

Enabling tangible music and sound composition

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Markus Dietrich

Matrikelnummer 0927782

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Wien, 02.12.2013

(Unterschrift Verfasser)

(Unterschrift Betreuung)

Enabling tangible music and sound composition

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Markus Dietrich

Registration Number 0927782

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Vienna, 02.12.2013

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Markus Dietrich
Leipziger Strasse 58/17, 1200 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.

(Ort, Datum)

(Unterschrift Verfasser)

Danksagung

Im Folgenden möchte ich mich bei allen bedanken die mich bei der Erstellung dieser Arbeit unterstützt haben.

Zuerst gilt mein Dank Herrn Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber für die ausgezeichnete Betreuung meiner Diplomarbeit. Auf die Unterstützung durch seine fachliche Kompetenz konnte ich immer zählen.

Mein ganz besonderer Dank gilt auch meinen Eltern für die liebevolle Unterstützung während des gesamten Studiums.

Weiters möchte ich mich bei meinem Bruder bedanken, der stets ein offenes Ohr für mich und meine Arbeit hatte. Die vielen Diskussionen mit ihm haben diese Arbeit sehr bereichert.

Abschließend möchte ich mich bei allen meinen Freunden bedanken die mich während der letzten Jahre des Studiums stets begleitet und unterstützt haben.

Abstract

Throughout the last decades the Graphical User Interface (GUI) has become the main type of user interface. GUIs present digital information in the form of painted bits on a display and therefore are restricted to very limited communication channels. Tangible User Interfaces (TUI) describe a new method of human-computer interaction (HCI) and try to extend interaction beyond the traditional approach of a graphical user interface, consisting of a rectangular display, a mouse and a keyboard. The focus lies on the development of new ways of interacting with digital information directly over the physical environment and real world features. In recent years a growing number of work dealing with tangible user interfaces has focused on learning. Designers are increasingly inventing tangible interfaces for educational purposes, based on the assumption that hands-on activities on physical objects or the manipulation of such can be of educational benefit. This thesis focuses on tangible musical interfaces, a term used to describe user interfaces which directly interact with music over physical objects. We present a prototype for tangible music and sound composition for educational and artistic purposes. Based on an extensive literature review application scenarios are evaluated, design requirements defined and sketches created. During that design process we examine how data objects containing information about music and sound can be realized with tangibles and how music and sound can be generated with such. We further discuss how tangible composition can be modeled especially for children. Besides designing simple tangible objects for children, also more sophisticated objects are created for older users. Further an evaluation of the implemented functional prototype is performed and possibilities and limitations of tangible music and sound composition are discussed.

Kurzfassung

Über die letzten Jahrzehnte hinweg wurden grafische Benutzeroberflächen zur weitverbreitetsten Benutzerschnittstelle. In grafischen Benutzeroberflächen werden digitale Daten auf einem Bildschirm dargestellt. Die Interaktion findet meist über eine Maus und eine Tastatur statt. Angreifbare Benutzerschnittstellen beschreiben eine neue Methode der Mensch-Computer-Interaktion. Der Fokus liegt hier bei der Entwicklung von neuen Interaktionsmöglichkeiten mit digitalen Daten über die physische Umgebung. In den letzten Jahren haben sich vermehrt wissenschaftliche Arbeiten mit angreifbaren Benutzeroberflächen im Zusammenhang mit Lernen beschäftigt. Viele Projekte stellen mittlerweile angreifbare Benutzeroberflächen speziell zu Bildungszwecken vor, basierend auf der Annahme, dass spielerische Aktivitäten mit physischen Objekten pädagogisch wertvoll sein können. Diese Arbeit präsentiert eine angreifbare Benutzeroberfläche zur Musikkomposition und Tonerzeugung für künstlerische und Bildungszwecke. Aufbauend auf einem umfangreichen Literaturüberblick werden Anwendungsszenarien entwickelt, Anforderungen an das Design definiert und Skizzen erstellt. In diesem Zusammenhang wird untersucht wie sich Datenobjekte die Informationen über Musik und Töne enthalten mit angreifbaren Objekten realisieren lassen und wie Musik und Ton mit solchen erzeugt werden kann. Des Weiteren werden Möglichkeiten zur Musikkomposition und Tonerzeugung speziell für Kinder vorgestellt. Darüber hinaus wird der fertige Prototyp auf seine Funktionalität getestet und Möglichkeiten und Grenzen eines solchen Systems diskutiert.

Contents

List of Figures	vii
1 Introduction	1
1.1 Motivation	1
1.2 Contribution	2
1.3 Organization	2
2 Tangible User Interfaces	4
2.1 History	4
2.1.1 Early stages of Human-Computer Interaction (HCI)	4
2.1.2 The evolution of Tangible User Interfaces (TUI)	5
2.2 Tangible technologies	8
2.2.1 RFID	8
2.2.2 Computer Vision	9
2.2.3 Microcontrollers and Sensors	9
2.3 Tangible Interfaces in the music domain	10
2.3.1 reacTable	10
2.3.2 Audiopad	11
2.3.3 AudioCubes	14
2.3.4 Instant City	15
2.4 Tangible Interfaces and Learning	16
2.4.1 An analytic framework on tangibles for learning	17
2.4.2 Projects	21

3	Methodology	25
3.1	Offline prototyping	25
3.1.1	Sketches	26
3.1.2	Storyboard	27
3.1.3	Mock-up	28
3.1.4	Wizard of Oz	29
3.2	Online prototyping	30
4	Implementation	31
4.1	Design	31
4.1.1	Design requirements	31
4.1.2	Scenarios	33
4.1.3	Sketches	35
4.2	Technologies	39
4.2.1	Scratch	39
4.2.2	LilyPond	44
4.2.3	PureData	46
4.2.4	RFID technology	47
4.2.5	PicoBoard	47
4.3	Prototype	48
5	Evaluation	64
5.1	Functionality of the prototype	64
5.2	Possibilities and limitations	70
6	Conclusion and Future Work	72
6.1	Future work	73
A	Music notation	74
	Bibliography	77

List of Figures

2.1	The Star interface laid the foundation for the modern GUI [18]	5
2.2	The marble answering machine invented by Bishop [17]	6
2.3	A graspable object consisting of a brick and a virtual object [8]	7
2.4	Moving and rotating the brick directly changes the virtual objects [8]	7
2.5	Moving and rotating both bricks simultaneously transforms the object [8] .	7
2.6	The architecture of a reacTable [19]	11
2.7	Overview of reacTable objects [19]	12
2.8	The reacTable surface with tangible objects [19]	12
2.9	Audiopad system in action [32]	13
2.10	Selection of a sample from the tree with the selector puck [32]	14
2.11	A set of AudioCubes [34]	14
2.12	Instant city with multiple players [15]	16
2.13	Analytic framework on tangibles for learning [26]	17
2.14	Wooden blocks shaped like jigsaw puzzle pieces [13]	22
3.1	Sketches of PDA agenda screens [5]	26
3.2	Sequential storyboard [10]	27
3.3	Branching storyboard [10]	28
3.4	Narrative storyboard [10]	29
3.5	Example mock-up of a hand-held display [4]	30
4.1	Music building block	35
4.2	Multiple music building blocks with different notes and note values	36
4.3	Tower building blocks with their respective note below	37

4.4	Multiple tower building blocks of different form and color	38
4.5	Building blocks for sound composition	38
4.6	Toy animals with a tractor	39
4.7	Hello world program in Java [24]	40
4.8	Hello World program in Scratch [24]	40
4.9	Screenshot of Scratch Interface	41
4.10	Scratch sound controls	42
4.11	Simple LilyPond input fragment with output [31]	44
4.12	Graphical objects from the unformatted score [31]	45
4.13	A more complex music sheet produced with LilyPond [31]	46
4.14	PicoBoard with features highlighted [38]	48
4.15	Overview of the prototype system	49
4.16	Overview of all prototype components	56
4.17	Music building blocks	57
4.18	Control and setting building blocks	58
4.19	Tower building blocks of different size	59
4.20	Sound building blocks with control unit	61
4.21	Tractor with RFID reader and animals with matchboxes	62
5.1	Mary had a little lamb building blocks	65
5.2	Mary had a little lamb LilyPond score	65
5.3	Drunken Sailor building blocks	66
5.4	Drunken Sailor LilyPond score	67
5.5	Recorded sound composition	68
5.6	Recorded animal noises	69
A.1	Treble clef with different notes	75
A.2	Notes with different value	75
A.3	Rests with different value	75
A.4	Accidentals and repeats	76

CHAPTER 1

Introduction

This chapter is split into three sections. In the opening part the motivation for this work is outlined. The second part presents the contributions and points out the research questions examined in the course of this thesis. In the third and last part of this chapter the further structure is specified.

1.1 Motivation

Throughout the last decades the Graphical User Interface (GUI) has become the main type of user interface. GUIs present digital information in the form of painted bits on a display and therefore are restricted to very limited communication channels. Tangible User Interfaces (TUI) describe a new method of human-computer interaction (HCI) and try to extend interaction beyond the traditional approach of a graphical user interface, consisting of a rectangular display, a mouse and a keyboard. The focus lies on the development of new ways of interacting with digital information directly over the physical environment and real world features [17]. In recent years a growing number of work dealing with tangible user interfaces has focused on learning. Designers are increasingly inventing tangible interfaces for educational purposes, based on the assumption that hands-on activities on physical objects or the manipulation of such can be of edu-

cational benefit [26]. This thesis focuses on tangible musical interfaces, a term used to describe user interfaces which directly interact with music over physical objects.

1.2 Contribution

In the scope of this thesis a prototype for tangible music and sound composition is developed for educational and artistic purposes. Based on an extensive literature review considering the history of human-computer interaction, the evolution of tangible interfaces, tangible technologies and state-of-the-art tangible musical interfaces, possible application scenarios are evaluated, design requirements defined and sketches created. The outcome of the design and implementation process is a functional prototype system for music and sound composition. The scenarios implemented offer a variety of features and target different people from different age groups. Simple tangible objects for music and sound generation are designed especially for children while more sophisticated objects are created for older users. Thus the prototype with its different application possibilities addresses a wide audience including children, musicians, other artists and everyone interested in tangible music and sound composition. In addition the following questions are examined:

- How can data objects containing information about music and sound be realized with tangibles?
- How can music and sound be generated with tangibles?
- How can tangible music and sound composition be modeled for children?
- What are the possibilities and limitations of tangible music and sound composition?

1.3 Organization

The remainder of this thesis is organized as follows:

- Chapter 2 covers the historical background of tangible user interfaces, discusses tangible technologies, introduces tangible interfaces in the music domain and provides a framework for possible educational benefits of tangible interfaces.
- Chapter 3 discusses the methodology by introducing prototyping as method to design new systems.
- Chapter 4 deals with the design process and implementation of the prototype system.
- Chapter 5 evaluates the prototypes functionality and discusses possibilities and limitations.
- Chapter 6 concludes this thesis and provides an outlook on future work.

Tangible User Interfaces

This chapter is split into four sections. The first section focuses on the history of Human-Computer Interaction (HCI) and the evaluation of Tangible User Interfaces. The second section discusses popular technologies with which TUI projects are realized. The third gives an overview of State-of-the-art developments focusing on Tangible Musical Interfaces. The last section deals with tangible interfaces in the context of learning.

2.1 History

This section discusses the beginnings of Human-Computer-Interaction and how tangible user interfaces have evolved over time.

2.1.1 Early stages of Human-Computer Interaction (HCI)

Human Computer Interaction is a research area that emerged in the early 1980s and combines computer science, behavioral sciences, cognitive sciences and several other related fields of study. It deals with the interaction between people and computers.

Until the late 1970s, only very few experts were interacting with computers. This changed rapidly in the following years when personal computing emerged. Personal computing included both software (e.g. text editors or first computer games) and computer platforms (e.g. operating systems or programming languages) and made every-

body a potential user of computers. It also highlighted the shortcomings of computers considering usability aspects [6].

In 1981, Xerox built the Xerox Star workstation which is considered as the first commercial system demonstrating the use of a Graphical User Interface utilizing a keyboard and a mouse for user interaction the way still popular today [18]. Figure 2.1 shows the Star interface which set important design principles in HCI and the foundation for the modern GUI. A lot of design principles like desktop icons, folders, documents and different windows introduced with the Xerox Star are still present in today's GUIs.

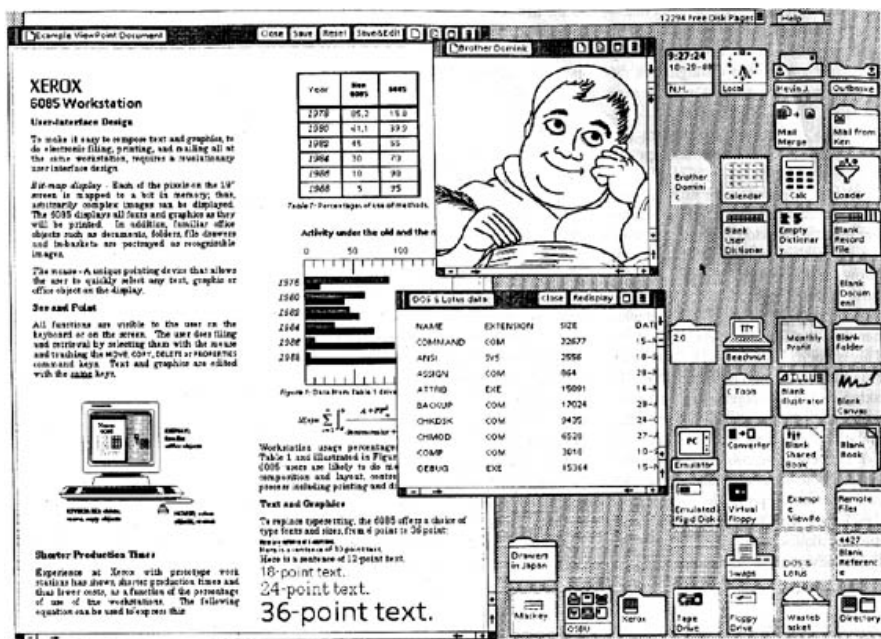


Figure 2.1: The Star interface laid the foundation for the modern GUI [18]

Since that time interactions between people and cyberspace have largely been based on traditional graphical user interfaces (GUI). The interactions with these GUIs are separated from the ordinary physical environment in which we live.

2.1.2 The evolution of Tangible User Interfaces (TUI)

In 1992, Durrell Bishop invented the Marble Answering Machine, a prototype of a telephone answering machine where marbles represent voice messages. Whenever an in-

coming call gets answered by the machine, the voice message is physically implemented in a marble. Figure 2.2 shows the marble answering machine with colored marbles. A voice message can be played by dropping the respective marble on a designated spot on the machine. Moreover a special telephone invented along with the answering machine can dial the number of the caller automatically when the marble is dropped on it [7]. The marble answering machine is one of the first tangible user interfaces using physical objects to represent digital information.

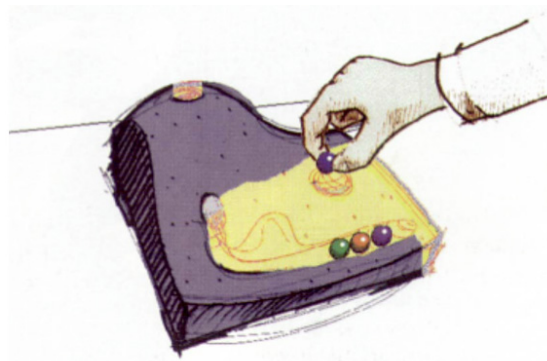


Figure 2.2: The marble answering machine invented by Bishop [17]

In 1995, the concept of Graspable User Interfaces was introduced in [8]. In a traditional graphical user interface a display serves as graphical interface with elements such as icons, menus and windows. These graphical elements exist in a completely virtual form and input devices like a keyboard or mouse are used for interaction with these elements. Thus, input devices and virtual graphical elements are needed for manipulating electronic objects. In graspable user interfaces however physical artifacts are used for directly manipulating electronic objects. These physical artifacts present new input devices and are more tightly coupled to the virtual objects they are interacting with.

Figure 2.3 shows the concept of bricks which are physical artifacts attached to and interacting with virtual objects on a horizontal computer display surface. The brick and the virtual object combined represent a graspable object. Figure 2.4 shows a simple example of how moving and rotating of the virtual object is done by changing the location and direction of the respective physical artifact (i.e., the brick). In Figure 2.5 two bricks are placed on the display and rotated at the same time transforming the virtual object

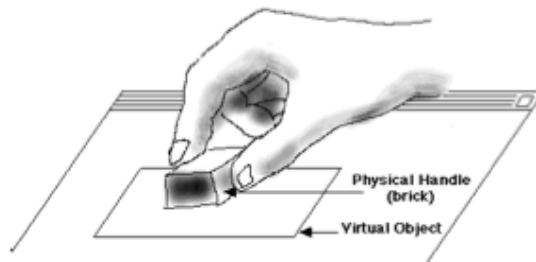


Figure 2.3: A graspable object consisting of a brick and a virtual object [8]

from a rectangular shape to a bended shape. It should be evident by now that bricks offer a wider variety of possible modifications to the virtual object than any pointing device such as a mouse.



Figure 2.4: Moving and rotating the brick directly changes the virtual objects [8]

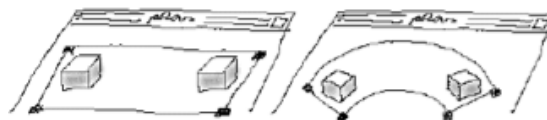


Figure 2.5: Moving and rotating both bricks simultaneously transforms the object [8]

In 1997, based on the concept of graspable user interfaces introduced in [8], Hiroshi Ishii and Brygg Ullmer at the MIT presented their vision of “Tangible Bits” in [17]. Tangible Bits should allow users to “grasp and manipulate” bits by coupling these bits with physical artifacts in the real world. The goal was to overcome the gap between virtual objects and the physical environment in which we live and show concrete ways beyond the dominant model of Graphical User Interfaces usually operating with a flat rectangular display, a mouse and a keyboard. The ultimate goal of Tangible Bits is to turn the whole physical environment into interfaces between people and digital information.

In 1999 researchers at the MIT Tangible Media Group presented an urban planning system called URP [46]. The system is capable of interactively simulating the effects

building placement has on sunlight and wind flow. The tangible representations of buildings generate a digital shadow that is projected onto the surface. Different tangible tools let the user control the system. It is possible to experiment with wind speed, changing material properties of buildings or changing the time of day for different sunlight effects. The changes made directly affect the projected shadows and wind simulation.

Since then lots of related systems have been developed and the concept of tangible user interfaces was taken up by many other research groups around the globe. The next section discusses popular technologies used for developing tangible user interfaces.

2.2 Tangible technologies

A wide range of technologies is used by developers for implementing TUI projects as there is no single standard device. Desired functionalities are among others, object and gesture detection and the sensing of changes in the physical world. In order to select the right technologies for a project one has to consider which properties, including physical properties sensed, cost, performance, aesthetics, robustness, reliability and scalability, fit best to their purpose. Below some popular technologies are introduced [37].

2.2.1 RFID

Radio-frequency identification (RFID) enables the identification of tagged objects using wireless radio-based technology. An RFID reader is used to sense the presence and identity of a tag whenever the tag is within the readers range. Tags can be placed on any object, making it easy to keep track of them. An RFID system usually consists of at least a reader and a tag. There are active and passive RFID tags, both containing a transponder with a circuit to store and process information as well as an antenna for signal transmission. Active tags contain a battery and thus are able to initiate signal transmission themselves while passive tags do not carry a battery and need an external source (i.e. a reader) to start transmission. RFID-based TUI projects usually make use of cheap passive tags. Communication between a reader and a tag only happens when they are within range of each other. Due to the fact that larger antennas, which are able to cover a bigger distance, are a lot more expensive, detection is mostly limited to short

distances. Some RFID readers are able to detect multiple tags at the same time or even to write small amounts of data to tags. Others can only detect one single tag at a time and are read-only. Whenever a tag is detected by the RFID reader, the reader sends the ID of the respective tag to the TUI application which can then interpret the ID and react accordingly [37]. Typical operating frequencies are 128 KHz, 13.6 MHz, 915 MHz, or 2.45 GHz for passive tags and 455 MHz, 2.45 GHz, or 5.8 GHz for active tags [48].

2.2.2 Computer Vision

In TUI projects computer vision is mostly used for applications where users are interacting with objects on any type of surface. This technology can sense the position of objects on a two dimensional surface (e.g. a table) in real time. Additional features like shape, size or orientation of the objects can also be detected. Computer vision systems are mostly either of the artificial intelligence type or tag type. Artificial intelligence applications apply algorithms to automatically interpret a picture while applications of the tag variety track special markers which are attached to the physical objects used. These markers allow accurate calculations of positions and rotations of objects on a two dimensional surface. Due to the fact that specially developed computer algorithms are optimized for detecting such markers, the tag-based approach is considered to be more precise, more robust and also requires less processing power than AI systems. Computer vision systems used in TUI projects usually consist of a camera for the data input, a projector for providing a visual output, and a software package for the actual computer vision part [37].

2.2.3 Microcontrollers and Sensors

Microcontrollers are small computers that serve as a bridge between the physical world and the computing world. They are simple and mostly rather cheap devices that receive information about the physical environment from sensors. A wide range of different sensors exists to measure physical properties like noise level, light intensity, motion, acceleration, proximity, touch, direction and temperature. Microcontrollers and sensors are popular technologies used for TUI projects. Some of these devices require little

programming skills, others are especially designed for developers with a non-technical background. These microcontrollers can be perfectly used for prototyping and thus reduce the amount of time used for TUI development [37].

A popular example for such a device is the Arduino, a physical computing platform containing open source hardware and software. It consists of a simple I/O board and an easy-to-use development environment implementing the Processing language used especially by artists and designers. Interactive objects can be developed either standalone or connected to other software on a computer like Flash. [2]. Another such device is the PicoBoard which can be used along with the Scratch programming language [37].

2.3 Tangible Interfaces in the music domain

This section presents state-of-the-art developments in the field of tangible user interfaces and especially focuses on tangible musical interfaces, a term used to describe user interfaces which directly interact with music over physical objects.

2.3.1 reacTable

The reacTable is a modern musical instrument described in [20] and is one of the most prominent examples of a tangible musical interface. The surface of the table serves as tangible user interface where physical objects can be placed on. Dependent on the location and orientation of these objects a sound is generated. Figure 2.6 illustrates the architecture of a reacTable with all its components.

The tangible objects are in different shapes and passive which means the objects have no sensors or other hardware integrated. They can be distributed anywhere on the table surface. A video camera located beneath the transparent surface analyzes the current state of the objects on the table. reactIVsion [21], a computer vision framework developed for the fast tracking of physical objects in a real-time video stream, serves as the video engine of the reacTable. Data about all the objects distributed on the table including their respective type, location and orientation is then sent to the connection manager via a protocol called TUIO [22]. The connection manager then calculates the connections between the different physical objects according to their affinities and

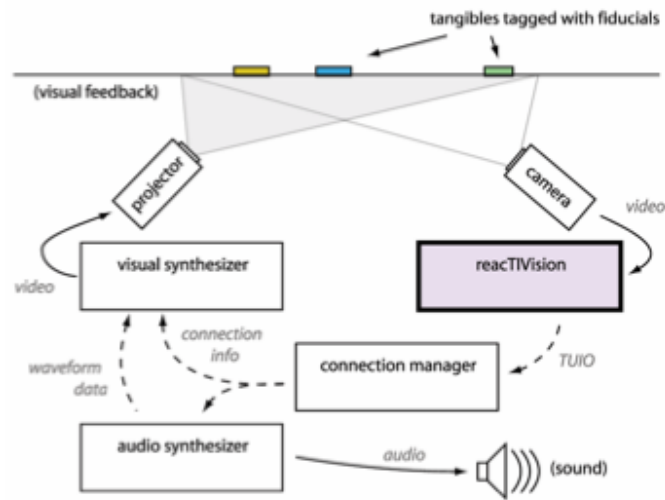


Figure 2.6: The architecture of a reacTable [19]

some trivial proximity rules. Figure 2.7 lists all the different reacTable object types and their respective in- and output connectors [19]. Each type has a specific function like generating sound, filtering audio or controlling sound parameters. The resulting network is then sent to the audio and visual synthesizer. The audio synthesizer takes care of the sound generation. The visual synthesizer sends its data to the projector which draws the objects at their positions including the connections they have with other objects. Moreover the visual synthesizer receives data about the waveforms from the audio synthesizer to correctly draw the audio connections in real time.

Figure 2.8 shows the reacTable while multiple users are interacting with the physical objects. Remote collaborations are also possible for several simultaneous players [23].

2.3.2 Audiopad

Audiopad is a tag-based interface for musical performance presented in [32]. It is like a tabletop DJ system utilizing knobs as controls with the goal to combine the modularity of such controls with the power of multidimensional tracking interfaces. Throughout the last decade most electronic musicians switched to laptop based performances and hereby created a gap between the performer and the audience as stage presence decreased. The Audiopad addresses this issue by visually highlighting the interaction







	Connections	Shape	Examples
Generators	<ul style="list-style-type: none"> ▪ 1 audio out ▪ N control in 		<ul style="list-style-type: none"> ▪ square wave ▪ sampler player
Audio filters	<ul style="list-style-type: none"> ▪ 1 audio in ▪ 1 audio out ▪ N cntrl in 		<ul style="list-style-type: none"> ▪ resonant filter ▪ flanger
Controllers	<ul style="list-style-type: none"> ▪ 1 cntrl out 		<ul style="list-style-type: none"> ▪ sine wave low frequency oscillator ▪ 12-step amplitude sequencer
Control filters	<ul style="list-style-type: none"> ▪ 1 cntrl in ▪ 1 cntrl out 		<ul style="list-style-type: none"> ▪ decimator ▪ sample & hold
Audio mixers	<ul style="list-style-type: none"> ▪ 2 audio in ▪ 1 audio out ▪ N cntrl in 		<ul style="list-style-type: none"> ▪ mixer bus ▪ ring modulator
Global	<ul style="list-style-type: none"> ▪ N cntrl in 		<ul style="list-style-type: none"> ▪ metronome ▪ tonalizer

Figure 2.7: Overview of reacTable objects [19]



Figure 2.8: The reacTable surface with tangible objects [19]

between the user and instrument making it more interesting for the audience to watch.

Objects, called pucks are used as controls and are tracked electromagnetically on a table. Each puck represents a different piece of sound. By moving these objects around on the table the user can switch between different settings and thus dynamically create different sounds, based on the location and orientation of the pucks. Figure 2.9 shows the Audiopad system while a user is interacting with the physical objects.

The pucks contain RF tags which can be tracked on the table by measuring the am-

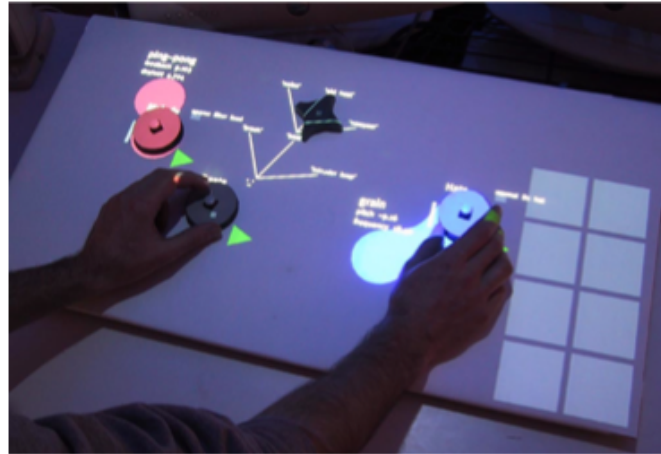


Figure 2.9: Audiopad system in action [32]

plitudes of the resonant frequencies of the tags. Several antennas are used to measure the amplitude of the tags' resonance, obtaining different values to calculate the 2D position accurately to within a few millimeters. Since each tag resonates at a different frequency, the positions of all tags can be determined independently. If two tags are attached to an object, the orientation can be determined as well.

To interact with the Audiopad first a puck has to be mapped to a sample group (e.g. Beats or Melody). After that a user can select the samples of the respective group to be played, modify effects or change the volume. Figure 2.10 shows a puck together with a selector puck navigating through the tree of samples. The tree of samples follows the puck it is associated with when moved, while the selector puck can be used to navigate through the tree and eventually select a node to be played. The volume of a track can be adjusted by rotating the associated puck or simply moving it closer to or further away from a special microphone puck. By pressing the button on top of a puck, information about effect settings are shown which can then be controlled by moving the puck over the desired setting. The graphical feedback on the surface is projected from a video projector mounted above the table on the ceiling [32].

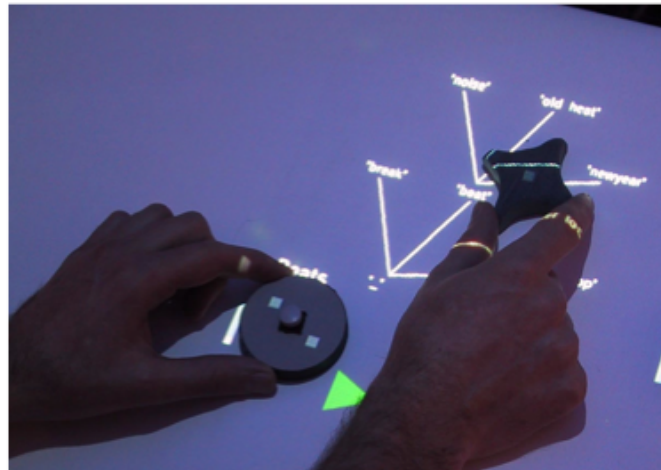


Figure 2.10: Selection of a sample from the tree with the selector puck [32]

2.3.3 AudioCubes

AudioCubes, first introduced in [33], is another tangible musical interface to playfully create dynamically changing sound by interacting with cubes. Figure 2.11 shows several AudioCubes on a table. The cubes consist of some plastic material and include a digital signal processor, optical sensors and emitters for communication with other cubes and are powered by a rechargeable battery. Each cube can be connected to a PC through a cable to download new algorithms for the digital signal processor. A disconnected cube runs the software by itself being powered by the rechargeable battery.



Figure 2.11: A set of AudioCubes [34]

The cubes can measure distances to other objects and locate other cubes and their

respective orientation if they are placed within their interaction range. Within the interaction range, which is based on the range angle, the minimal and maximal ranges, coupling may occur. Coupling describes the process of linking objects to enable a certain functionality. A cube that is placed within the interaction range of another cube is said to be in range, respectively, out of range if placed outside. Coupling takes place when all conditions required for coupling are met. This means that not only two cubes have to be in the interaction range of each other but also that the respective faces are compatible (for instance an emitter and a receiver). If one cube is already coupled to one or more other cubes and, depending on its properties, cannot couple with any additional cubes, it is said to be locked. Therefore also cubes that are placed within the interaction range of a locked cube cannot interact with it.

The cubes communicate by sending and receiving sound using infrared light to and from their neighbors. By placing the cubes on a surface relative to each other a network of sound processors is created and a certain sound is generated. The network changes dynamically whenever the cubes are moved and as a consequence also the generated sound. The cubes are colored and have different functions like filtering, converting sound transmitted optically to a real sound signal, recording sound from the environment and changing settings embedded in other cubes [34].

2.3.4 Instant City

Instant city is a music building game where semitransparent building blocks are placed on a specially designed table to create different sound configurations [11]. Multiple users can experiment with the interface to manipulate sounds at the same time and several compositions are available to choose from at the beginning of each game. Figure 2.12 shows the instant city table with multiple players and building blocks on the table.

The architecture consists of the main table and a spot light which is located above the table. A glass plate serves as the game board on top of the table on which all the semitransparent building blocks are placed on. Below the glass plate multiple light sensors are placed which register the intensity of the flashing spotlight. Every building block that is placed on the table serves as a filter and dims down the intensity of light registered by the sensors. The result is a variation of different levels of gray which



Figure 2.12: Instant city with multiple players [15]

all relate to a specific parameter of the chosen composition. Therefore the final sound output depends on how many building blocks are placed on the table, where they are placed, how high they are and in what order they have been used. Since the hands of the users also produce a shadow on the table and thus affect the resulting levels of gray which the light sensors record, a hand detection is located in the outermost ring of the glass plate. Additional lights are located in that ring below the glass plate and shine as long as hands are in the airspace of the table. The detection of building blocks happens not until all hands are removed from that area and the lights in the outermost ring are turned off. A composition is represented by a special glass object and can be chosen out of a selection of several objects located beside the glass plate by placing it within a specific slot [15].

2.4 Tangible Interfaces and Learning

This section is split into two parts: the first part gives an introduction about learning with tangible interfaces, especially children learning, and provides a framework of possible educational benefits of tangibles. The second section discusses several projects dealing with tangible interfaces and children learning.

2.4.1 An analytic framework on tangibles for learning

Throughout the last two decades tangible interfaces have gained popularity within the field of computing by introducing new ways of interaction with the physical environment and objects within it. Much research in this area has focused on solely technical developments and empirical as well as theoretical work has not managed to keep up with the numerous new technical inventions in this area. One of the areas where a lot of work has been done in the last couple years is learning. Designers are increasingly inventing tangible interfaces for educational purposes, based on the assumption that hands-on activities on physical objects or the manipulation of such can be of educational benefit. Marshall [26] therefore describes six perspectives on learning with tangible interfaces: possible learning benefits, typical learning domains, exploratory and expressive activity, integration of representations, concreteness and sensory directness, and effects of physicality. Figure 2.13 highlights all the key points of the framework.

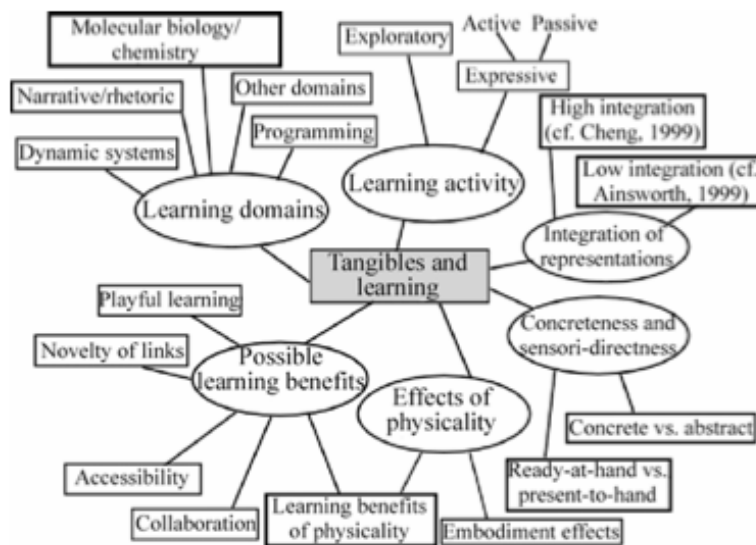


Figure 2.13: Analytic framework on tangibles for learning [26]

Possible learning benefits

There are several reasons why tangible interfaces can bring a benefit to learning. Assuming that perception and cognition are connected with each other, the use of physical

objects for learning activities can have an effect on the way the knowledge is gained compared to activities where knowledge is gained through interacting with virtual materials. Three-dimensional forms, for instance, are likely to be perceived and understood more easily through physical interaction with tangible objects than through visual representation resulting in more effective and more natural learning. Research work also suggests that tangible interfaces can be especially useful for children by engaging them in playful learning. Since tangible interfaces and interaction with them seem to be more natural than other types of interfaces (e.g. graphical interfaces), they also might be easier to interact with. Several projects focusing on design have further suggested that tangible interfaces can be especially applicable for collaborative learning. They can be designed in a way to provide a shared space for collaborative activities and also enable children to watch the things done by other participants. Concurrent interaction by two or more participants is often possible in tangible interfaces, while the typical desktop setup of screen, keyboard and mouse mostly lacks this feature.

Typical learning domains

While tangible interfaces exist in many different forms and a wide range of application areas, some learning domains occur to be especially popular for tangible interface designers. Marshall [26] mentions the following domains among others: narrative (e.g. [39]), programming (e.g. [40]), molecular biology, chemistry (e.g. [9]) and dynamic systems (e.g. [49]). The main feature these domains have in common is their spatial nature. Physical models of molecules are used to support scientists in understanding their three-dimensional structure. Narratives are represented in a spatial format for children to point out the sequence of events, often with flow charts or maps which help structuring a story. In programming physical objects are used to represent program elements which can be put together to create language constructs. Physical components which can be put together are also used to create abstract models of dynamic systems and their behavior. Besides these learning domains which largely focus on the spatiality of the physical materials used, also other physical properties of materials, like their mass, texture or temperature can be of use to support designers in creating tangible interfaces that support learning in all kinds of domains.

Exploratory and expressive activity

For his framework, Marshall [26] adopts the classification made by Mellar [28] who distinguished two types of learning activities when dealing with tangible interfaces: exploratory activities and expressive activities. Exploratory learning uses a representation or certain model of a domain and the user explores that existing representation. The user learns from exploring by comparing the new information to personal experience. The information obtained can confirm, extend or conflict with the user's current level of understanding which may lead to restructuring and cognitive growth. Tangible interfaces can especially be of use for exploratory learning if the process of interacting with physical objects (i.e. tangibles) is really more natural and intuitive for children and other learners than e.g. interaction with graphical interfaces. In that case tangible interfaces can provide an especially useful environment for fast experiments with immediate feedback. Systems could be understood more easily with less cognitive effort and a larger focus on the underlying domain. Expressive learning does not use existing representations but rather lets users create their own representations of a topic based on their personal ideas and understanding. Tools support users to implement their ideas and once finished the own representation can be compared to the real situation. Tangible interfaces can be of use for expressive learning because they enable users to create demonstrative applications almost passively, while focussing on the underlying domain or a different task. Moreover tangible interfaces, due to their novel and creative nature, allow users to create applications which would not be possible in existing media.

Integration of representations

According to Marshall [26] taxonomic frameworks suggest to make a distinction between different types of systems, based on the integration of physical and digital representations. The integration of representations is proposed to be one of the key features of tangible interfaces. Ullmer [45] highlights that tangible interfaces physically embody and integrate representations and controls, compared to graphical interfaces where there is a clear distinction between input devices (i.e. controls) such as a mouse or a keyboard and output devices (i.e. representation) such as a monitor. Marshall [26] critically concludes that taxonomic work on the integration of representations offers little guidance

on how tangible interfaces should be designed to support learning activities and suggests to further look into literature about cognitive science.

Concreteness and sensory directness

As far as concreteness is concerned, both concrete and abstract representations, can be of use for learning activities. Concrete materials can result in a better task performance, abstract materials instead can lead to a better learning transfer. Svendsen [41] highlights that the best and most user-friendly interfaces in terms of simplicity and concreteness, do not guarantee the best performance in problem solving and learning. Marshall [26] further considers what combinations of concreteness might be the most suitable for the physical as well as digital components of tangible interfaces when looking at the learning potential. The probably best choice is the combination of concrete physical representations and abstract digital ones. While this way would clearly be the easiest one for users in terms of working with a new domain, other combinations could encourage users to reflect more as things might get less intuitive. Zuckerman [49] in his work on manipulatives also points out the importance to differentiate between concrete and abstract representations by categorizing manipulatives as either FiMs (Froebel-inspired Manipulatives) or MiMs (Montessori-inspired Manipulatives). Even though both are described as building blocks, FiMs are used for creating concrete physical structures, while MiMs are rather used to create abstract conceptual structures.

Effects of physicality

An increasing amount of cognitive science literature is dealing with embodiment and the connection between physical activity and cognition and the effects this might have on learning. The role of physical objects and manipulatives and its influence on learning has been pointed out also in research about education and psychology. Marshall [26] however stresses, even physical materials are commonly considered especially suitable for learning tasks, the importance of more empirical research to underline that assumption. Further research should particularly investigate in which domains and situations physical objects and its manipulation can be of educational benefit.

2.4.2 Projects

In her work about designing tangible interfaces for children's collaboration Africano [1] presents Ely the Explorer, an interactive educational play system intended for children aged six to seven. The learning tool is designed for the school environment and consists of several tangible tools and software. A tabletop touch screen, RFID tags and readers, several knobs and PDAs serve as user interface. Up to three soft-toys named Ely can be put in an interactive stand, the teleporter, which recognizes the dolls and shows them on the tabletop touch screen. The children then have to solve various tasks using control knobs and the touch screen. After successfully completing tasks postcards represented by RFID tagged cards are printed to provide the children with memories Ely brought home from traveling the world. Putting the postcards into the backpack (i.e. a PDA) each doll is carrying shows the pictures Ely made while traveling.

Evaluating the system showed that it supports both sequential and concurrent interaction and the participating children were highly motivated and engaged in the play. This may lead to the conclusion that such systems can make a positive contribution to the existing number of learning tools. Africano [1] also mentions challenges faced and the demands put on the implementation when prototyping for children. Especially a high level of detail is needed in the implementation process to convey realistic experiences for children since they can not be expected to understand all descriptions or complete missing steps mentally. The author concludes by highlighting the importance of further evaluation and comparison with other teaching methods focusing on collaboration, interaction, engagement and motivation.

Bean [3] introduces the Marble Track Audio Manipulator (MTAM), a construction kit for children in which marbles represent different sound clips and tracks represent sound effects. In order to create musical compositions, the user has to build a tower with different track parts and then drop a marble into the tower. As long as the marble rolls through the tower, a sound clip from a play list is played. In addition each track type has a specific sound effect associated with it which is applied to the sound clip when the marble passes the respective track section. These sound effects are delay, distortion, reverb and low-pass, high-pass filters. Interaction can also take place while a marble is rolling through the tower by stacking a finger into a track to stop or delay a marble.

Additional marbles can also be dropped into the tower to improvise with compositions. The Marble Track Audio Manipulator thus provides children with a simple, creative and playful encounter with music.

In his work about tangible programming Horn [13] mentions Tern, a programming language for elementary school students which consists of several puzzle shaped wooden blocks. Figure 2.14 shows such a collection of wooden blocks. Children can create computer programs by connecting the wooden blocks in a valid order. The design process especially focused on making Tern inexpensive, durable and practical for classroom use. There are different types of wooden blocks which all execute a different action like start, move, turn right, turn left or stop. The program controls robots which live in a virtual grid world on a computer screen. In order to minimize potential syntax errors the wooden blocks were designed in a way that only compatible blocks can be connected with each other. Technically the wooden blocks do not contain any embedded electronics or power supplies, but are recorded and processed by a digital camera and computer vision software which compiles the tangible constructions into digital code. The goal of Tern is to provide children with a simple technology for first programming activities.



Figure 2.14: Wooden blocks shaped like jigsaw puzzle pieces [13]

MusicPets is a tangible musical toy for preschool children. It allows its users to record and store audio data on physical objects represented by soft toys. Each toy contains an RFID tag and therefore can be identified by a reader. Two RFID readers serve as base stations for record and playback. The base station for record features an interface for recording sounds with a microphone and composing simple music with a slider which allows setting the pitch and three buttons for recording, creating chords and inserting pauses. Also the possibility to choose from existing audio files exists. To replay audio data stored on the toys the user only has to place the respective toy on the base station for playback. Evaluation showed that children enjoyed playing with MusicPets and especially liked recording and exchanging messages as well as composing tunes. Besides offering the possibility to compose and record own music and messages, MusicPets also allows children to store that audio data in a physical object. Thus they can physically transport their own recordings and tunes around and show them to friends or parents at home, assuming a playback station is present there [44]. The approach of using RFID technology for generating music is similar to the way music composition is enabled in this work.

The Learning Cube is a simple tangible learning appliance. The cube features several different test based quizzes where the correct answers have to be selected from a set of solutions. The cube consists of two acceleration sensors to track it while moving in a three dimensional space and thus to determine which side of the cube is currently on top. The six sides of the cube embed displays which are used for the questions and answers. In each step a question is shown on the top display and five answers are shown on the remaining sides with one answer being the correct one. The user then has to turn the cube to the side with the right answer on top and shake it. If the side is correct the program continues with the next question, otherwise the user can try again. With this basic functionality the learning cube offers a variety of different applications like a vocabulary trainer where the correct translation of a word has to be determined, a math trainer with basic operations or a letter matching game. The cube form pushes initial engagement when presented to children, since they do not see the learning cube as a traditional learning tool or computer but rather as a toy which leads to increased motivation [42].

Vaucelle [47] introduces Dolltalk, an approach to encourage children in storytelling. The application can record stories of a child and then alter the voice and play back the speech. Dolltalk consists of two dolls and a virtual peer giving instructions to enhance creativity. Whenever both dolls are removed from the platform they are located on the recording starts and the voice of children is recorded. The moment the dolls are returned to their platform the recorded speech is played back but the voice is transformed by shifting the pitch, speeding it up or slowing it down. The inventors of Dolltalk believe that children can discover by themselves what a good story is all about by simply hearing their own stories played back in different voices. The voice modification is used because children are expected to be better critics when a story is told to them by a voice other than their own. Thus by recognizing inconsistencies they can improve their own storytelling abilities.

Methodology

This chapter introduces methods to obtain ideas and design systems which are partly employed in this thesis. It deals with prototyping, an activity which helps developing ideas and designs. Therefore several different methods to obtain a prototype are presented.

According to Beaudouin-Lafon [4], a prototype can be defined as a concrete representation of an interactive system. This representation can either show the whole system or just a part of it. A prototype can be seen as a tangible artifact which helps designers, developers, customers and other stakeholders to get an impression on the final system and thus enables them to reflect upon it. Prototypes are made for a wide range of applications with different purposes and therefore differ in their forms. A couple of fast sketches on paper can serve as a prototype and be useful to designers as well as a detailed computer simulation. Two types of prototypes are distinguished, which are offline and online prototypes.

3.1 Offline prototyping

Offline or paper prototypes are prototypes that can be created without a computer. They are especially useful in the early design stages and support a quick and efficient development of ideas. Compared to online prototypes they are usually cheaper and no specific

knowledge is required to create an offline prototype. Therefore they can be created by almost anybody, not just designers with technical knowledge. Having all the parties including customers, managers and designers involved in a project participating in the prototyping, increases communication and the likelihood that the final design will be accepted by everybody. Typical offline prototypes are paper sketches, storyboards and mock-ups [4].

3.1.1 Sketches

Sketches are a very important tool to develop ideas and create design concepts. According to [43] they are used in lots of different areas like architecture, graphic design, industrial design but also in user interface and interaction design. Sketching offers a quick and cheap way to develop, communicate and evaluate ideas and is therefore more often used in early project phases than more sophisticated alternatives which are also more expensive and take longer to develop. In fact most paper prototypes and even other prototypes begin with simple sketches on paper in the first place before progressing to more detailed prototypes where software tools are often used. This type of sketching, which does not need more than a pencil and paper, is the best way to generate lots of different ideas and thus expand the design space while the final solution is still far from being determined [4]. Figure 3.1 shows some example sketches illustrating a PDA agenda screen [5].

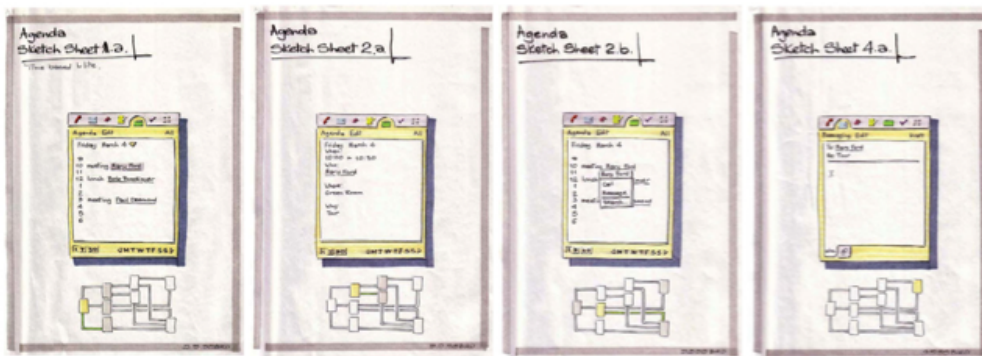


Figure 3.1: Sketches of PDA agenda screens [5]

3.1.2 Storyboard

After the first sketches of an idea have been drawn storyboards can be created. Storyboards are basically collections of several single pictures which show a sequence of events in a process. These pictures can be anything from simple paper sketches to photos or images created with a camera or software tools. [10] distinguishes between sequential, branching and narrative storyboards. A sequential storyboard shows a sequence of frames which highlight the key ideas and interactions within a process. Figure 3.2 illustrates such a storyboard with detailed annotations so even people unfamiliar with the process can understand it.

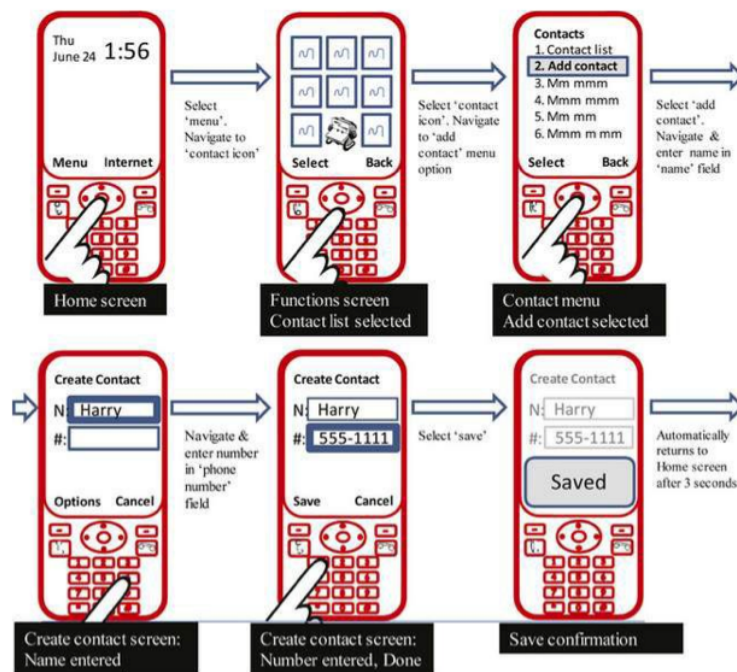


Figure 3.2: Sequential storyboard [10]

A branching storyboard shows decision paths within a process that unfold over time. Figure 3.3 pictures such a branching storyboard where the upper area lists all the possible options for future states and the lower area further describes one of these options.

A narrative storyboard tells a story about the interaction context. It is a sequence of pictures with each picture showing one event of the process. Figure 3.4 shows the step

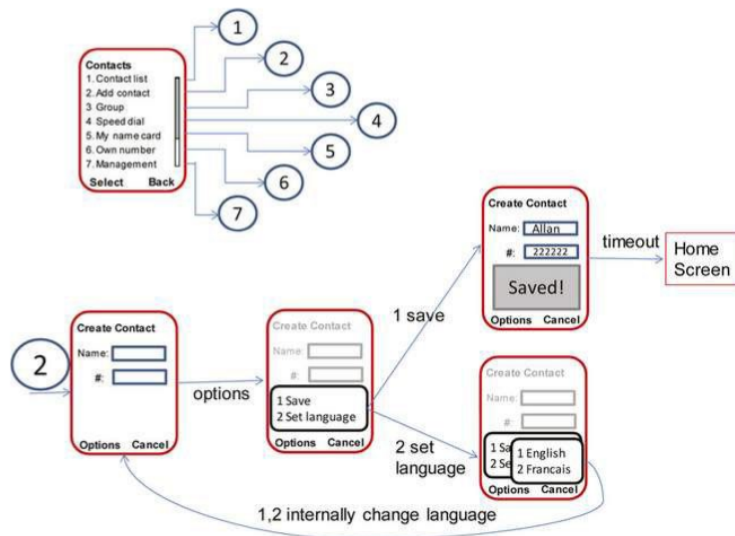


Figure 3.3: Branching storyboard [10]

by step development of a narrative storyboard. The first step is usually the development of a storyline and the drawing of blank frames. After each frame is assigned an event by adding a short description below each frame, sketches showing the respective events are drawn within the blank frames. Finally actions and motions are highlighted for better understanding of the whole process.

3.1.3 Mock-up

Mock-ups are used by designers in the earlier phases of a project to visualize how the final design might look like. They are three-dimensional physical prototypes of a system but do not have any functionality implemented. Their main purpose for designers and other people involved in a project is to gain a deeper understanding of how interactions might take place in the real world and thus collect feedback and generate new ideas for better functionality. Since no functionality is implemented in mock-ups, designers can concentrate on the physical appearance like the positioning of buttons, knobs, screens and other features. Figure 3.5 shows the mock-up of a hand-held display [4].

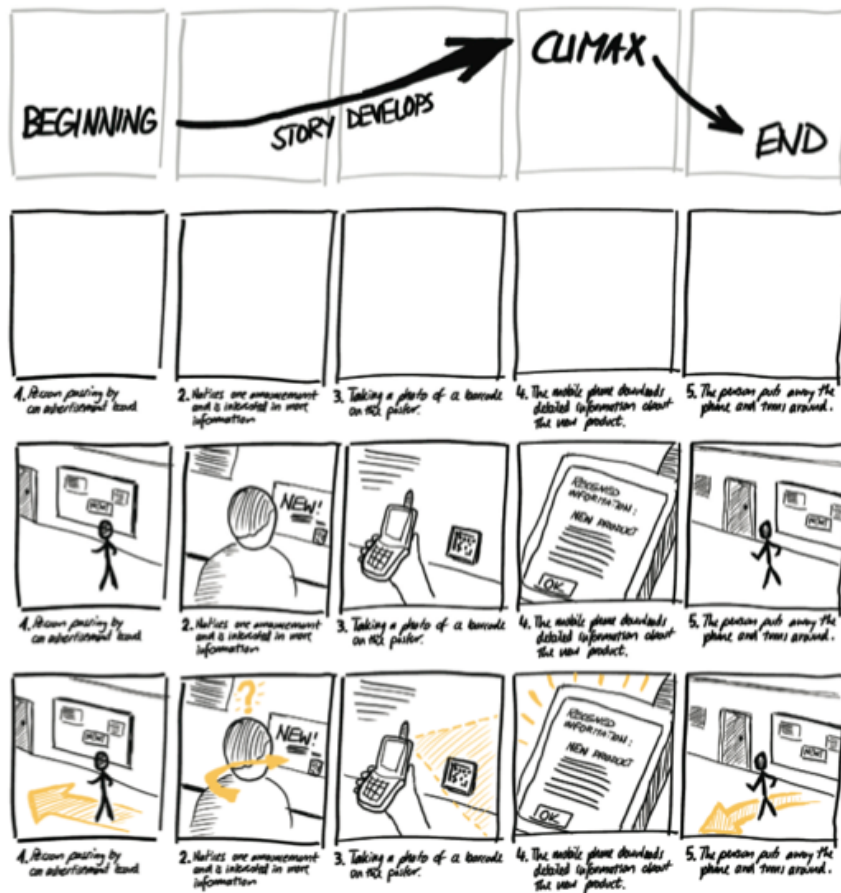


Figure 3.4: Narrative storyboard [10]

3.1.4 Wizard of Oz

Wizard of Oz is another prototyping technique which is already close to digital prototyping. A user interacts with a more detailed sketch or mock-up of an interface while a human 'wizard' simulates all the responses. The interface itself usually does not have any functionality implemented and the intelligence of the system is represented solely by the wizard [27]. According to [4] the technique can also be used with software which is partially functional. The user interacts with a software and whenever a bug or unfinished functionality is encountered, a human developer (i.e. the wizard) who supervises the interaction manually corrects the prototype system.

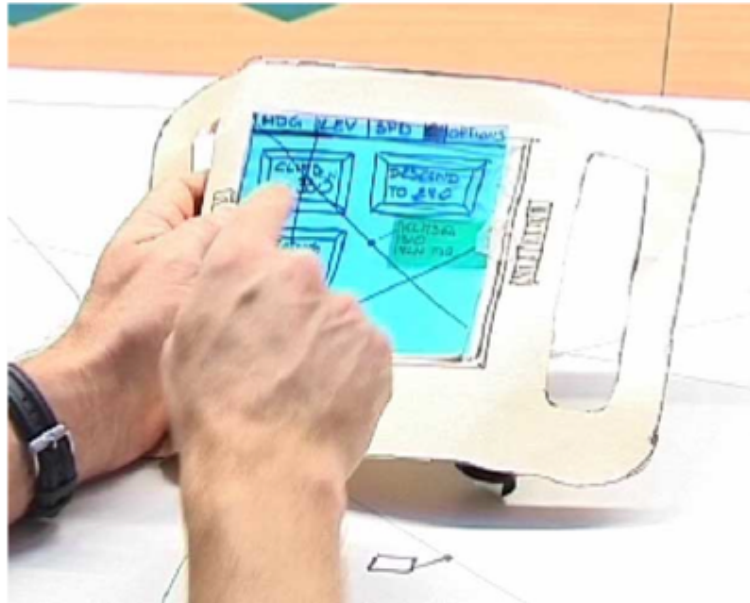


Figure 3.5: Example mock-up of a hand-held display [4]

3.2 Online prototyping

Online or software prototypes are prototypes that require a computer. They are created with software tools and include animations, video presentations as well as programs and applications developed with interface builders and programming languages. One advantage of online prototyping is that prototypes can be created more precisely and thus are considered to be more effective in the later design stages. Compared to offline prototypes they are usually more expensive as they require skilled programmers who can work with the software tools used to create these prototypes [4].

Each prototype highlights some aspects while ignoring others. Therefore designers have to choose carefully which prototyping method they use with regard to the respective purpose and often different methods are used in different design phases, depending on what method is best suited for the current phase.

Implementation

This chapter is split into three sections. The first section covers the design process and outlines the design requirements, describes the application scenarios and shows sketches for the prototype environment. The second section introduces some of the technologies for the prototype and the third and last section presents the functional prototype with all its components and the implemented application scenarios.

4.1 Design

This section deals with the design process and development of the prototype system. In the first part the design requirements are defined. In the next part the application scenarios are clarified and in the last part the sketches for the prototype system are described and illustrated.

4.1.1 Design requirements

The prototype with its different application possibilities shall target artists, researchers and other persons interested in tangible music and sound composition as well as children with cognitive abilities still developing. Therefore the following design requirements have to be considered.

Tangibles

The prototype will be a tangible user interface for music and sound composition and thus consist of tangible objects for interaction. In order to enable meaningful composition a certain number of tangibles is required which preferably should be cheap, small in size and lightweight. The component reading the tangible music and sound building blocks should be fast and precise because many tangibles have to be processed which are mostly located very close to each other.

Usability

The intended users include artists, researchers, but also non-technical people and children without any knowledge of tangible technologies or music and different cognitive abilities. Therefore a simple and intuitive handling has to be enabled and no configuration of the system will be needed before its use. The controls should be mostly self-describing and minimal instructions be necessary prior to usage. Since a large amount of tangibles is used a focus has to be put on an efficient initial setup and configuration strategy.

User Groups

There are primarily two user groups considered when designing the prototype:

- **Children:** In this category are children aged 4-7 whose cognitive abilities are still developing and who most likely cannot read already or just started doing so, but are old enough to recognize simple interrelations. The scenarios for these age groups are designed simpler (e.g. no reading or recognition of symbols is necessary) and focus on basic interaction with music and sound.
- **Others:** In this category are artists, researchers and other people interested in tangible music and sound composition but also older children who are able to read already and recognize different symbols.

Design

In order to make the interaction with the tangibles as pleasurable as possible for the user, the prototype system should be esthetic and visually appealing. Since different application possibilities for different age groups including children will be available in the functional prototype, a focus will be put not only on the previously mentioned usability aspects but also on the visual aspects. Therefore each of the scenarios will be designed for its intended user group.

Correctness

The musical symbols used in the prototype will be in accordance with modern music notation. Thus the application can be used by people from any background because the notation is in effect worldwide. For further information on music notation see Appendix A which lists the most important music notation symbols and explains their meaning.

4.1.2 Scenarios

In this part the prototype scenarios are described. Two scenarios will be designed for the user group *Children*, the other two for the user group *Others*. Both user groups are defined in the design requirements. The functionality of the prototype will cover the following four application scenarios:

Tangible music composition

The first scenario describes the process of tangible music composition. The user is provided with numerous music and control building blocks and one reading component. Each music building block is assigned a visible musical symbol (i.e. notes, rests, special symbols). After placing the different music building blocks in the desired order on a table, the user can move the reading component over the building blocks and hereby create music. Besides composing different music by arranging the music building blocks in a different order, control building blocks exist to record music, play back the resulting piece of music with different settings or transform the recorded music into a score.

Tangible music composition for children

The second scenario introduces an approach to enable tangible music composition especially for children aged 4-7. In this scenario the music building blocks are designed as towers which are all of different size depending on their pitch. These towers are bigger than the building blocks in the first scenario and colored so children can easily handle them and do not get bored that quickly. Also it is not necessary to recognize music symbols or being able to read because a pitch is represented only by the height of a tower. Notes with higher pitches will be represented by taller towers while notes with lower pitches will be found in smaller towers. Children can learn to associate higher towers with higher pitches and train their musical hearing. To assure a certain simplicity for children only one key and one octave will be covered.

Tangible composing with sounds

The third scenario presents a modification of tangible music composition which provides an environment to experiment with different sounds in a less formal manner. In this scenario the user can create different sounds rather than notes by moving the reading component over different sound building blocks. Additionally a sound sequence can be recorded and played back in exactly the same order and time intervals as the original recording. In this scenario users do not have to think about notes and musical theory and can easily experiment with different sounds.

Tangible sound generation for children

This scenario deals with sound generation too and provides a simple environment for children aged 4-7 to experiment with different animal noises. The reading component is attached to a toy tractor and by moving the tractor around on a toy farm different sounds are created when passing by different animals living on the farm. This scenario provides children with an opportunity to playfully experiment with different sounds and helps them to associate a certain animal with its respective noise.

4.1.3 Sketches

In order to evaluate ideas and test different designs, sketches are created. The sketches are drawn with a special focus on the design requirements and application scenarios. Based on these sketches the prototype system will be built. First numerous of simple paper sketches are created and evaluated. The best of these sketches are then selected and modeled on the computer.

The building blocks are the main components of the application. For the music and sound composition part containers are used as building blocks. In order to distinguish between the different building blocks, each block carries an RFID tag with a unique ID in it. Each tag has its own music or sound symbol assigned which is also pictured on the top of the container the tag is in. Figure 4.1 shows one of such building blocks with a note symbol pictured on its top.

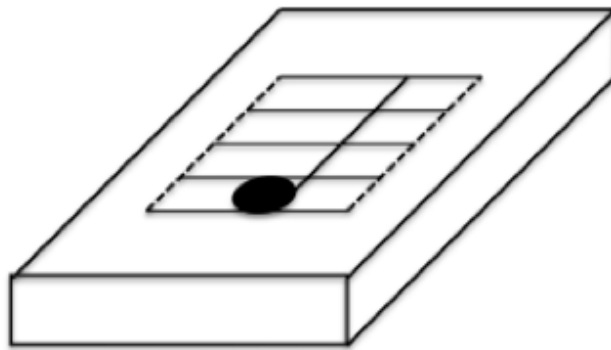


Figure 4.1: Music building block

Figure 4.2 shows multiple music building blocks with different notes and note values assigned to them.

Tangible music composition

The music composition shall work with the most common notes, note values and rests, repeat signs and the accidentals sharp and flat. For further information on music notation see Appendix A which lists the most important music notation symbols and explains their meaning. Each container represents exactly one music symbol. Repeat signs and

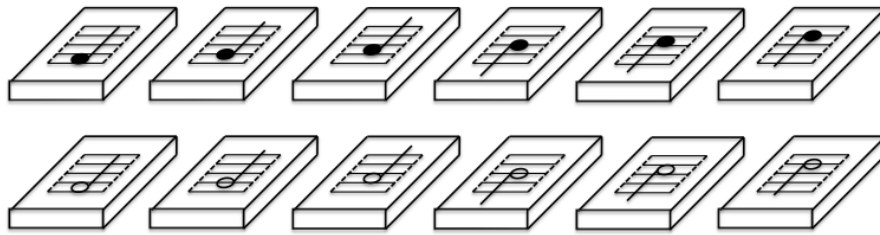


Figure 4.2: Multiple music building blocks with different notes and note values

accidentals always look the same, while rests differ in their value (full, half, quarter, etc.) and notes differ in their value and pitch (i.e. where they are located on the staff). The absolute pitch is defined by the clef used. To simplify matters the only clef used in the music composition is the treble clef.

Since each music building block represents exactly one music symbol it might occur that two building blocks are necessary to define one note (i.e. when accidentals are used). A design alternative would be to leave out the sharp and flat sign as own building blocks and add additional building blocks which have the sharp or flat sign in front of the respective note. This approach is not used in the prototype because a lot more different building blocks would be necessary. Another design alternative would be to define the key beforehand along with the treble which makes the additional sharp and flat signs almost redundant. This approach might be an interesting extension for the future but since complexity would increase it will not be implemented in the current version because according to the design requirements the prototype should be as simple and intuitive as possible. The last design alternative mentioned could be to have own building blocks for the note value which are located before the notes or rests themselves. This would decrease the number of blocks types because for each pitch only one building block type is necessary. This approach is not used because the total number of building blocks would increase a lot because two blocks (i.e. one for pitch and one for duration) are needed to represent one note.

Tangible music composition for children

For this scenario special tower building blocks are used. Figure 4.3 shows how these blocks will look like with the respective note below each tower. The pitch of each tower depends on the height. Lower pitches are represented by smaller towers while higher pitches are represented by higher ones.

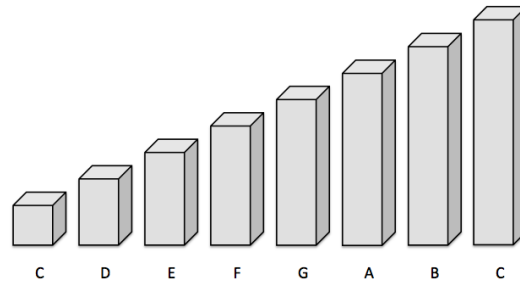


Figure 4.3: Tower building blocks with their respective note below

Moreover the building blocks could vary in form and color. Figure 4.4 shows how such building blocks could look like where the form represents the note value (full note, half note, quarter note, etc.) and the color represents the instrument (piano, guitar, violin, etc.).

Tangible composing with sounds

For composing with sounds, building blocks are used as well but without music symbols assigned to them. Instead these building blocks have their own symbols like a star, cross or circle to distinguish them from each other. Figure 4.5 shows how some of these building blocks could look like.

Tangible sound generation for children

For this scenario toy animals are used instead of the building blocks. An RFID reader is attached to a toy tractor which can be moved around on a toy farm. Whenever the tractor passes an animal the reader identifies its tag and a sound is played which is typical for the respective animal. Figure 4.6 shows how such an environment could look like.

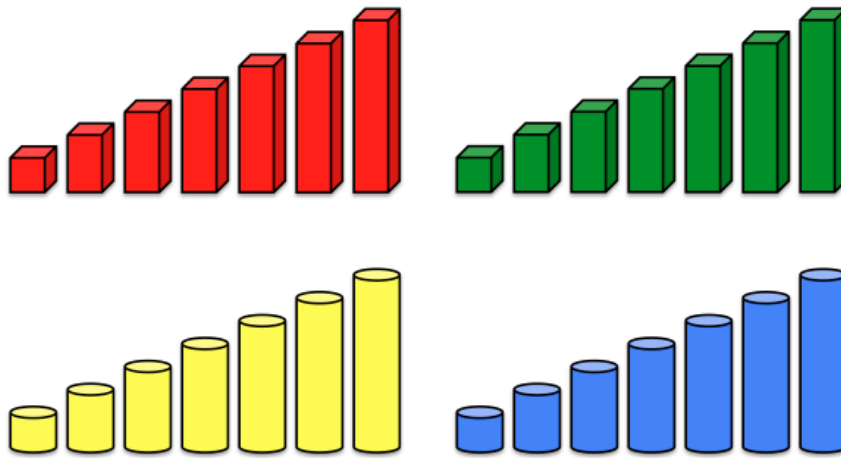


Figure 4.4: Multiple tower building blocks of different form and color



Figure 4.5: Building blocks for sound composition

Additionally for all scenarios control building blocks exist for recording and playback. When music is recorded and played back the note durations and breaks are considered. Thus for the music composition the original recording speed does not matter, only the order in which the music symbols were processed. When playing back a recorded sound sequence the time intervals between the sounds played in the original recording are considered. Therefore the playback sounds exactly the same as the original recording.



Figure 4.6: Toy animals with a tractor

4.2 Technologies

In this section the technologies used for the prototype are presented and technology decisions are explained. On the software side the visual programming environment Scratch, the music notation processor LilyPond and the visual programming language PureData are introduced. On the hardware side RFID technology and a PicoBoard are used.

4.2.1 Scratch

Scratch is a programming environment developed by the Lifelong Kindergarten Group at the MIT Media Lab. It primarily aims to support kids and young students without any programming education to gain first experience and enhance understanding of key programming concepts. Therefore Scratch especially supports programming activities that reflect the interests of youths, like creating games, interactive presentations and animated stories [25].

Most programming languages are hardly understandable for people without any previous training in that field because of their unfamiliar syntax, even in the simplest programs. Figures 4.7 and 4.8 show the difference between a simple Java program, compared to a Scratch command block, both achieving the same outcome: the text “hello, world” is printed on the screen [24].

```

class Hello
{
    public static void main(String [] args)
    {
        System.out.println("hello, world");
    }
}

```

Figure 4.7: Hello world program in Java [24]



Figure 4.8: Hello World program in Scratch [24]

When looking at the figures it becomes obvious that languages like Java challenge students to first master certain programmatic overhead before really starting programming. The learning of the syntax becomes more important in the beginning than solving problems. Additionally people today are accustomed to graphical interfaces much more than to hardly understandable abstract code [24]. Scratch offers both, a simple syntax and a graphical interface where programming is done by dragging blocks from a palette into a scripting pane. These blocks can be put together like puzzle pieces, if applied “syntactically” appropriate. The results are stacks of blocks which can be run individually or simultaneously [25].

In Figure 4.9 the graphical interface of a Scratch project is shown which is divided into four areas. The pane on the left side is the palette including the command blocks that can be dragged in the scripting area, which is located in the middle. In the basic version of Scratch there are eight different categories of command blocks (Motion, Control, Looks, Sensing, Sound, Operators, Pen and Variables) available. A stage (background) is located in the top right corner and below is an area that shows thumbnails of all different sprites in the project. Sprites are the main objects in a Scratch project. Each sprite has its own variables, images, sounds and especially scripting area. In a new project the Scratch Cat is shown on the stage as the default sprite [25].

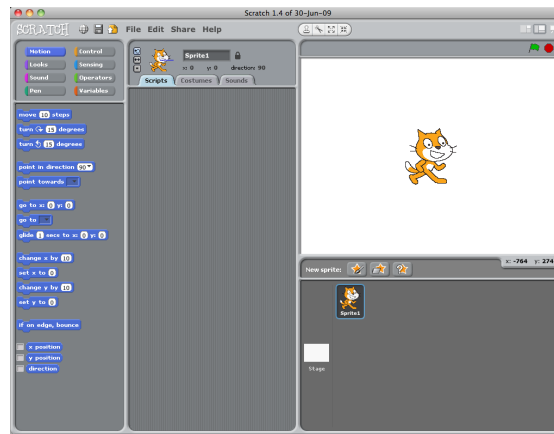


Figure 4.9: Screenshot of Scratch Interface

Technically Scratch runs on a virtual machine and is written in Squeak, an implementation of Smalltalk-80. Besides its basic functionality, Scratch allows students to import their own costumes and sounds. Additionally Scratch can be connected with sensors via USB and therefore allows interaction with the physical world [24].

Although extending Scratch is not an easy task, eager programmers are constantly developing new Scratch Modifications (often shortened to 'mod') to add even more elements and blocks to the basic application. These edited versions however are not supported by the original Scratch program and cannot be shared on the MIT website. Moreover they are not allowed to use the word "Scratch", but only "Based on Scratch by MIT" [35].

Sound functionality of Scratch

Scratch offers several controls for sound generation. Figure 4.10 shows all the different command blocks in the sound category. Sound files can be imported to Scratch and played, different types of drum sounds can be created and notes can be played with different instruments. Additionally the volume and the tempo can be adjusted. Putting several command blocks together, a progression of different sounds and notes can be created.

The following explains the commands the prototype will use:

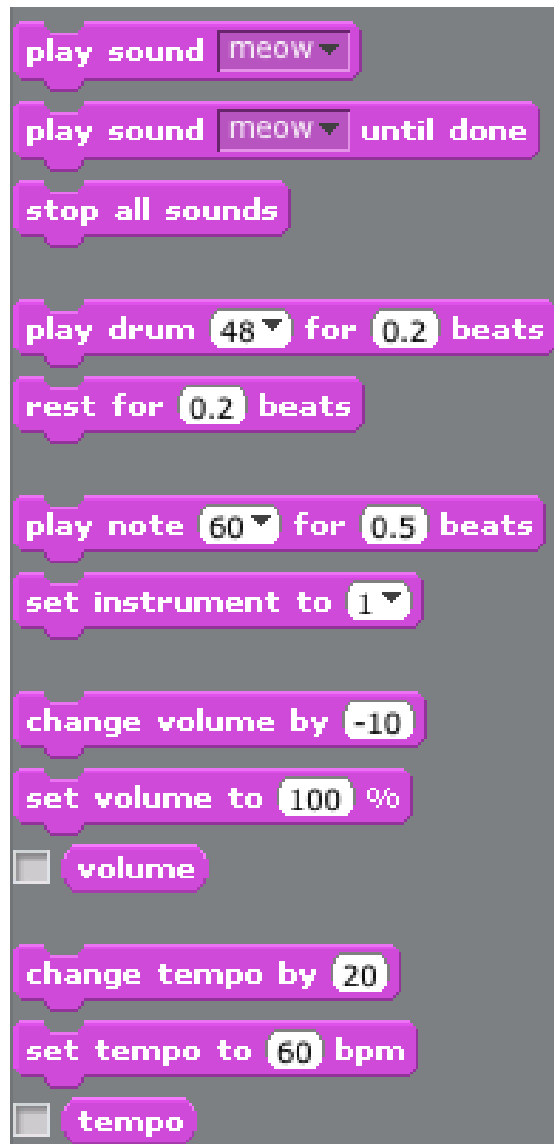


Figure 4.10: Scratch sound controls

- **play sound:** This command block has one parameter which defines the sound file to be played. A sound file has to be imported to Scratch before it can be played.
- **play note:** This command block has two parameters. The first parameter defines the pitch of the note to be played and the second defines how long it shall be heard. In Scratch each pitch is represented by a number. Each increase or decrease by 1

results in a note a semitone higher or lower. All notes available on a traditional keyboard are covered by the play note command block. 60 refers to the note C, 61 to Cis, 62 to D and so on with 72 being the note C again one octave higher and 48 being the note C one octave lower. By putting together a sequence of notes a whole song can be played with the play note control. The second parameter determines for how many beats a note shall be played. The beat length depends on the tempo setting. If the tempo is set to 60 beats per minute and the note shall be played for 1 beat it can be heard for exactly 1 second. All notes are played according to the instrument, volume and tempo settings.

- **set instrument to:** This command block has one parameter which defines the instrument. In Scratch each instrument has its own number (e.g. 41 refers to a violin).
- **set volume to:** This command block has one parameter which defines the volume. The span goes from 0 referring to completely silent up to 100 referring to the loudest option.
- **set tempo to:** This command block has one parameter which defines the beats per minute. The default value is 60 which means that one beat equals one second. A note played for one beat can be heard for 1 second in this case. If the tempo is increased to e.g. 120 beats per minute the same note would be played only for half a second. On the other hand a decrease to 20 beats per minute results in the note being played for 3 seconds.

Connecting Scratch to other languages

It is possible to connect the Scratch programming environment to other programming languages such as Java, C/C++ and others via the remote sensor connections protocol. This protocol allows other programs to connect with Scratch which can then send and receive messages called broadcasts over the port 42001. By doing so the possibilities for sound generation and additional tasks are extended immensely. In Java, which is used for this project, a socket has to be created to enable communication with the port Scratch is communicating with. Once a connection exists, messages can be broadcast

from Scratch to Java and vice versa.

In the prototype the sound functionality of Scratch is used for the music composition to play notes of a certain pitch for a certain duration. Additionally the instrument, tempo and volume can be changed via the sound functionality offered and the PicoBoard can be easily accessed. Also the simple interface allows quick adoptions. All of this makes Scratch especially suitable for the music composition scenario. In the tangible sound generation for children scenario the sound files of the animals are played via Scratch.

4.2.2 LilyPond

LilyPond [31] is a compiler to create high-quality music notation. The input of the program is a specially formatted file which holds the formal representation of the music. This file is read and processed by the compiler and either a PostScript or PDF file is produced. Figure 4.11 shows a very simple input fragment on the left with its respective output on the right side. The two notes in this example are c'4 and d'8. The first letters c and d are the note names while the numbers stand for the note value. 1 represents a full note, 2 a half note, 3 a quarter note, 4 an eighth note, and so on.

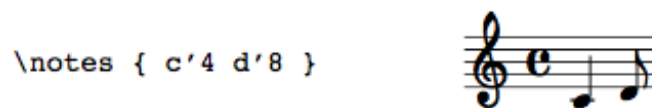


Figure 4.11: Simple LilyPond input fragment with output [31]

Once the input file is ready, the process of compiling can begin. This process is basically split into two parts. In the first part the input is parsed and the descriptions of the musical content are converted to graphical objects. This first step is called interpreting the input and the result is an unformatted score. Figure 4.12 shows the collection of graphical objects after interpreting the input from figure 4.11.

Every single musical symbol is represented by a graphical object. Each of these objects stores information about style and layout in variables. `obj8` from figure 4.12 is abstract which means it does not produce any output. Once the unformatted score








 obj1: glyph-name="treble"	 obj2: glyph-name="fourfour"
 obj3: position=-6 stem=→obj4	 obj4: line-thickness=0.12 note-head=→obj3
 obj5: position=-5 stem=→obj6	 obj6: line-thickness=0.12 note-head=→obj5
 obj7: line-count=5 staff-space=1.0 line-thickness=0.1	obj8: <i>container</i> elements= →obj1, ..., →obj7 height= <i>group-height</i>

Figure 4.12: Graphical objects from the unformatted score [31]

is created the next step performed in the process is the formatting. In this step all the spacing and line breaks are calculated and layout details are processed. The result of this step is a finished score which looks like a music sheet and can be saved as either PostScript or PDF file. Figure 4.13 shows a more complex music sheet created with the LilyPond compiler.

Technically LilyPond is implemented in the programming language Scheme and includes the GUILE Scheme interpreter which allows programmers to extend LilyPond and write their own features. LilyPond is open source and licensed under terms of the GNU General Public License, and can thus be freely used and modified [31]. Other software with similar functionality include Guido [12] and Haskore [14].

In the prototype LilyPond is used in both of the music composition scenarios to create high-quality music notation. After recording a sequence of music symbols LilyPond can transform the resulting piece into a score. The well-arranged syntax, extensive features and high-quality printing make LilyPond an excellent selection for the music notation processor.



Figure 4.13: A more complex music sheet produced with LilyPond [31]

4.2.3 PureData

PureData, also simply called Pd, is a powerful visual programming language used to create applications without writing any lines of code. PureData is used by musicians, other artists, researchers and developers to develop software graphically. PureData can be used to generate or process e.g. sound, video, graphics and MIDI. It can be used to learn simple multimedia processing, but also to learn methods of visual programming. Additionally it is used for developing complex systems for large projects. PureData is distributed open source and can thus be used by everybody interested in visual music and multimedia programming [16].

In the prototype a PureData application provides sound effects for the tangible composing with sounds. PureData has a powerful interface, a large community, plenty free-to-use compositions available online and is used by many state-of-the-art tangible mu-

sical interfaces, most notably the reactTable [20]. This makes it a good choice for our sound composing scenario.

4.2.4 RFID technology

Radio-frequency identification (RFID) is a popular technology for tangible interfaces. The basics how RFID works are covered in 2.2.

For the prototype an RFID reader and tags are used for the primary interaction. Since a lot of building blocks are required for the prototype and each building block carries an RFID tag within, a large number of tags is required. Thus we decided to use passive tags which are very inexpensive and a lot cheaper than active tags. In addition to their low cost, these tags can also be pretty small up to the size of a small coin which makes them especially suitable for the containers used as building blocks. Passive tags also have a shorter reading distance and do not contain a battery [48]. However the application does not require a large reading distance. The RFID reader used for the prototype has a reading distance of about 20 cm according to the specifications which is more than enough. The operating frequency of both the reader and tags is 125Khz.

4.2.5 PicoBoard

The PicoBoard is a small device which allows users of Scratch to interact with the real world with various sensors. The device incorporates a slider, a light sensor, a sound sensor, a button and four inputs that can sense electrical resistance. Figure 4.14 shows a PicoBoard and points out its various features. The slider value reaches from 0 (left) to 100 (right), the light value from 0 (completely dark) to 100 (very bright), the sound value from 0 (silent) to 100 (very loud), the button value is either true (pressed) or false (not pressed). The other four sensors measure the electrical resistance and have a default value of 100 if no alligator clips are used [36].

In the prototype the PicoBoard is used for changing different settings of the music playback. The simple interaction with the various sensors and the easy connection to

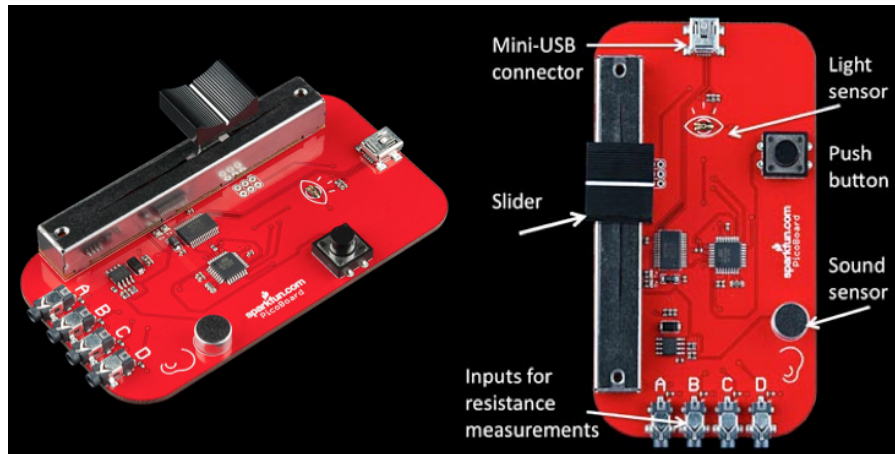


Figure 4.14: PicoBoard with features highlighted [38]

Scratch make the PicoBoard a good choice as tangible interface for the prototype.

4.3 Prototype

The prototype system covers the four application scenarios described in 4.1.2 and consists of passive RFID tags, an RFID reader (125Khz operating frequency), a PicoBoard, different music and sound building blocks and utilizes the visual programming environment Scratch (version 1.4), the music notation processor LilyPond (version 2.16.2-1), the visual programming language PureData (version 0.43.4-extended) and the programming language Java (version 1.6). A notebook is used to run the software and communicate with hardware components via USB.

Figure 4.15 gives an overview of the prototype system. The core of the prototype system is a Java application which is used to connect all different pieces of hardware and software. User interaction with the system takes place over the hardware components either by reading an RFID tag with the reader or by changing the slider value of the PicoBoard.

Prior to using the application the RFID tags can be configured. The class *Config.java* provides a simple graphical user interface to do this in a fast and efficient way. The following describes the configuration process of the RFID tags:

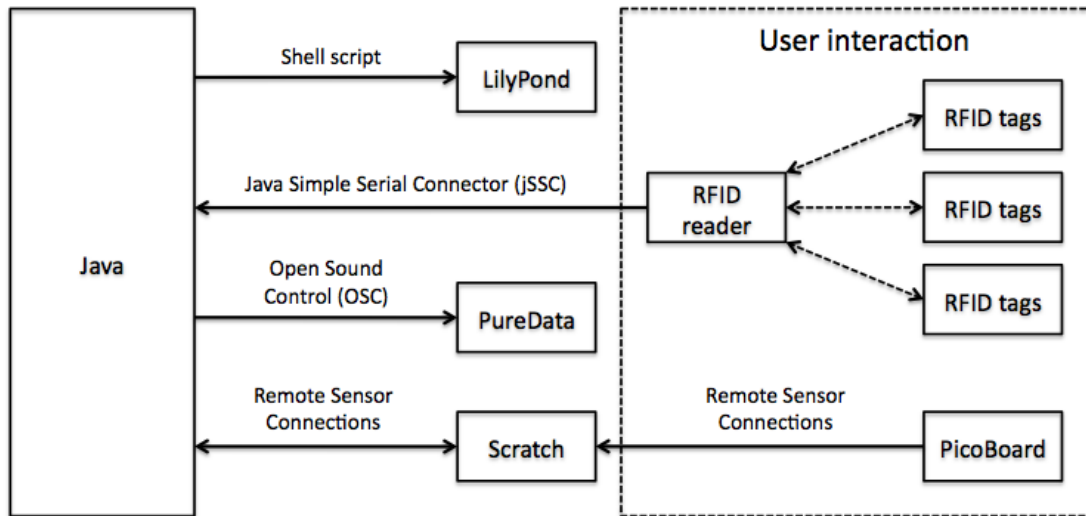


Figure 4.15: Overview of the prototype system

1. Config.java is run as Java application.
2. A graphical user interface appears showing all action commands in one checkbox.
3. The desired action command is selected.
4. Whenever a tag is read the selected action command is saved together with the unique ID of the respective tag in the configuration file *config.txt*.

Basic functionality

The main components of the prototype are containers which are represented by matchboxes storing RFID tags within them. For the different application scenarios different actions are necessary. Thus each RFID tag gets a specific action command assigned to its unique ID. In order to identify the action command associated with an RFID tag a symbol is labeled on the matchbox the respective RFID tag is located in. The following part explains the basic functionality of the system:

1. Composition.java is run as Java application.
2. The configuration is read from the configuration file.
3. The application starts listening to the serial port the RFID reader is communicating with.
4. An RFID tag is read by the RFID reader.
5. The unique ID of the RFID tag is sent to the Java application via the *Java Simple Serial Connector*.
6. Based on the configuration the Java application determines the action command associated with the tag ID.
7. The action command is processed.
8. Repeat from 4.

A total of 89 action commands exist which can be classified as:

- **Notes:** The notes available cover 14 pitches (from the note *B* to the note *A* two octaves higher) and four note values for each of them (whole, half, quarter, eighth). This results in a total of 56 notes. The respective action commands are *b1, b2, b4, b8, c'1, c'2, c'4, c'8, d'1, d'2, d'4, d'8, e'1, e'2, e'4, e'8, f'1, f'2, f'4, f'8, g'1, g'2, g'4, g'8, a'1, a'2, a'4, a'8, b"1, b"2, b"4, b"8, c"1, c"2, c"4, c"8, d"1, d"2, d"4, d"8, e"1, e"2, e"4, e"8, f"1, f"2, f"4, f"8, g"1, g"2, g"4, g"8, a"1, a"2, a"4, and a"8*. (Note: the ' and " signs indicate that a note is one or two octaves higher than the default octave. This notation is taken from *LilyPond* and used so it can be easily processed later.) Whenever a note is processed a broadcast containing the action command is sent to Scratch to play the specific note with the *play note* command in the correct pitch and duration.
- **Rests:** Four rests are available which are a whole, half, quarter, and eighth rest. The respective action commands are *r1, r2, r4, and r8*. (Note: Rests only have an effect when recording is active.)

- **Accidentals:** The signs flat and sharp exist which lower / raise the pitch of the next note by a semitone. This extends the number of pitches which can be reached with the application from 14 to 25 and the span of pitches from Bb to Bb two octaves higher. The respective action commands are *flat* and *sharp*.
- **Repeat signs:** The left repeat sign and right repeat sign are needed for defining the start and end of a repeat passage. The whole part between the left and the right repeat sign is played once more during playback. The respective action commands are *repeatStart* and *repeatEnd*. (*Note: Repeat only has an effect when recording is active.*)
- **Sound effects:** There are a total of six sound effects with one additional control unit to enable / disable the sound effects. Whenever a sound effect is processed a message is sent to a PureData application via *Open Sound Control* to enable / disable the associated sound effect. Whenever the control unit is processed it enables / disables all sound effects depending on its previous state. The respective action commands are *circle*, *cross*, *pentagon*, *rectangle*, *star*, *triangle*, and *control*.
- **Animal noises:** There are a total of seven animal noises which are stored in a sound file each. Whenever an animal noise is processed a broadcast containing the action command is sent to Scratch to play the sound file with the *play sound* command. The respective action commands are *chicken*, *cow*, *dog*, *duck*, *horse*, *pig*, and *sheep*.
- **Record / Stop:** Starts / stops a recording. Whenever a new recording is started the first tag processed determines whether the recording is a music or sound recording. If the first tag processed is associated with a note, rest, accidental or repeat sign the recording is set to music recording which means no sounds are recorded. Otherwise when the first tag processed is associated to a sound effect or animal noise the recording is set to sound recording which means no music is recorded. Thus a music recording can consist of only notes, rests, accidentals and repeats and occurs in the two music composition scenarios. A sound recording consists of only sound effects and animal sounds and occurs in the two tangible sound generation scenarios. In both, music and sound recordings, all action commands are

saved chronologically ordered together with a time value in milliseconds which defines the time elapsed since the last read tag. This time value is important for later playback. The respective action commands are *record* and *stop*.

- **Play:** Plays back the recorded music or sound sequence. It is important to distinguish between music and sound recordings.

In music recordings all action commands are associated with music symbols which all have a fixed duration. Whenever a music symbol is processed the next one follows immediately. Thus it is not necessary to consider the time intervals between reading the building blocks from the original recording. However a modifiable tempo value defines how fast the music symbols are processed. A faster tempo value results in shorter intervals between sending broadcasts containing information about notes to Scratch which takes care of playing the notes. A slower tempo value results in longer intervals between processing the notes. Besides the current tempo value the intervals between sending broadcasts also depend on the note value. A whole note compared to a half note obviously takes twice the time before the next note can be processed. On the other hand four eighth notes can be processed in the same time one half note is processed. The instrument, pitch, tempo and volume of a playback can be modified with the respective settings which are explained further down this list.

In sound recordings it is important to know the time elapsed since the last action processed because unlike music symbols, action commands associated with sound effects and animal noises do not provide any information about time. In this case the additional saved time value is used. Whenever a sound recording is played back the application measures the time elapsed and waits until the first time value of the chronologically first action command is reached. It then processes the action command and waits until the time value of the second action command is reached and so on. Thus the played back sound recording sounds exactly like the original recording. The respective action command is *play*.

- **Write:** Transforms the recorded music into a score with the music notation processor LilyPond. Therefore a *.lytex* file with some default layout settings is cre-

ated in which the recorded musical symbols are written. After writing the file a shell script is executed which compiles the file with LilyPond and subsequently saves and opens the resulting score as PDF file. To assure a certain simplicity all scores printed by LilyPond have a default time signature of 4/4 which is indicated in a score by the symbol which looks like a *c* to the right of the treble clef. The respective action command is *write*.

- **Save / Load:** Recorded music and sound sequences can be saved into a text file and loaded again. When saving a recording all action commands recorded are saved together with the time elapsed since the previously processed action command. This time value is important for later playback. The respective action commands are *save* and *load*.
- **Delete:** Deletes the currently active recording and enables a new music or sound recording. Saved files are not affected by this action. The respective action command is *delete*.
- **Instrument:** There are four settings which change the way a note is played in Scratch which are instrument, pitch, tempo and volume. This settings can be modified via the slider of the PicoBoard no matter if there is currently a playback or not. In Scratch each instrument has its own number (e.g. 41 refers to a violin) and thus the instrument which is associated with the current slider value is selected. Whenever the slider is moved the instrument associated with the new slider value is selected. The respective action command is *instrument*.
- **Pitch:** This option changes the pitch according to the PicoBoard slider value. In Scratch each pitch is represented by a number with 60 referring to a *c'* in our notation. 61 is one semitone higher than *c'* and 62 therefore refers to a *d'*. 59 on the other hand is one semitone lower than *c'* and thus results in a *b*. *Note: The b is in the default octave and therefore does not have the ' sign unlike the c' which is a semitone higher.* Usually when playing notes with Scratch the number of the desired note is played, e.g. 62. In our approach however we define the variable *c'* with a value of 60 and let Scratch play our variable. In addition all other notes

refer to the variable c' which means a d' would be represented by a $c' + 2$ because a d' is two semitones higher than a c' . Whenever a broadcast containing the note d' is sent to Scratch, Scratch does not play a 62, but a $c' + 2$ with the *play note* command. The same applies for all notes in the whole application. Whenever the slider value changes, Scratch updates the c' variable to the new slider value. Since all notes refer to the modified c' variable the pitch of the whole playback changes. If the slider value changes by ± 1 the whole music piece is played back a semitone higher or lower. The respective action command is *pitch*.

- **Tempo:** This option changes the playback speed according to the PicoBoard slider value. The higher the slider value the faster the song is played back (more beats per minute). The respective action command is *tempo*.
- **Volume:** This option changes the volume according to the PicoBoard slider value (0 = completely silent, 100 = loudest option). The respective action command is *volume*.

Note: Only one of the four settings can be active at a specific time which is always the last one selected. The system however keeps all changes made to the settings. Thus it is possible to e.g. first lower the pitch, then increase the speed and finally change the instrument. All settings will be kept.

The RFID tags which have an action command associated are put into matchboxes. In the following we will refer to these matchboxes as building blocks. For the four different scenarios the following building blocks exist:

- Music building blocks (Action commands: notes, rests, accidentals, and repeat signs)
- Tower building blocks (Action commands: notes)
- Sound building blocks (Action commands: sound effects)
- Animal building blocks (Action commands: animal noises)

- Control building blocks (Action commands: record, stop, play, write, save, load, and delete)
- Setting building blocks (Action commands: instrument, pitch, tempo, volume)

Figure 4.16 shows all prototype components which are:

- 1 RFID reader
- 1 PicoBoard
- 85 music building blocks
- 7 tower building blocks
- 7 sound building blocks
- 7 animal building blocks
- 7 control building blocks
- 4 setting building blocks
- 7 toy animal types
- 1 toy tractor to carry the RFID reader

In the following we take a closer look at the application scenarios individually.

Tangible music composition

Description: This scenario deals with tangible music composition and is the most comprehensive of the four scenarios introduced in this thesis. The main components are music building blocks and controls which are represented by matchboxes storing RFID tags within them. Each RFID tag has a unique ID which is associated with a music or control symbol. This symbol is labeled on the matchbox the respective RFID tag is located in. In the following the components used are listed and the basic functionality

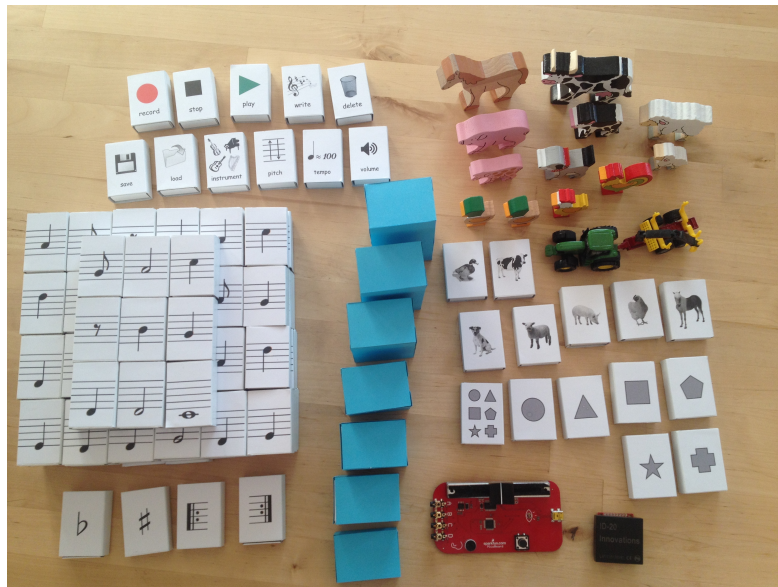


Figure 4.16: Overview of all prototype components

of the music composition is described.

Components used in scenario: RFID reader, PicoBoard, 85 music building blocks, 7 control building blocks, 4 setting building blocks.

Basic functionality:

1. Music building blocks are placed in the desired order.
2. The RFID reader is moved over the record control.
3. The RFID reader is moved over the music building blocks which are hereby processed.
4. The RFID reader is moved over the stop control.
5. After recording, the play and write controls can be used. The settings for music playback can be used along with the play control and modified with the PicoBoard.

6. The RFID reader is moved over the delete control to start from the beginning.

Additionally a recording can be saved anytime with the save control or loaded again with the load control. Figure 4.17 shows some building blocks with musical symbols.



Figure 4.17: Music building blocks

Figure 4.18 shows the control and setting building blocks for the music composition.

Tangible music composition for children

Description: This scenario introduces an approach to enable tangible music composition for children. Similar to the previous scenario music can be created by moving the RFID reader over the music building blocks. The only difference is that the building blocks in this scenario are designed as towers which are all of different size. The towers consist of matchboxes which are piled up and surrounded with colored carton in order to look like one single tower. Higher pitches are represented by taller towers while lower pitches are found in smaller towers. The higher a tower is the more matchboxes are piled up. The RFID tag is located in the top matchbox so it can be easily read when moving



Figure 4.18: Control and setting building blocks

the reader over the tower. Figure 4.19 shows some of these tower building blocks. To assure a certain simplicity only one key and one octave are covered by the tower building blocks and they do not differ in form and color. All controls like record, stop, write, etc. developed for the music composition also work in this scenario. The instrument, pitch, tempo and volume can be changed by the respective setting building blocks. Scratch takes care of the sound generation. In the following the components used are listed and the basic functionality of the music composition for children is described.

Components used in scenario: RFID reader, PicoBoard, 7 tower building blocks, 7 control building blocks, 4 setting building blocks.

Basic functionality:

1. Tower building blocks are placed in the desired order.
2. The RFID reader is moved over the record control. (*optional*)
3. The RFID reader is moved over the tower building blocks which are hereby processed.

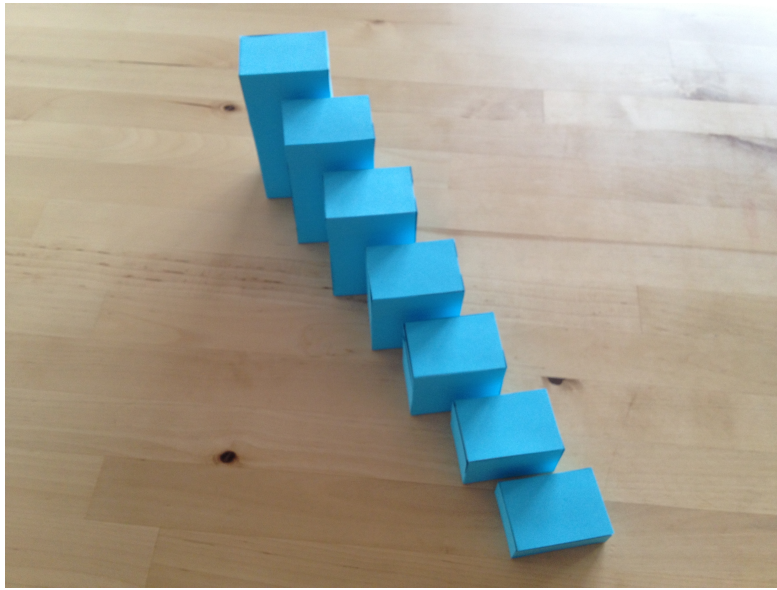


Figure 4.19: Tower building blocks of different size

4. The RFID reader is moved over the stop control. (*optional*)
5. After recording, the play and write controls can be used. The settings for music playback can be used along with the play control and modified with the Pi-coBoard. (*optional*)
6. The RFID reader is moved over the delete control to start from the beginning. (*optional*)

This scenario is designed for children aged 4-7 and thus does not require them to read or recognize symbols. The main functionality is simply reading the tower building blocks with the RFID reader. All other actions require understanding of the symbols which cannot be assumed for children this age. For the sake of completeness however we mention that the implementation of the prototype would allow the actions marked as (*optional*) in the basic functionality description.

Tangible composing with sounds

Description: Tangible composing with sounds also uses matchboxes with RFID tags and an RFID reader for the user interaction, but instead of Scratch, the visual programming language PureData takes care of the sound generation. The Java application communicates with PureData via Open Sound Control (OSC). There are six sound building blocks with one more control to turn the sound generation on and off. The sound building blocks have simple symbols labeled on it which are a circle, a cross, a pentagon, a rectangle, a star and a triangle. The control unit has all six symbols labeled on it. The only purpose of these symbols is to distinguish them from each other. Figure 4.20 shows the six sound building blocks with the control unit. Each symbol is assigned a different sound like a drum, piano, oboe or other instrument in a PureData application which is a modified version of the composition *christchurch.pd* from <http://www.obiwannabe.co.uk/html/compositions/compositions.html>. The version used has less sound effects than the original version and is extended to enable communication with Java via Open Sound Control. As soon as the RFID reader is moved over the control building block which turns on the sound generation, the sounds can be activated by moving the reader over the blocks and turned off by doing it again.

Components used in scenario: RFID reader, 7 sound building blocks, and 6 control building blocks (all except write).

Basic functionality:

1. The RFID reader is moved over the record control.
2. The RFID reader is moved over the sound control unit to enable sound generation.
(optional)
3. The RFID reader is moved over the sound building blocks which turn a sound on (if state is off) or off (if state is on).
4. The RFID reader is moved over the sound control unit to disable sound generation which means all sounds are turned off if not off already.



Figure 4.20: Sound building blocks with control unit

5. The RFID reader is moved over the stop control.
6. After recording, the play control can be used to play back the recorded sequence.
7. The RFID reader is moved over the delete control to start from the beginning.

Tangible sound generation for children

Description: Similar to the previous scenario this one deals with sound generation too by providing a simple environment for children to experiment with different animal noises. The RFID reader is placed on the trailer of a toy tractor and the RFID tags are attached to toy animals standing around on a toy farm. User interaction takes place by moving the toy tractor around and whenever it passes an animal the RFID reader on the trailer reads the RFID tag attached to the animal and a sound typical for the respective animal is played via Scratch which offers a command for playing sound files. The sound files with the animal noises are either selected from the sound files included with Scratch (dog, duck, horse, rooster) or if not available in Scratch taken from the website <http://www.animal-sounds.org/> (cow, pig, sheep). Since animals with a hollow interior for the RFID tags are not common the tags have to be attached on the outside of the

animal or alternatively are located in a matchbox the animal is standing on. Figure 4.21 shows the toy tractor carrying the RFID reader on a trailer and the animals standing on matchboxes which store the respective RFID tag of each animal.



Figure 4.21: Tractor with RFID reader and animals with matchboxes

Components used in scenario: RFID reader, 7 animal building blocks, 6 control building blocks (all except write), 7 toy animal types, and 1 toy tractor to carry the RFID reader.

Basic functionality:

1. The RFID reader is placed on the trailer of the toy tractor and the animals are placed on the animal building blocks.
2. The toy tractor passes the record control. (*optional*)
3. The toy tractor passes the toy animals which results in reading the animal building blocks which are hereby processed.
4. The toy tractor passes the stop control. (*optional*)

5. After recording, the play control can be used to play back the recorded sequence. *(optional)*
6. The toy tractor passes the delete control to start from the beginning. *(optional)*

This scenario is designed for children aged 4-7 and thus does not require them to read or recognize symbols. The main functionality is simply moving the toy tractor and passing animals with it. All other actions require understanding of the symbols which cannot be assumed for children this age. For the sake of completeness however we mention that the implementation of the prototype would allow the actions marked as *(optional)* in the basic functionality description.

CHAPTER 5

Evaluation

This chapter is split into two sections. In the first part the prototype is analyzed focusing on the functionality of the designed artifacts regarding the application scenarios and references to notable related work are mentioned. The second part discusses possibilities and limitations of the prototype system by looking at both the software used and the hardware components.

5.1 Functionality of the prototype

This section focuses on the functionality of the prototype and covers the four application scenarios for music and sound composition.

The most comprehensive scenario is the tangible music composition which will be tested with two songs. The first song is a popular English nursery rhyme called *Mary had a little lamb*. The second song is the famous chanty *Drunken Sailor*. The notes for these two songs are slightly simplified versions from the sheet music found at [30] because some special symbols had to be removed and thus some notes be adopted to work with the music symbols introduced. *Mary had a little lamb* is a simple song with 26 notes in the version used. These notes are selected from all music building blocks and lined up in the correct order in two lines which is shown in figure 5.1. In the next step the recording is started by reading the record control block with the RFID reader

and subsequently all music building blocks are read. Then the recording is stopped by reading the stop control block and the recording is transformed into a score by reading the write control block. The resulting score can be seen in figure 5.2. Comparing the symbols of the lined up music building blocks and the symbols in the finished score one can see that all symbols are equal. The symbol looking like a *c* to the right of the treble clef stands for common time and indicates the measure 4/4 which is the most common time signature and the default time signature in LilyPond if not defined otherwise. To assure a certain simplicity for the music composition only common time is used. The number 5 at the beginning of the second line indicates that the line starts with the fifth measure of the score. The line-breaks are added by LilyPond automatically to assure a visually appealing score.

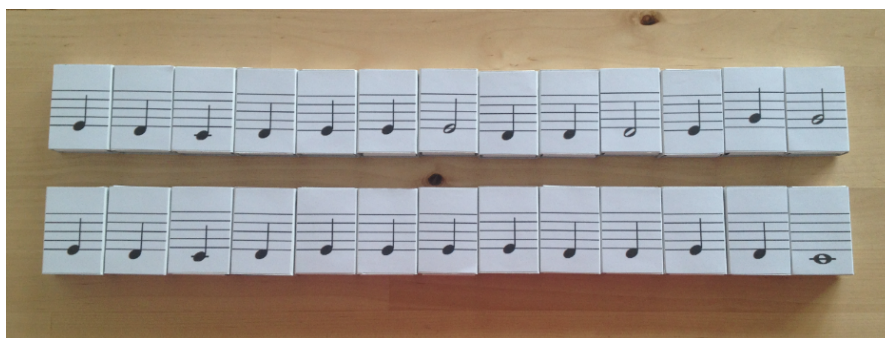


Figure 5.1: Mary had a little lamb building blocks



Figure 5.2: Mary had a little lamb LilyPond score

The second song Drunken sailor has 66 notes in the version used which are also selected from all music building blocks and lined up in the correct order which is shown

in figure 5.3. Then the recording is started and the music building blocks are read. Then the recording is stopped and like before transformed into a score. The result can be seen in figure 5.4. Comparing the symbols in this example one can see that all symbols from the building blocks and in the score are equal.

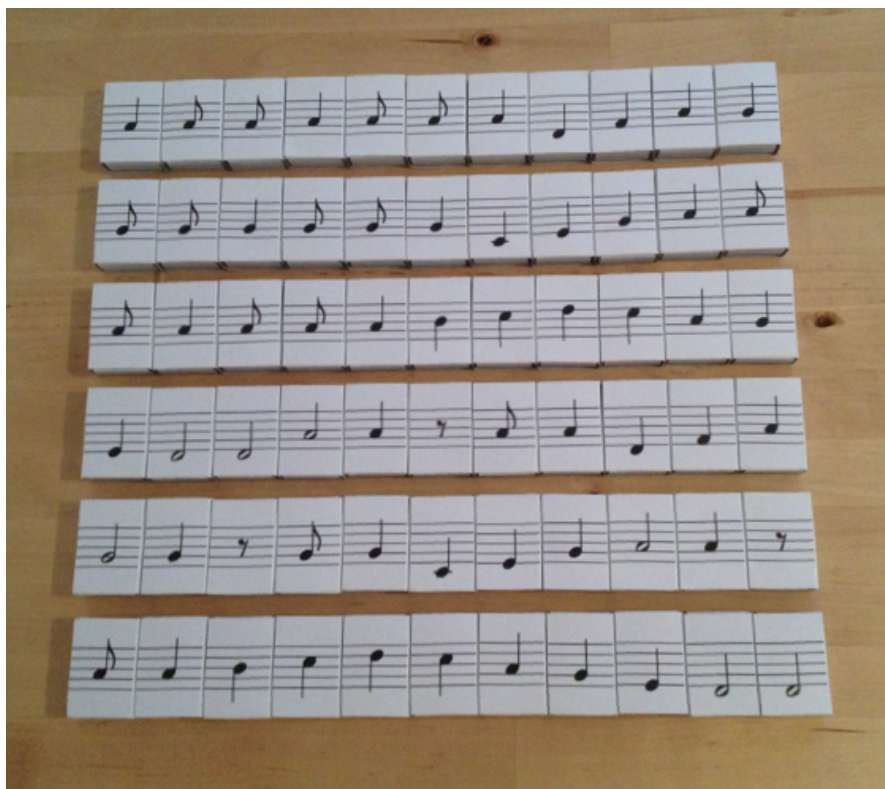


Figure 5.3: Drunken Sailor building blocks

Whenever a music building block with a note is read the note is played in its correct pitch and duration. However the time intervals between two notes depend only on the reading speed and no tempo is considered. No acoustic feedback is given when a rest or special symbol is read.

After recording, the compositions can be played back via the play building block. All four settings (instrument, pitch, tempo, volume) were tested and work with both examples. During the playback each note is played in its correct pitch and duration. The intervals from the original recording are also not considered anymore because all



Figure 5.4: Drunken Sailor LilyPond score

music symbols are processed subsequently without any time intervals in between. Thus both versions sound more accurate during playback than during their original recording.

The tower building blocks for the music composition for children are tested by recording some tower building blocks and playing the respective notes back afterwards. This scenario is similar to Instant City [11] in terms of user interaction where also building blocks are used to create and modify music.

Even though the music composition and the music composition for children are two separate scenarios they also work together when mixing the music building blocks with the tower building blocks.

The tangible composing with sound is tested by reading different sound building blocks and observing the behavior. The sound building blocks are not processed until the control block is read. Then each time a building block is read the respective sound either starts playing or stops playing depending on the previous state. By reading the control block again all sounds stop. Figure 5.5 shows how a saved file of such a sound recording looks like. The first two columns specify the ID of the RFID tag read and the associated action command. The third column indicates the time elapsed in milliseconds since the previous command processed. In this example the first command is the *control* command which turns on the sound composition. The 2125 time value indicates that

about 2.125 seconds passed since the last command (in this case the *record* command). The second command processed is the *star* command 2.205 seconds after the *control* command. This turns on the sound effect associated with the *star* command. In the 6th line the *star* command is processed again which indicates that the sound effect associated with the command is turned off again. At the end the *control* command turns off the sound composition and thus turns off all sound effects still active. When playing back the whole sound sequence the commands are processed with the exact time intervals from the original recording.

```
08009D0E30AB control 2125
0A00506DD1E6 star 2205
0A004F8F569C pentagon 3796
0A00501BB4F5 triangle 4085
0A004F9FFB21 cross 3035
0A00506DD1E6 star 408
08009C754DAC rectangle 2510
0A004F7ECDF6 circle 500
0A004F8F569C pentagon 3310
0A004F7ECDF6 circle 2579
0A00506DD1E6 star 1749
08009D0E30AB control 3062
```

Figure 5.5: Recorded sound composition

This scenario is similar to Audiocubes [34] in terms of sound generation. In Audiocubes different cubes are used to create simple sound effects when placed next to each other for the main purpose of experimenting with different sounds rather than enabling meaningful composition. Audiopad [32], which uses knob based controllers equipped with RF tags to experiment with different sounds, also utilizes radio waves as technology. The reacTable [19] inspired this approach to use PureData as sound synthesizer.

The sound generation for children works by moving the toy tractor with the RFID

reader by the different toy animals standing upon the respective matchboxes which store their sound. Figure 5.6 shows how a saved file of such recording looks like. The first command is the *duck* command which sends a broadcast to Scratch to play the respective sound file. The second command is a *horse* command which is processed 4.062 seconds after the first command. The time intervals in this example are a bit longer than those in the sound composition because it usually takes longer to move the toy tractor around and past the animals compared to moving the reader around directly.

```
0A005036E68A duck 3734
0A004CBAD32F horse 4062
0A004F253E5E cow 3148
0A004F232244 dog 6446
0A004F047233 sheep 4192
0A004F1EFCA7 chicken 4980
0A005036E68A duck 3404
0A004F2383E5 pig 3254
0A004F232244 dog 6010
0A004F1EFCA7 chicken 3784
0A004CBAD32F horse 4821
0A004F253E5E cow 5852
```

Figure 5.6: Recorded animal noises

In terms of tangible representation of sounds this scenario is similar to MusicPets [44] which allows its users to store sounds on tangible containers in the form of soft toys. Another related project is the Marble Track Audio Manipulator [3] where different marbles represent different sound clips and tracks represent sound effects.

The RFID reader delivered a great performance by moving it over the RFID tags at about 3-7 cm height. The reading was fast and precise, but got worse when the reader was closer to the matchboxes or further away. When the matchboxes were put in an upright position and arranged close to each other the system could not process the tags

anymore. No matter what the reading distance was no reliable reading of the tags was possible in that case.

5.2 Possibilities and limitations

First and foremost it is important to mention the limitations of this work. This thesis presents a prototype for tangible music and sound composition. As such it introduces concepts how tangible interaction might take place and discusses possible benefits for its users. The evaluation is done by testing the functionality of the prototype and highlighting possibilities and limitations regarding performance, reliability, accuracy and usability. The analysis of the prototype system however does not include any test persons. An empirical study is needed to evaluate user behavior with the designed prototype and explore if and how users benefit from the discussed possibilities. The remainder of this section is spent pointing out deficits of this work and ideas for improvement.

The music composition currently works with the treble clef, the most common notes, note values and rests, repeat signs and the accidentals sharp and flat. In order to print also more complex scores the program could be extended to work also with dotted notes, additional clefs like the bass clef, time signatures, note relationships, dynamics, articulation marks and other special signs. Also for complex composition more music building blocks would be necessary obviously.

The change of instruments currently works with the PicoBoard slider and selects the instrument which is represented by the current slider value in Scratch. In order to make the instrument change more transparent additional building blocks could be added with instrument symbols so one can select a desired instrument directly without looking for the right slider value. Therefore additional setting building blocks would need to be created with symbols of the respective instruments labeled on them. An RFID tag in such a block would contain e.g. the action command *violin* and when processed sets the Scratch instrument value to *41*.

Currently the PicoBoard is only used for changing the tempo, instrument, volume and pitch of the music composition. By adding more sophisticated effects to the sound composition part the PicoBoard sensors could also be used for composing and not just

for the play back settings.

The matchboxes used for the building blocks are of the size 5.3 x 3.7 x 1.5 cm (length x width x height) and the RFID tags are 3.4 x 2.8 cm (length x diameter). When measuring the exact free space in the matchboxes besides the tags the width of the matchboxes could be about 0.5 cm smaller and still carry the tags within. This would increase the number of building blocks that could be used in one line on a table or any other surface if space is limited because less is needed. If matchboxes are 0.5 cm smaller (3.2 cm instead of 3.7 cm), the number of matchboxes which can be used in one line would increase by 15.625 percent.

The reading distance of the RFID reader should be 20 cm according to the specifications but is actually only little less than 10 cm. Also both the RFID reader and the PicoBoard could be deployed easier if they would be wireless.

For the music composition for children towers are used which consist of matchboxes also. The higher a tower is the more matchboxes are piled up and surrounded with colored carton so it looks like one single tower. This solution works fine for a couple of towers and clearly visualizes the concept but in order to play simple songs more towers might be needed. Since the most common toy bricks are too small to fit the RFID tags used within them, a good solution still needs to be found for larger amounts.

The tangible composing with sounds currently works only with very limited interaction possibilities. The different symbols like a star, circle or cross are only for distinguishing the different sound building blocks and do not provide any information about the sound effect generated when a block is processed. The scenario serves basically as an alternative for the music composition because users do not have to think about notes and musical theory and can just experiment with different sounds. However the scenario provides the basic infrastructure for communicating with PureData and thus more sophisticated, future sound applications can be easily integrated in the current system.

For the sound composition for children toy animals are used but no animals were found which have a hollow interior. Therefore the RFID tag could not be stored in the animal but had to be attached to it on the outside or stored in an own matchbox the toy animal is standing on which makes the overall appearance less appealing.

Conclusion and Future Work

This thesis deals with tangible musical interfaces, a term used to describe user interfaces which directly interact with music over physical objects. In the scope of this work a prototype for tangible music and sound composition was developed for educational and artistic purposes. Based on an extensive literature review considering the history of human-computer interaction, the evolution of tangible interfaces, tangible technologies and state-of-the-art tangible musical interfaces, we introduced prototyping as method to design systems, possible application scenarios were evaluated, design requirements defined, and sketches created. During that design process we examined how data objects containing information about music and sound can be realized with tangibles and how music and sound can be generated with such. We further discussed how tangible composition can be modeled especially for children. We presented our approach of tangible music and sound composition with RFID technology and different music and sound building blocks. The outcome of the design and implementation process was a functional prototype system for music and sound composition offering four scenarios to experiment with. In the music composition users can compose their own songs with music building blocks and record, play back and transform their compositions into a score. The sound composition offers several sound building blocks to creatively interact with different sounds. Additional scenarios for music and sound composition were designed for children. The evaluation conducted demonstrates the functionality of the prototype

system and its application scenarios and discusses possibilities and limitations of the designed artifacts in terms of performance, reliability, accuracy and usability.

6.1 Future work

The presented prototype system for tangible music and sound composition can be seen as another step of developing new tangible musical interfaces for educational and artistic purposes. The approach of using RFID technology to build music building blocks is just a basic foundation for further research and can be extended in many ways. The following part discusses possible future work.

- Especially important will be further empirical work in the area of tangible user interfaces in general and children learning in particular. So far this work focuses on inventing new approaches of tangible interaction with music without investigating the user behavior or benefits such applications can have.
- The music composition currently works with a limited number of only the most important music symbols. In order to print also more complex scores the program needs to be extended to work also with additional symbols like dotted notes, additional clefs like the bass clef, time signatures, note relationships, dynamics, articulation marks and other special signs.
- The interaction in the tangible composing scenario could be improved by designing symbols which give a clue about what sound is generated. This would make the whole interaction more meaningful since a user can accurately control which sound effect is added or muted.
- Besides adding more building blocks also additional sensors (e.g. tilt, distance) and sound effects (e.g. flanger, phaser, echo) could be added to extend the existing sound composition.

Music notation

This section explains the most important music notation basics. The music symbols introduced are the same used in this thesis for the music composition. The music symbols assigned to the building blocks are in accordance to the symbols found in sheet music. In the following we take a closer look at these musical symbols.

Figure A.1 shows a typical line in a score. The staff is the basis of each score and consists of five lines and four spaces in between. On the staff the notes and other symbols like rests are presented. The first sign from the left is the treble clef and defines the pitch. Different pitches are named by letters which are also the note names. These are C, D, E, F, G, A and B in ascending order by pitch. Each note is represented by a little oval symbol on the staff. The notes are located either directly on a line or in the space between two lines. The higher a note is located on the staff the higher pitch it has. The treble clef mentioned before defines that the note located on the lowest line corresponds to the note E. The note located between the lowest two lines in the empty space corresponds to the note F and so on.

Each note is played for a specific duration. The total length of a note depends on the time signature which is discussed later. Figure A.2 shows the relative note durations which are represented by note values. An empty oval symbol represents a whole note. An empty oval symbol with a vertical line represents a half note. A whole note is equal to two half notes. A filled oval symbol with a line represents a quarter note. Thus a



Figure A.1: Treble clef with different notes

whole note is equal to four quarter notes and a half note equal to two quarter notes. A filled oval symbol with a line and a flag represents an eighth symbol. The scheme is the same with all note values so a whole note is equal to eight eighths.



Figure A.2: Notes with different value

Figure A.3 shows the same concept for rests where the symbols from left to right indicate a whole rest, a half rest, a quarter rest and an eighth rest.



Figure A.3: Rests with different value

The vertical lines on the staff represent the measures which are used to organize music. The so called time signature determines the number of beats in a measure and what note value (i.e. whole, half, quarter, etc.) represents the beat. The symbol which looks like a *c* next to the treble clef stands for common time and indicates the measure 4/4 which is the most common time signature. It means that there are four beats per measure and a quarter note represents the beat. Thus one measure can consist for example of one whole note, two half notes, four quarter notes, eight eighth notes and so on.

Figure A.4 introduces two new concepts which are accidentals and repeats. The most common accidentals are flats represented by a *b* and sharps represented by a *#*. They modify the pitch of a note by one half step. A flat lowers the pitch of a note by one semitone, a sharp raises the pitch of a note by one semitone. The accidental always is located before (i.e. to the left) of the note it modifies. The bold vertical lines with two dots represent repeat signs. There are two types of repeat signs which are begin and end repeat signs. The begin sign always has the two dots on the right side of the bold vertical line while the end sign has the dots on the left side. All musical symbols between a begin and end sign are repeated [29].



Figure A.4: Accidentals and repeats

Bibliography

- [1] Diana Africano, Sara Berg, Kent Lindbergh, Peter Lundholm, Fredrik Nilbrink, and Anna Persson. Designing tangible interfaces for children's collaboration. In *Proceedings of the CHI '04 Conference on Human Factors in Computing Systems*, CHI '04, pages 853–868, New York, NY, USA, 2004. ACM.
- [2] Massimo Banzi. *Getting Started with Arduino*. O'Reilly Media, Inc., 2009.
- [3] Alex Bean, Sabina Siddiqi, Anila Chowdhury, Billy Whited, Orit Shaer, and Robert J. K Jacob. Marble track audio manipulator (mtam): a tangible user interface for audio composition. In *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*, TEI '08, pages 27–30, New York, NY, USA, 2008. ACM.
- [4] Michel Beaudouin-Lafon and Wendy Mackay. The human-computer interaction handbook. chapter Prototyping tools and techniques, pages 1006–1031. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.
- [5] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.
- [6] John M. Carroll. *Human Computer Interaction - Brief Intro*. The Interaction Design Foundation, Aarhus, Denmark, 2013.
- [7] Gillian Crampton Smith. The hand that rocks the cradle. *I.D. The International Design Magazine*, pages 60–65, 1995.

- [8] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the CHI '95 Conference on Human Factors in Computing Systems*, CHI '95, pages 442–449, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
- [9] Alexandre Gillet, Michel Sanner, Daniel Stoffler, and Arthur Olson. Tangible interfaces for structural molecular biology. *Structure*, 13(3):483 – 491, 2005.
- [10] Saul Greenberg, Sheelagh Carpendale, Nicolai Marquardt, and Bill Buxton. *Sketching User Experiences: The Workbook*. Morgan Kaufmann Publishers Incorporated, 2012.
- [11] Sibylle Hauert, Daniel Reichmuth, and Volker Böhm. instant city: a music building game table. In *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, NIME '07, pages 422–422, New York, NY, USA, 2007. ACM.
- [12] Holger H Hoos, Keith A Hamel, Kai Renz, and Jürgen Kilian. The guido notation format—a novel approach for adequately representing score-level music, 1998.
- [13] Michael S. Horn and Robert J. K. Jacob. Tangible programming in the classroom with tern. In *Proceedings of the CHI '07 Conference on Human Factors in Computing Systems*, CHI '07, pages 1965–1970, New York, NY, USA, 2007. ACM.
- [14] Paul Hudak, Tom Makucevich, Syam Gadde, and Bo Whong. Haskore music notation—an algebra of music. *Journal of Functional Programming*, 6(03):465–484, 1996.
- [15] Instant City Website. <http://www.instantcity.ch>. Accessed: 2013-11-25.
- [16] Institute of Electronic Music and Acoustics. Pure data – pd community site. Accessed: 2013-11-25.
- [17] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the CHI '97 Conference on Human Factors in Computing Systems*, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM.

- [18] Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Kevin Mackey. The xerox star: A retrospective. *Computer*, 22(9):11–26, 28–29, 1989.
- [19] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The re-actable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pages 139–146. ACM, 2007.
- [20] Sergi Jordà, Martin Kaltenbrunner, Günter Geiger, and Ross Bencina. The re-actable. In *Proceedings of the International Computer Music Conference (ICMC 2005)*, pages 579–582, 2005.
- [21] Martin Kaltenbrunner and Ross Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 69–74, New York, NY, USA, 2007. ACM.
- [22] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. Tuio: A protocol for table-top tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.
- [23] Martin Kaltenbrunner, Sergi Jorda, Gunter Geiger, and Marcos Alonso. The re-actable*: A collaborative musical instrument. In *Proceedings of the 2006 Workshop on Tangible Interaction in Collaborative Environments (TICE), at the 15th International IEEE Workshops on Enabling Technologies (WETICE 2006)*, pages 406–411. IEEE, 2006.
- [24] David J. Malan and Henry H. Leitner. Scratch for budding computer scientists. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 223–227, New York, NY, USA, 2007. ACM.

- [25] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. Programming by choice: Urban youth learning programming with scratch. *SIGCSE Bulletin*, 40(1):367–371, March 2008.
- [26] Paul Marshall. Do tangible interfaces enhance learning? In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 163–170, New York, NY, USA, 2007. ACM.
- [27] David Maulsby, Saul Greenberg, and Richard Mander. Prototyping an intelligent agent through wizard of oz. In *Proceedings of the CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 277–284, New York, NY, USA, 1993. ACM.
- [28] H. Mellar. *Learning with Artificial Worlds: Computer-based Modelling in the Curriculum*. Falmer Press, 1994.
- [29] Method behind the music website. <http://method-behind-the-music.com>. Accessed: 2013-11-25.
- [30] Music for music teachers website. <http://www.music-for-music-teachers.com>. Accessed: 2013-11-25.
- [31] Han-Wen Nienhuys and Jan Nieuwenhuizen. Lilypond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, pages 167–172. Citeseer, 2003.
- [32] James Patten, Ben Recht, and Hiroshi Ishii. Audiopad: a tag-based interface for musical performance. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*, NIME '02, pages 1–6, Singapore, Singapore, 2002. National University of Singapore.
- [33] Bert Schiettecatte. Interaction design for electronic musical interfaces. In *Proceedings of the CHI '04 Conference on Human Factors in Computing Systems*, CHI '04, pages 1549–1549, New York, NY, USA, 2004. ACM.

- [34] Bert Schiettecatte and Jean Vanderdonckt. Audiocubes: a distributed cube tangible interface based on interaction range for sound design. In *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*, TEI '08, pages 3–10, New York, NY, USA, 2008. ACM.
- [35] Scratch MIT Website. <http://scratch.mit.edu>. Accessed: 2013-11-25.
- [36] Scratch Wiki. <http://wiki.scratch.mit.edu>. Accessed: 2013-11-25.
- [37] Orit Shaer and Eva Hornecker. Tangible user interfaces: Past, present, and future directions. *Foundations and Trends in Human-Computer Interaction*, 3(1–2):1–137, January 2010.
- [38] SparkFun Electronics website. <https://www.sparkfun.com/products/10311>. Accessed: 2013-11-25.
- [39] Danae Stanton, Victor Bayon, Helen Neale, Ahmed Ghali, Steve Benford, Sue Cobb, Rob Ingram, Claire O'Malley, John Wilson, and Tony Pridmore. Classroom collaboration in the design of tangible interfaces for storytelling. In *Proceedings of the CHI '01 Conference on Human Factors in Computing Systems*, CHI '01, pages 482–489, New York, NY, USA, 2001. ACM.
- [40] Hideyuki Suzuki and Hiroshi Kato. Interaction-level support for collaborative learning: Algoblock – an open programming language. In *Proceedings of the 1st International Conference on Computer Support for Collaborative Learning*, CSCL '95, pages 349–355, Hillsdale, NJ, USA, 1995. L. Erlbaum Associates Inc.
- [41] Gunnvald B. Svendsen. The influence of interface style on problem solving. *International Journal of Man-Machine Studies*, 35(3):379–397, 1991.
- [42] Lucia Terrenghi, Matthias Kranz, Paul Holleis, and Albrecht Schmidt. A cube to learn: A tangible user interface for the design of a learning appliance. *Personal Ubiquitous Computing*, 10(2-3):153–158, January 2006.
- [43] Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. User sketches: a quick, inexpensive, and effective way to elicit more reflective user

- feedback. In *Proceedings of the 4th Nordic Conference on Human-Computer Interaction*, NordiCHI '06, pages 105–114, New York, NY, USA, 2006. ACM.
- [44] Martin Tomitsch, Thomas Grechenig, Karin Kappel, and Thomas Költringer. Experiences from designing a tangible musical toy for children. In *Proceedings of the 2006 Conference on Interaction Design and Children*, IDC '06, pages 169–170, New York, NY, USA, 2006. ACM.
- [45] B. Ullmer and H. Ishii. Emerging frameworks for tangible user interfaces. *IBM Systems Journal*, 39(3-4):915–931, July 2000.
- [46] John Underkoffler and Hiroshi Ishii. Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the CHI '99 Conference on Human Factors in Computing Systems*, CHI '99, pages 386–393, New York, NY, USA, 1999. ACM.
- [47] Catherine Vaucelle and Tristan Jehan. Dolltalk: a computational toy to enhance children's creativity. In *Proceedings of the CHI '02 Conference on Human Factors in Computing Systems*, CHI '02, pages 776–777, New York, NY, USA, 2002. ACM.
- [48] Ron Weinstein. Rfid: A technical overview and its application to the enterprise. *IT Professional*, 7(3):27–33, 2005.
- [49] Oren Zuckerman, Saeed Arida, and Mitchel Resnick. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In *Proceedings of the CHI '05 Conference on Human Factors in Computing Systems*, CHI '05, pages 859–868, New York, NY, USA, 2005. ACM.