentatonic scale a note is chosen and a joker icon is displayed at the according pitch position. Manipulations according tone height do not cause a new suggestion, except for also changing the beat position.

Flow of Events

The joker token is placed on the table, recognized via id by the system and a joker symbol gets displayed.

Special Requirements

- Business requirement 1.
- System requirement 2.
- Non-functional requirements 1., 2., 3., 4., and 16.

Preconditions

The user approached the installation, put the headphones on and created music (see Use Case 1).

Postconditions

None.

Extension Points

The user can edit the created melody (see Use Case 2), change the volume (see Use Case 4) and speed (see Use Case 3) at any given point of the interaction.

Resulting Questions

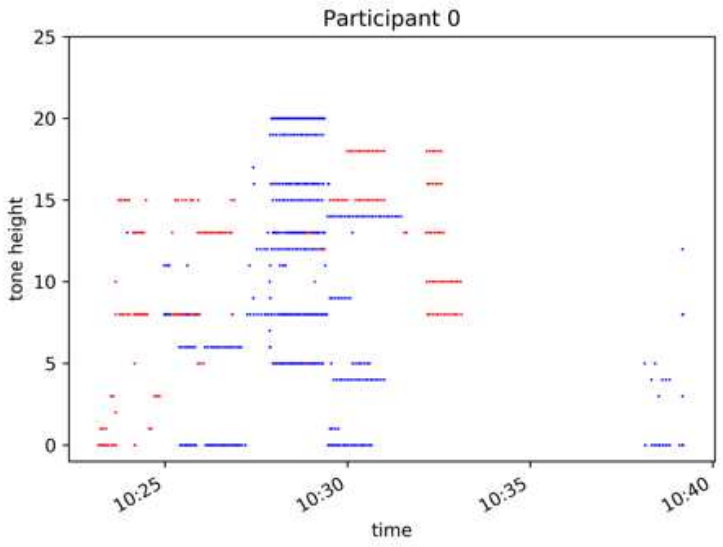
- Software
  - Q: Where is the suggestion displayed?
    A: The suggestion will not be displayed automatically, but a joker symbol appears at the beat, the joker token was placed.
- Feedback
  - Q: When is the suggestion displayed? What technical parameters trigger them?
    A: There will not be a suggestion, the decision is carried out to the user.
  - Q: How does the suggestion visualization look like?
    A: Like a joker symbol.
  - Q: What is the right amount of suggestions?
    A: The user should not feel that the creation process is being taken away from him.
  - Q: Does the suggestion need to be non-intrusive?
    A: Yes.

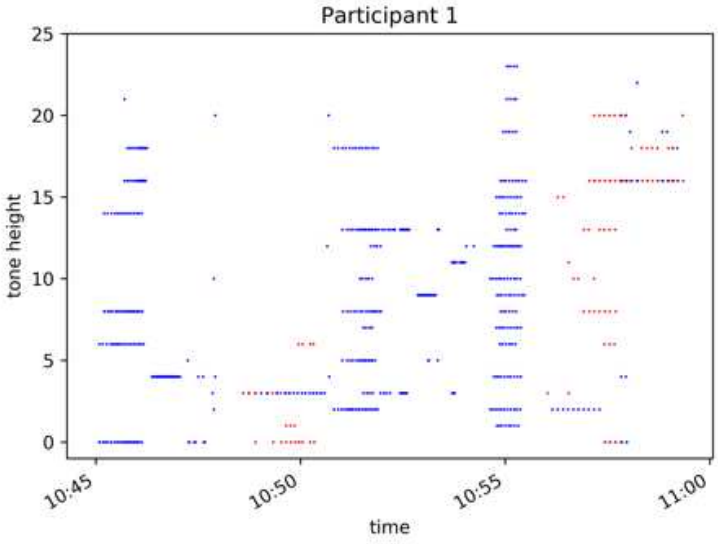
  - Q: How can the suggestion be non-intrusive?
    A: By carrying out the choice to the user.

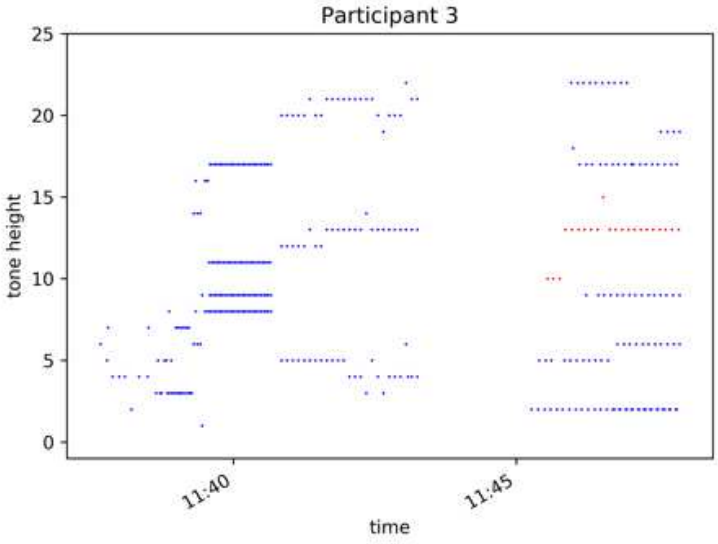
- Input
  - Q: How can the user accept or decline suggestions?
    A: The choice of use is up to the user.

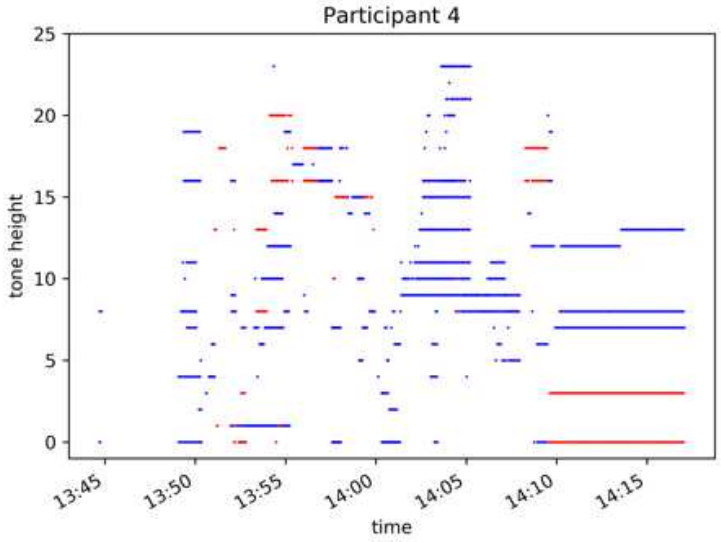
APPENDIX F

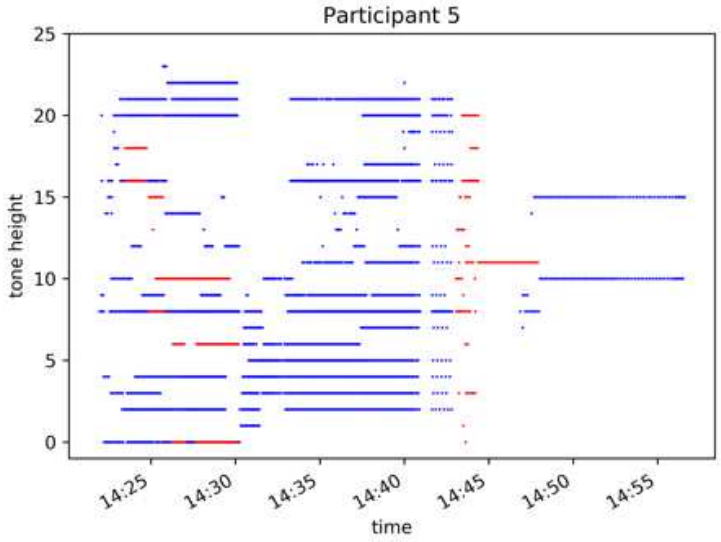
Scatter Plots

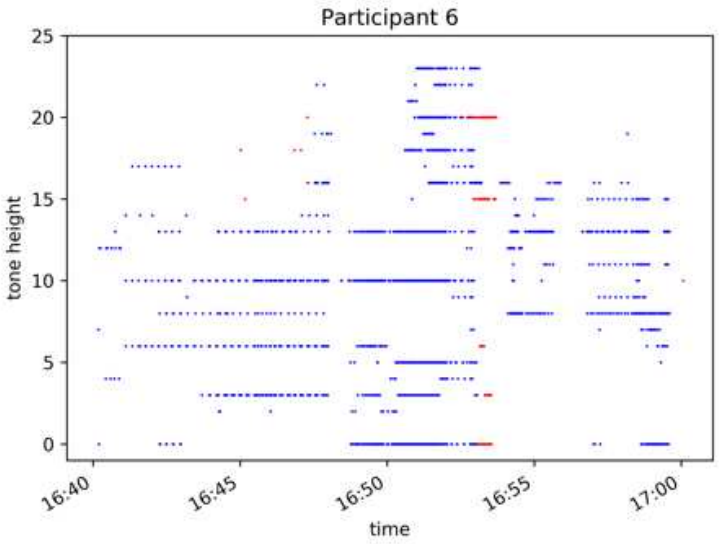


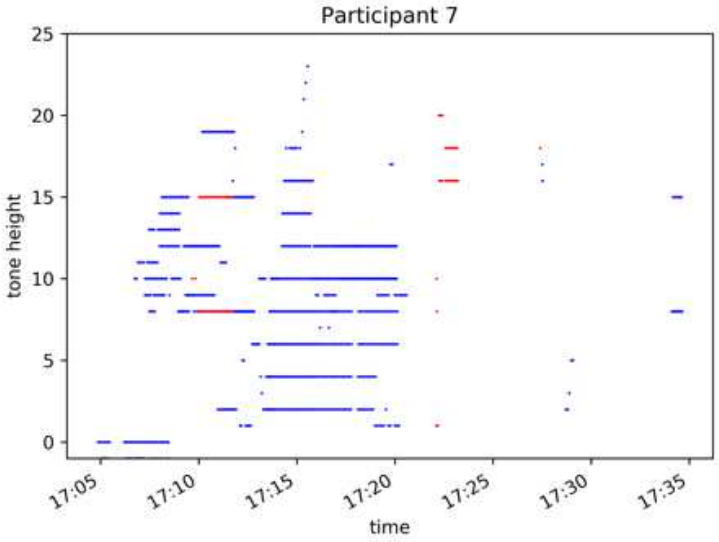


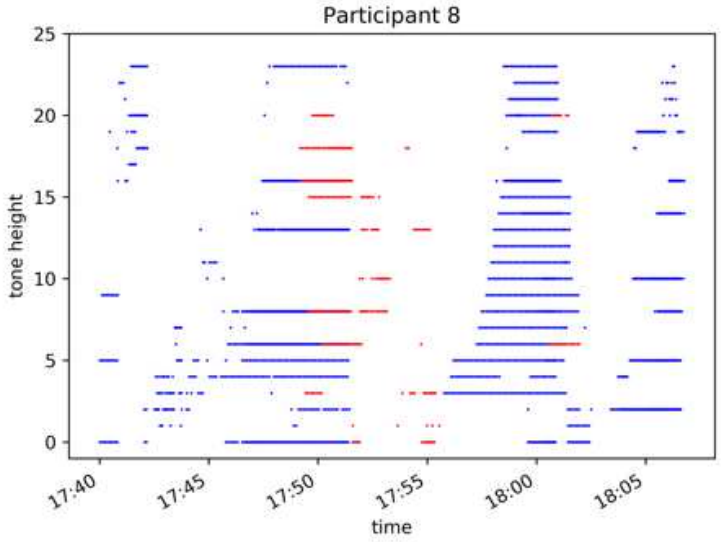


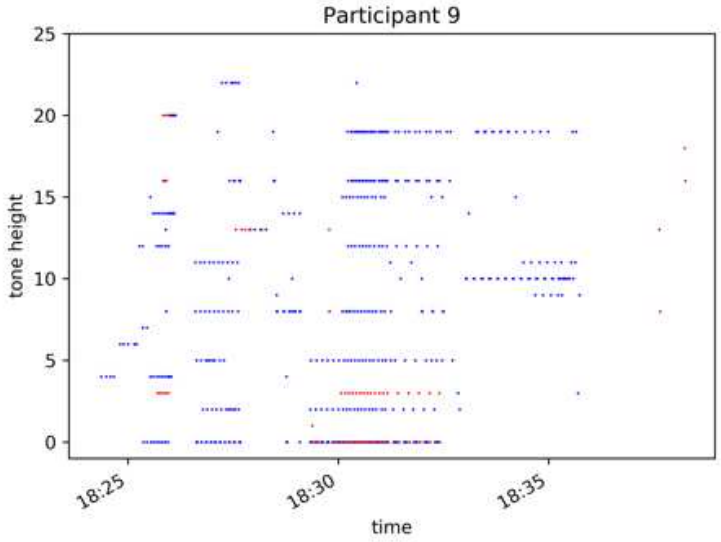


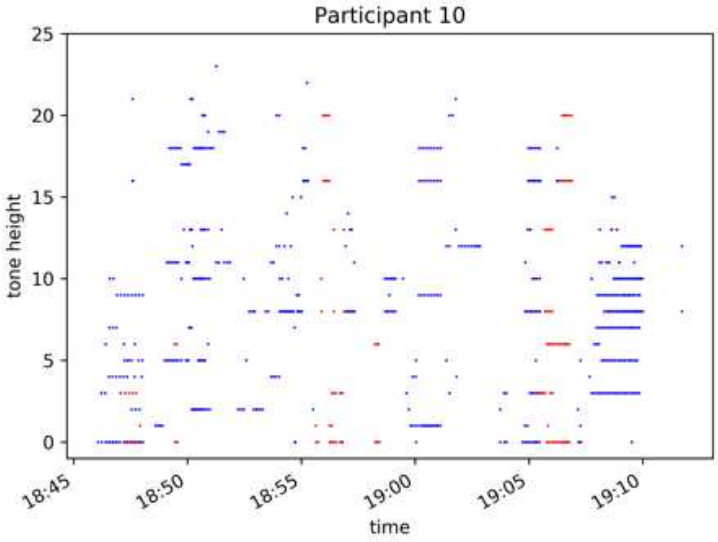


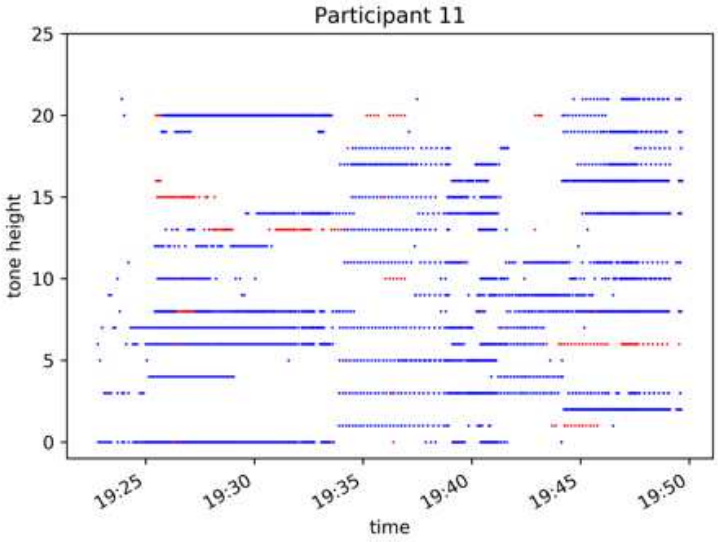




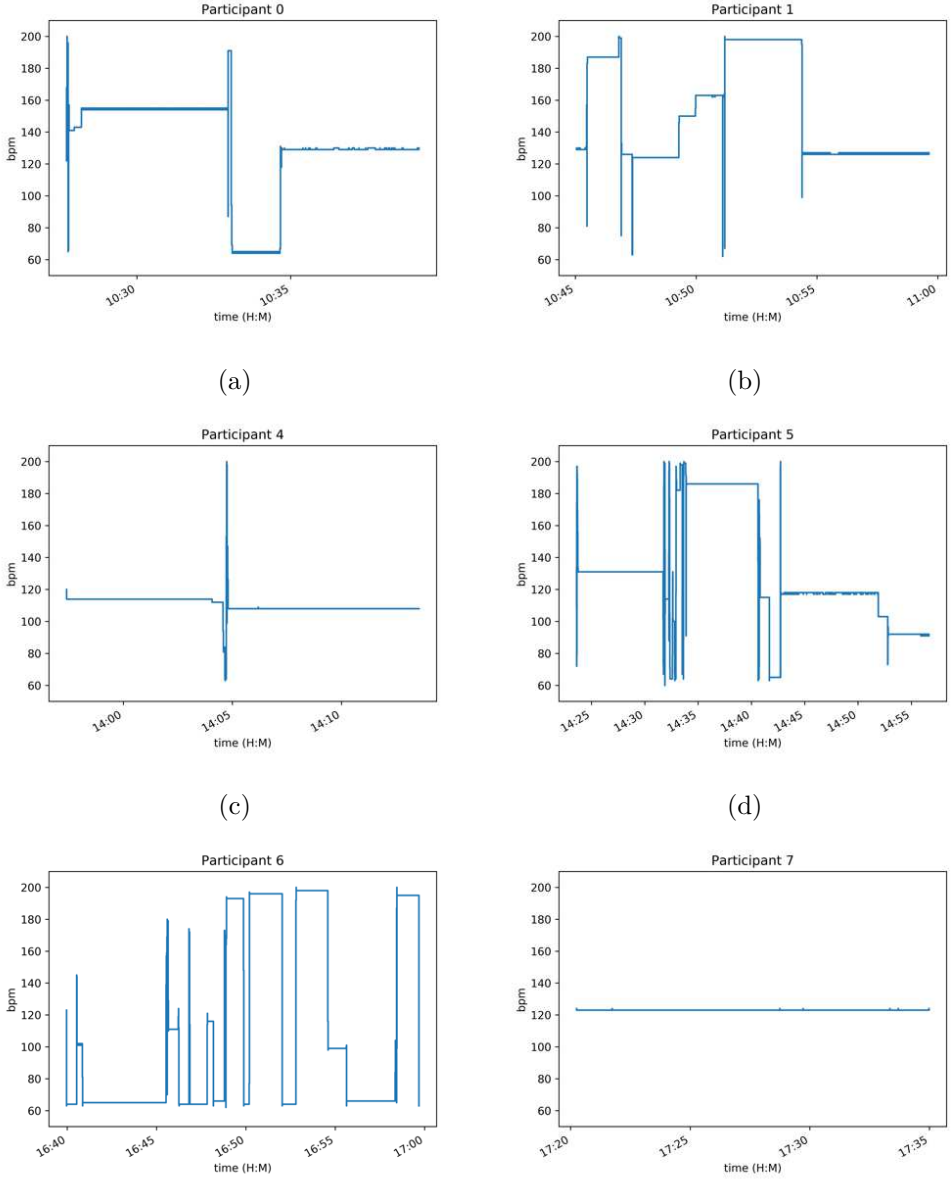








F.0.1 Bpm Values

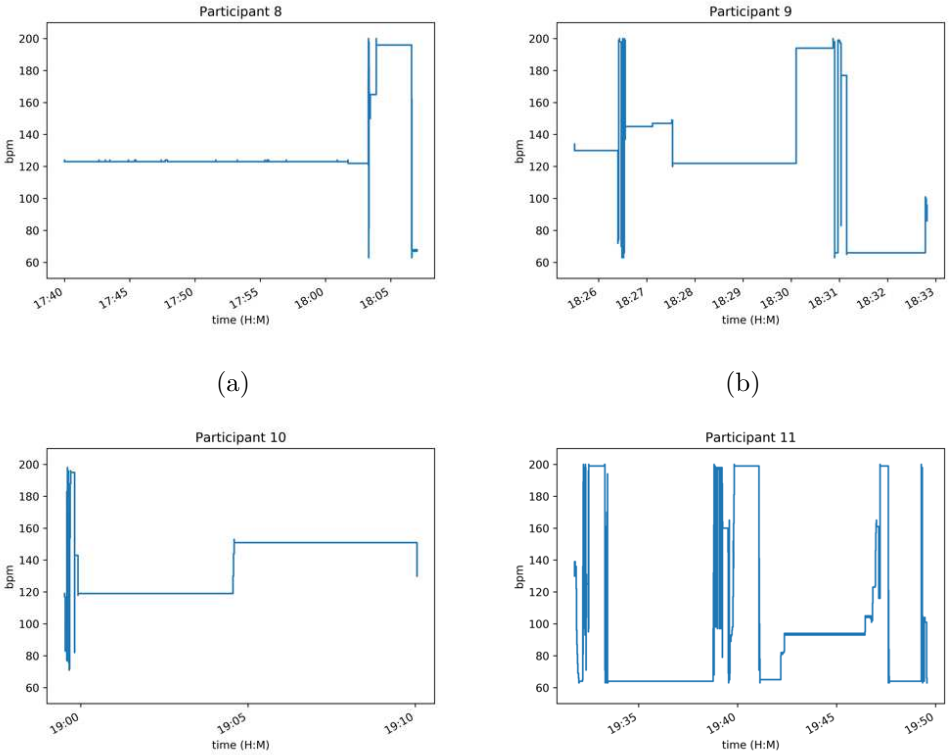


(a) (b)

(c) (d)

(e) (f)

Figure F.1: Bpm values participants 0 - 7



(a) (b)

(c) (d)

Figure F.2: Bpm values participants 8 - 11