RWTH AACHEN UNIVERSITY

Deformable User Interfaces

An Explorative Study of Malleable Input Devices in Mobile Applications

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Gero Herkenrath

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Stefan Kowalewski

Registration date: Oct 17th, 2007
Submission date: May 28th, 2008

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Contents

List of Figures

List of Tables

Abstract

Over the last years mobile electronic devices like cell phones or personal digital assistants (PDAs) have become more and more common. Their processing power increased and their size became smaller, allowing them to permeate people's everyday lives. An important issue with them are methods to provide input in a convenient way that does not interfere with their mobility. In most cases the main problem with this is to find a balance between space used for input and output of information, while at the same time making the device small enough so that it can be carried around. Traditional input methods found on desktop computers have proven not to be directly applicable to these devices, thus many new forms of interaction, like using special purpose keyboards, tilting the device or touching its screen have been developed. With technological advances a new form of interacting with a device are becoming possible. Flexible electronics allow a mobile computer to become deformable. This work describes a prototype framework meant to explore this new form of interaction by deformation. It focuses on bending a mobile device. The thesis shows what has been done in this field so far and where there still is need for research. To address this need, it proposes a system consisting of a hardware prototype that can measure how it is bend and an according software that enables to process this data into actions of an interactive application. It taps into the operating system of a hosting computer, thus allowing the hardware to control a large amount of possible applications by bending. Two user studies that show how the system works and at the same time explore bending as interaction gesture deliver new insights on this new concept and what forms of bending are suited for what kinds of actions.

Überblick

Über die letzten Jahre sind mobile Geräte immer alltäglicher geworden. Ihre Rechenleistung wuchs und ihre Gröše nahm ab, so daš sie den Alltag vieler Leute immer mehr durchdrangen. Ein wichtiger Aspekt was diese Geräte betrifft sind Methoden, Eingaben in einer bequemen Art und Weise auf ihnen zu tätigen ohne dabei mit ihrer Monilität zu interferieren. In den meisten Fällen ist dabei das Hauptproblem, eine Balance zwischen für Eingabe und Ausgabe von Information benutztem Platz zu finden und gleichzeitig das Gerät so klein zu halten, dass es tragbar bleibt. Traditionelle Eingabemethoden, wie man sie bei Desktop Computern findet haben sich als nicht ohne Weiteres für diese Geräte geeignet erwiesen, daher haben sich neue Formen der Interakltion entwickelt, wie etwa die Benutzung von speziellen Tastaturen, das Neigen des Gerätes or Berühren seines Bildschirms. Mit technologischen Fortschritten wird nun eine neue Form von Interaktion mit einem Gerät möglich. Flexible Elektronik erlaubt einem mobilen Computer, deformierbar zu werden. Diese Arbeit beschreibt ein Prototyprahmenwerk, das dazu gedacht ist diese neue Form der Interaktion durch Deformation zu erforschen. Es beschränkt sich dabei auf das Biegen eines mobilen Gerätes. Diese Diplomarbeit zeigt, was bisher in diesem Forschungsbereich geleistet wurde und wo noch Bedarf für weitere Forschungen besteht. Um diesen Bedarf zu klären, stellt sie ein System bestehend aus einem Hardware-Prototypen, der messen kann, wie er verbogen wird und einer entsprechenden Software vor, die es erlaubt, diese Messungen in Aktionen innerhalb einer interaktiven Applikation zu wandeln. Es h"angt sich dazu in das Betriebssystem eines Gastgebercomputers ein, wodurch die Hardware eine Vielfalt von möglichen Anwendungen durch Biegen steuern kann. Zwei Benutzerstudien, die zeigen, wie das System funktioniert und gleichzeitig Biegung als Interaktionsgeste erschlieše n, liefern Einsichten in dieses neue Konzept und welche Formen von Biegung f"ur welche Art von Aktionen geeignet erscheinen.

Acknowledgements

This thesis would not have been possible without the support and encouragement of many people. I want to thank my colleagues from the Media Computing Group for all the times they gave advice and constructive feedback, Silke Wettich and Reisgies Schaumstoffe for providing the used foam and all the people I met on the 2008 Conference on Human Computer Interaction.

Prof. Dr. Jan Borchers and Thorsten Karrer have supervised my thesis and constantly gave me the feeling I was going the right way. Both always answered my questions and helped me to even publish my project as a work in progress on CHI 2008. I also want to thank Pr. Dr. Kowalewsi who agreed to be second examiner, especially I almost literally assaulted him with this bidding.

Additional thanks go to all my friends, who never failed to lend me their ear when I was ranting about my project in some way or another. Especially Thomas Plätschke has to be mentioned for correcting some mistakes I wrote in my first draft of this thesis. I know how confusing it can be to read something from a completely foreign field of science, Tommy, thank you!

I also want to give thanks to all people who volunteered to take part in my user studies. Some even accepted to driver over 80 km to help me out and missed an important (or not so important) soccer game.

Special thanks go to my mother, who was always there for me and supported me emotionally, be it to cheer me up when I was down over some problem with my code or chime in when euphoria got the better of me. I wouldn't have been able to finish this without you, Mom, thank you a lot!

Conventions

Throughout this thesis the following conventions are used.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

`TWEvent`

The whole thesis is written in American English.

Download links are set off in coloured boxes. For the case the online resources are not accessible, a CD ROM provided with the print version of this thesis contains the linked files as well.

Command line tool: twend[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/twend.zip

Chapter 1

Introduction

Technological progress over the last decades has integrated computers into every person's daily life more and more. Gain in processing power allows ever more complicated tasks to be done and loss in size has made computers mobile. In fact, many devices have become so small that they can be carried around as conveniently as a watch but unfortunately their tiny dimensions give rise to a conflict between this mobility and their usefulness. Traditional input methods like buttons, knobs or gesturing devices, e. g. keyboards and mice on a desktop computer, often contradict the demand for a small, truly mobile device but without efficient input methods such a device is next to useless. The same is true for any mobile device's output method.

This dilemma means, the ways of human computer interaction have to be continuously expanded, balancing the real estate needs and limitations of input, output and mobility. A reasonable approach is to divide available space between input and output as it is needed by utilizing a touch sensitive display of some form. The display then functions as usual output and it marks, depending on the current context, certain areas as so-called "softbuttons" for input. This basically results in a as found in present standard PC operating systems, but it still has some drawbacks. So is the use of available space dynamically adapted to the current context and thus more efficient, but softbuttons still use up space that could be otherwise left out or used for output.

Occlusion by the user's fingers or input devices such as the pen of many personal digital assistants (PDAs) can also be a problem and the poor haptic feedback of touching a display is often an issue.
Another approach to the real estate problem is to use a mobile device's position in space. Tilting sensors, for example, used for user interaction are becoming more common. This idea, however, has drawbacks as well: Due to the portable nature of most mobile devices, positional change does not always reflect a certain intention by the user, limiting it as an input gesture.

This work explores a rather new way of interacting with a computing device — bending. The technological foundations to create a deformable electronic device have reached a stage where it is reasonable to assume that devices do not necessarily have to be rigid physical objects in the years to come. Bendable circuit boards, flexible displays and deformable batteries all have have already been realized. Allowing a user to control a computing task by actively manipulating the shape of a mobile device offers some benefits and improvements over conventional input methods. In point-and-click interfaces found in mobile devices, part of the dextrous system is used to just hold the device, while another part performs the input: Imagine a PDA being hold in one hand while the other hand types in the input. If the PDA was bendable, deformation could be done with the same hand that holds it. Leaving out the pen and holding it with two hands would even allow utilizing the relative position of both holding hands and wrist-movements for deformation. In general, bending gestures would have two improvements over traditional point-and-click interfaces: Like tilting, they would introduce one additional part of the dextrous system to the interaction, the wrists, and they would allow making more use of the hand that holds the device.

The current problem with bending as an input method is that there is only little knowledge concerning good metaphors and natural mappings between this kind of gesture and actions a computer system could perform. Of course the most obvious mapping of any kind of deformation is found in 3D-modelling. The change in shape of the device is directly applied to a virtual representation of this

shape. However, this is an area usually not relevant to mobile devices. Therefore it would be interesting to know if there are other mappings, perhaps more abstract ones for a broad range of applications. A challenging yet important long-term research goal would be to establish a guideline for bending gestures as interaction method on which designers could rely once bendable devices become broadly available.

To provide a start in this research, this thesis introduces a prototype framework to investigate a subset of possible bending gestures. Of course the possible ways to bend a device are numerous, but may be limited by taking into account the usual shape of a mobile device, the way in which users usually hold the device and what kind of application with what type of actions mobile devices these days offer.

Following this line of thought, the prototype framework consists of a 15 cm x 25 cm (6″ x 10″) block of foam and plastic functioning as input device connected to a computer and a software recognizing predefined bending gestures and mapping them to standard system events. Due to financial limitations there currently is no display mounted on the block, i. e. the display of the connected computer is used for output. Of course this setup does not completely equal a mobile device, but it has the benefit that the greater processing power of a desktop computer can be used. This means that it is very easy to set up different experiments. Since the system uses standard system events, basically every application running on the computer can be controlled using the plastic block, be it written specifically for an experiment or a standard application usually controlled by mouse and keyboard. This is why it is called a prototype framework.
The work presented here will show two user studies that have been performed with the system, but more could and probably will be done in the future.

Following is a very brief summary of each chapter in this thesis to provide an overview of what to find where:

2—"Related work" lists other publications that deal with bending or deformation in general as input methods and works that provided important knowledge and impulses

that influenced the progress and development of this thesis.
3—"Design" explains what concepts, demands and constraints were identified before the implementation of the prototype framework began and what rationales are behind the various decisions that were made.
4—"Implementation" gives insight on the actual development process of the hardware and software, explaining how the different parts work together and how they can be built, used and extended.
5—"Evaluation" describes how the prototype framework itself and certain bending gestures were evaluated and what new insights in bending gestures this work contributes.
6—"Summary and future work" sums up the work so far, discusses what could be made different in following projects, where improvements to the current system could be made and what bending in general can be expected to become as an interaction method for mobile devices.

Chapter 2

Related work

There are several other publications which influenced this work. Of course the most obviously relevant ones are those that explore deformation as interaction technique, but research on new interaction gestures or technical foundations are important as well. This chapter provides a short summary on the different papers that directly had an impact on this work. Instead of alphabetically sorting the different references, they will be grouped in categories: "Deformation of physical objects as input", "Input techniques for mobile devices" and "Technical foundations". Those projects with a direct similarity to the work described here will be comparatively illustrated in table 2.1.

2.1 Deformation of physical objects as input

In [Schwesig et al., 2003] respectively [Schwesig et al., 2004] the authors propose a bendable computer. Of all related work, it is probably the most similar to this work's prototype. It however has some limitations the prototype introduced here does not have. The authors explore a mobile device that can be bent along its horizontal axis in a continuous way. It also has a non-bendable screen mounted on top and a touch pad on its backside. Experiments included evaluation of continuous and non-continuous bending ges-

tures controlling either selection, navigation in hierarchical menu structures or zooming in map navigation tasks. The authors also mention an "absence of established paradigm for interaction by deformation" [Schwesig et al., 2004, p. 4].

Balakrishnan et al. [1999] introduce a device for controlling the shape of a curve in virtual space. The so called "ShapeTape" has become commercially available and consists of a long, cable-like bendable device. Its shape is directly mapped to a virtual representation of the device allowing easy and accurate 3D modeling of curves. The paper also describes methods to further use the device's input by including buttons to one end and even integrating a foot pedal and foot mouse, e. g. to control the shape of a plane. Ways how to use the device for general command execution are also briefly discussed, but the focus lies on the 3D-modeling task area.

Scott et al. [2008] recently developed a rigid, mobile prototype device that measures forces applied to its casing. This means the device is not really deformable, but the same kind of forces that can alter the shape of the above mentioned prototypes is used as input here. The researchers implemented visual feedback that mimics bending and twisting by changing the shape of displayed windows on the device accordingly. Experiments explored the way humans can apply force to such a device, users had to acquire targets as quickly as possible by applying force as if they would either twist or bend the device. The authors found that there is a statistic significance between these two gestures and that Fitts' Law does not sufficiently model such target acquisition tasks in the context of using force as input, i. e. in their experiments closer targets were not always reached faster than targets farer away.

2.2 Input techniques for mobile devices

[Harrison et al., 1998] explores more general ways to interact with mobile devices. It focuses on using the device itself for input rather than to have additional physical objects like pens or other pointing devices. Interesting and

inspiring for the work described here were the different task oriented gestures like stroking an upper corner of a rigid, rectangular device for page-flipping. The authors embedded prototypes for three different interactions into two hardware devices and conducted user tests with them. These interactions were navigation within a book or document, navigation in long lists and annotation. In addition to the new page-flipping gestures they introduce detection of users handedness in their prototypes.

Rekimoto [1996] discusses how tilting can be used to interact with mobile devices, especially regarding tasks like map and menu navigation or navigation in 3D object views. The main idea is to use a devices orientation and rotation around its three axes for input.

2.3 Technical foundations

Kuang et al. [2002] investigate and describe optical bending sensors of the kind used in this work (see 4—"Implementation"). They give details on how to construct the sensors, how they function and how robust and accurate they are. Possible usage scenarios are also briefly touched, but none of the mentioned areas is related much to the field of human computer interaction.

Hinckley et al. [2002] propose a paradigm to help evaluate scrolling interaction. They applied their paradigm to the IBM ScrollPoint and the IntelliMouse Wheel, the first being a rate control input device, the second a position control input device. The conducted user studies systematically varied scrolling distance as well as the tolerance of scrolling. The authors found out that the position control device performs better than the rate control device for short distances and vice versa for long distances. These experiments and evaluations lead to the hypotheses that were tested in the second user study described in this work, see 5—"Evaluation".

2.4 Projects table

The following table 2.1 illustrates how this work is different from those projects previously mentioned that include a real prototype and focus on exploring bending as interaction method. The papers dealing with more general concepts or technical background needed to understand this thesis are not listed.

	GUMMI	ShapeTape	Force Sensing	This work
Display	Fixed, not flexible, mounted on top	PC display (no mobile device)	Fixed, non flexible	None, uses desktop monitor
Input Richness	Bending along one axis (1D)	Bending along one axis, but complete 3D representation of device	Two specific gesture types (bending and twisting, forward and backward)	Bending along 2 axes (18 continuous gestures)
Software Aspects	Specific software for tasks respectively experiments	Specific 3D modeling software	Mockup of Operating System behavior, specific experimental application	Software Framework to link prototype to standard Macintosh applications
Other Input	Touchpad on backside	buttons on one end, foot pedal, foot mouse	None	None
Task Focus	bending as interaction technique (zooming, menu navigation, selecting)	3D-modeling	Application switching, scrolling; general research into human ability to control force	Bending as interaction technique; allowing any experimental setup (examples for navigation and scrolling task)

Table 2.1: Projects Comparison Table

Chapter 3

Design

As was said before, the goal of this work is to research into bending as input gestures in a very general way. Thus, it was clear from the beginning that an actual prototype device had to be built with suiting software to recognize its bending state and actually perform a task. This broad approach of course has to satisfy several requirements concerning the whole system:

- It should resemble the common shape of today's mobile devices as close as possible while adapting a shape that can be bent conveniently.
- It should be bendable in different ways, i. e. it should allow many different bending gestures that can be experimentally observed.
- The actions those gestures provoke in the application controlled by the device should be easily configurable to allow experimenting with different mappings.
- The gestures themselves should provide continuous input.
- The device should have a nice feel and good grip.
- The device should be as mobile as possible.
- The recognition and action generating software should be extensible to further make the system reusable.

- The whole system, software and hardware, should be robust.

Along with these requirements there came some constraints, mostly due to limitations of time and money. Of course other constraints showed up while actually building the system, but the ones that were clear from the start are:

- Since bendable circuit boards and batteries are not standard electronics yet and building an actual bendable hardware computer or microcontroller takes up too much time, the system would have to be connected to a standard computer.
- Flexible displays are still quite expensive and not widely accessible from electronics dealers, so there would be no own display on the hardware device.
- Because the device would be built completely by hand, a certain tolerance in gesture recognition and the need to reopen it for maintenance would have to be expected.
- The unavailability of special purpose constructing hardware would probably make the device a little bigger than a usual mobile device.

The last two points had to do with the kind of sensors used to measure the device's bending, but since the decision to built them from scratch according to Kuang et al. [2002] was made quite early, these constraints were clear before the actual implementation began. The rationale for choosing the kind of sensors described in this paper are of a more technical nature and will thus be discussed in the chapter 4—"Implementation".

The first concept of the hardware device to be constructed to balance these requirements and constraints looked like the following:

- It would be a rectangular block containing several bending sensors and their wiring.

- The measurement of the sensor's values would be done by some sort of analog/digital converter (A/D converter) which then transfers the data to a Macintosh.
- This converter would not be embedded into the block, but stay near the Macintosh.
- The material of which the device consists would be foam to securely embed the sensors between two plates of plastic that protect the devices insides and allow for a flexible yet robust shell.
- The device would not have its own display but use the display of the Macintosh.

The biggest of issues with this concept was the lack of a display and the loss of mobility. Although the first problem has not yet been addressed, it turned out at a later stage of the project that a certain degree of mobility was still possible, see the description below.

The software concept that was made at that point has not changed very much over the course of the project. The general idea was to have a background process analyze the sensor data, distinguish between different bending gestures and then send system events according to a configuration file. To ensure extensibility, the language of choice would be C++ mixed with plain C. Robustness and a certain degree of platform-independency was achieved by tapping into the operating system at a relatively low level, omitting any higher level frameworks. The resulting driver for the hardware device is thus a command line tool running in user space, without much of a graphical user interface (GUI). Its different source code parts, respectively classes can be categorized by the three fields shown in figure **??**. More details about how this source code can be extended is discussed in section 4—"Implementation".

The motivation behind choosing the software to be this kind of special purpose driver rather than a contained desktop application integrating an interactive task already, was to achieve a high reusability of the prototype for future experiments. By sending standard system events for the

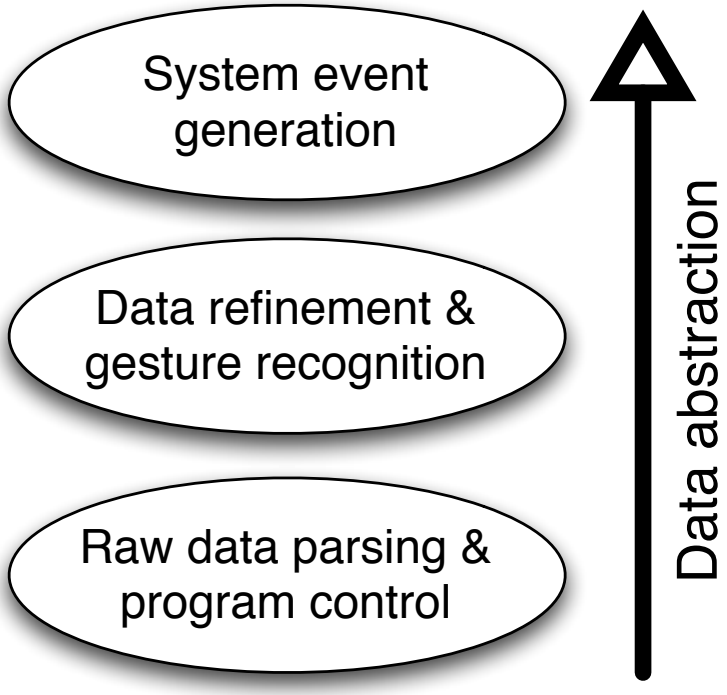

Figure 3.1: Software Concept Diagram

different gestures the prototype basically mimics standard input devices, allowing to control any application of the host operating system, in this case Mac OS X. This way, a new experiment does not necessarily involve modifying the software of the prototype. Several applications found on mobile devices, e. g. map navigation applications, are also available for desktop environments, so this setup allows using these instead of rewriting basically the same thing for the prototype. Even if there is no sufficient application, it is considered easier to quickly write one for the desktop environment instead of hardcoding it into the driver. This kind of flexibility was one major goal of this work and is the main reason why the whole project is called a prototype framework.

Open questions about the design of the hardware part of the prototype framework at this point mainly concerned the actual size of the device and the in-detail-implementation of the software under Mac OS X. To work

these last issues into the concept, a first, small sized device was built that contained just one sensor. This prototype version mainly served to get constructing experience and a first feel of how bendable the selected materials were. It was not intended to be used for user studies, but it was shown to and informally discussed with colleagues and several people from outside the laboratory. Additionally, the first parts of the software, namely the minimum code necessary to cover area one and two shown in figure 3.1 were written. Details about this first implementation will be given in chapter 4—"Implementation".

The software concept turned out quite well so far and the last parts of what the final hardware prototype would have to look like were also resolved. In the end, it was decided to make it approximately 25 cm x 15 cm (10" x 6 ") large in a landscape orientation so it could be held with both hands. Informally experimenting with one of the plastic plates of this size, later used in the prototype, suggested 18 bending gestures to be feasible for user studies. These are:

- Bending each of the devices edges back or forth (8 gestures, figure 3.2)
- Bending it back or forth along either the horizontal, vertical or the two diagonal axes (8 gestures, figure 3.3 and figure 3.4)
- Bending it into a "wave" or "S" form along the horizontal axis (2 gestures, figure 3.5)

From here on, this work will always refer to one of these when using the term gesture or bending gesture.

(BENDING) GESTURE:
Bending the prototype device in one of 18 different ways: One of its edges back or forth, along the horizontal, vertical or the diagonal axes or into a wave-like form along its horizontal axis with the wave peak either on the left or right side of the device.

Definition:
(bending) gesture

Other possible gestures, e. g. similar waves along the vertical axis, seemed too awkward to perform with the given form and material. A real twisting gesture is not contained either, because this would actually demand a stretchable

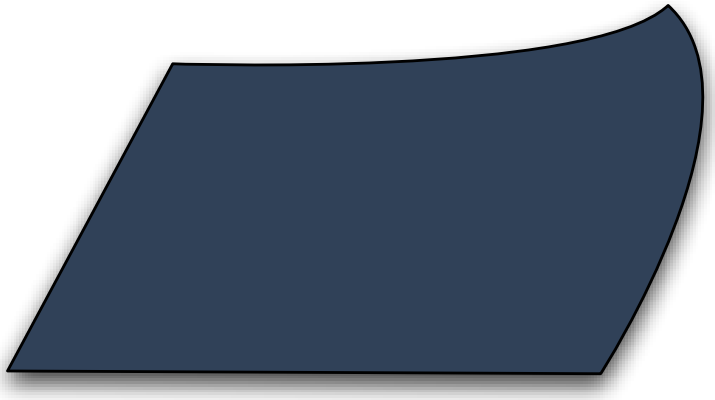

Figure 3.2: Bending one edge of the prototype

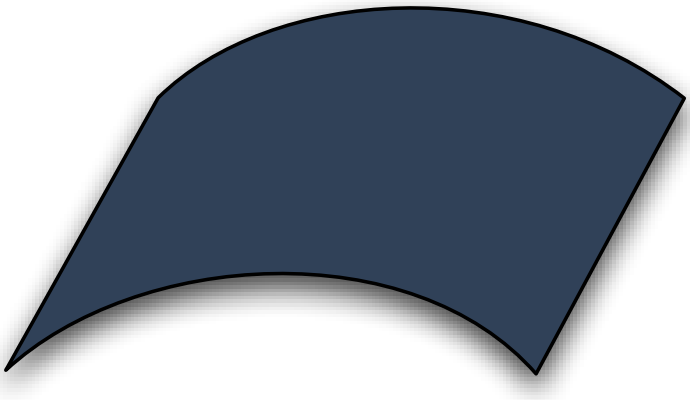

Figure 3.3: Bending the prototype along its horizontal axis

material unlike the used plastic plates. The bending along the diagonal axes, however, comes pretty close to twisting the device.

The layout and number of the sensors were a direct result of the form of the device and the gestures to be measured. Since experimenting with several layouts and sensor numbers would have meant constructing several prototype versions and due to the limited time, it was decided beforehand to better use more sensors than perhaps necessary and put them into the device in a way that clearly shows to

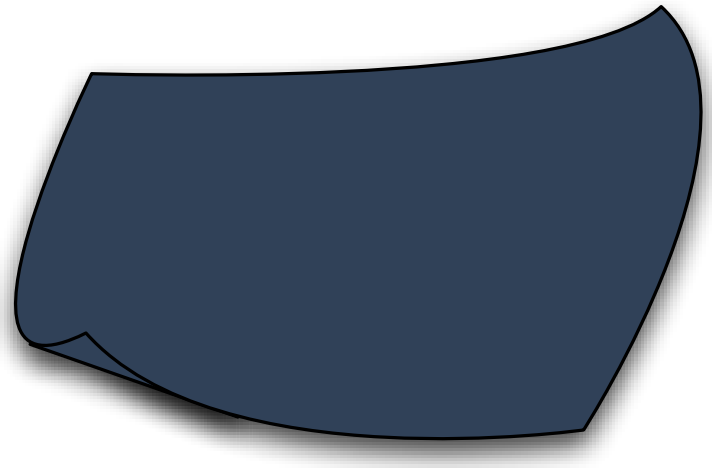

Figure 3.4: Bending the prototype along a diagonal axis

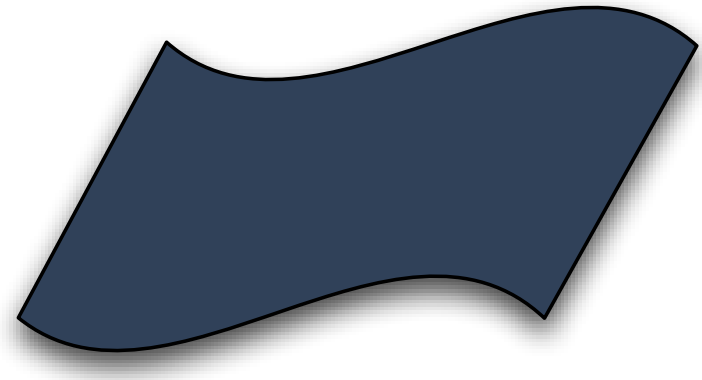

Figure 3.5: Bending the prototype into a “wave” form

fit the measurement task. This resulted in the layout shown in figure 3.6. To be able to invest as little time into the analog/digital conversion of the sensor data, an Arduino[1] microcontroller was planned to be used as A/D converter. It turned out that the only easily available model with eight analog inputs was the one connected via Bluetooth to the Mac. As was said before, this made the device a bit more mobile again, even if the controller had to hang on one side

[1] http://www.arduino.cc

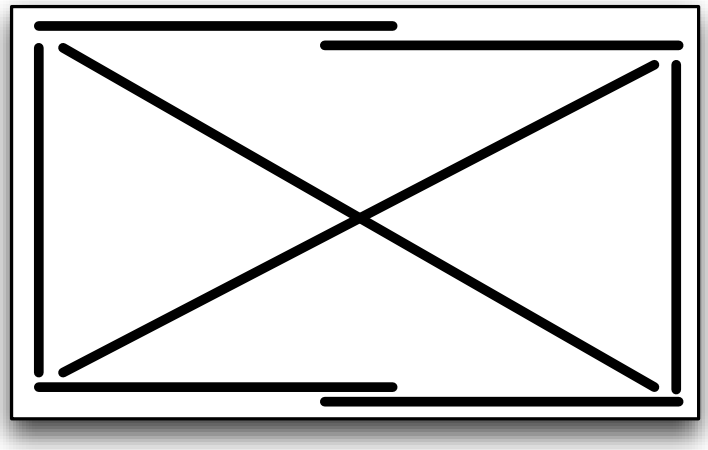

Figure 3.6: Sensor Layout

of the prototype. Currently, the power supply requires a power plug, but this could be easily adapted to the use of batteries. A sketch of the complete hardware setup can be seen in figure 3.7. The power supply is not depicted, but it is clear that the actual prototype device is not connected to the Macintosh directly. The software does not need much processing power and runs on even older laptops, so the whole system can be carried in a bag together with a host computer.

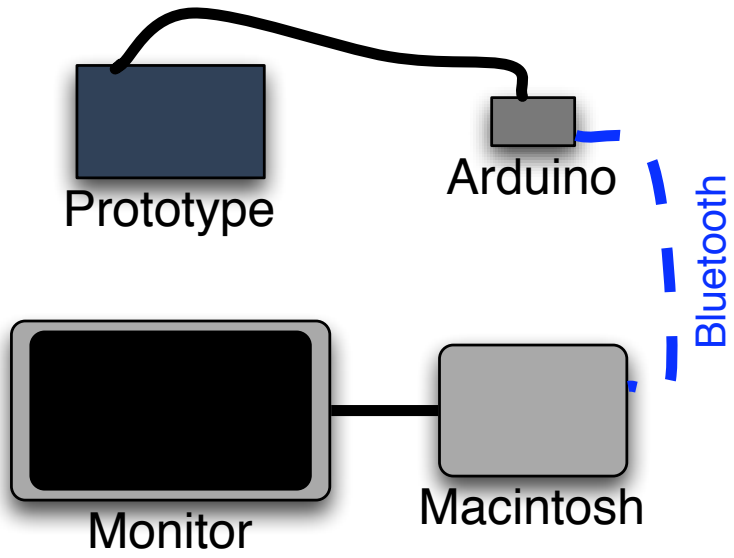

Figure 3.7: Sketch illustrating the hardware setup

Chapter 4

Implementation

The implementation was done in a very straightforward way, but it can be divided into three different parts. At first, a smaller hardware device was constructed to make sure the chosen concept actually works and to get experience at crafting everything by hand. This smaller version of the device contains only one sensor and is depicted in figure 4.1. The next part was then of course the construction of

Figure 4.1: Small Prototype

the larger, final prototype according to the plans explained in 3—"Design" and shown in figure 4.2. The third part of the implementation was the development of the software, which progressed in parallel to the two other steps.

Figure 4.2: Large Prototype

Construction of the small device went surprisingly well, so the following description will focus on the larger device and the necessary steps to construct it, referring to the smaller one only if necessary.

4.1 Hardware Construction

As was said before, the hardware devices consist of a layer of foam in between two plates of plastic. The reason for this is that any bendable physical object is also stretchable to a certain degree to compensate scissoring forces. For most materials this means the thicker they are, the harder it is to bend them respectively the easier it is to bend them, the less flexible they get. Another important aspect was that it had to be easy to embed the sensors into the material. Foam seemed perfectly suited for this. It is easy to bend, cuttable to allow sensor embedding and deformable enough to compensate scissoring forces. The downside was that once the

sensors would be put in it, foam would not protect them enough and be too soft to grip. The kind of foam used in this project also lacked some flexibility, i. e. it would not always completely return to its original shape due to its own weight. For this reason two plates of plastic were placed under respectively on top of it.

After the foam and the first, lower, layer of plastic were cut into shape, they were glued together. The next and probably most important part was crafting the sensors. Both devices use self-made bending sensors of the type described in Kuang et al. [2002]. The principle of these sensors is that a light emitting diode (LED) is connected to a photocell via a fiber cable. This cable has an abraded section where light is lost. This loss linearly varies with the degree of bending applied to this section. By building a voltage divider with the photocell and some standard resistors, this loss of light and thus the degree of bending becomes measurable. For more details on this please refer to the mentioned paper.
There were several reasons for using these sensors instead of commercially easier available and perhaps more traditional sensors like strain-gauge transducers. Discussion with colleagues who already had experimented with bending sensors showed that other kinds might have a very long response time, be less robust and only measure bending in one direction. Besides, they were more expensive than the self-made sensors. Another great advantage was that the self-made sensors would allow having more control over their actual form and length, adapting them to the needed layout more easily.

The sensor heads, i. e. everything of the sensors except the abraded fiber cable, were crafted first, according to figure 4.3. For this, the photocells together with their holder and the LEDs were each put into the end of a small plexiglass tube. To fixate and connect the fiber cable, basically by just reducing the diameter of the tube, a small piece of rubber tube was put in front of them. It is important to note that the fiber cable itself was not put between the two heads of each sensor at this point. Before that could be done, their exact positions on the foam had to be found so that the cable would have the correct length. Because of that, they were first put loosely at their final positions and the power supply cables were cut to their approximate length. Since

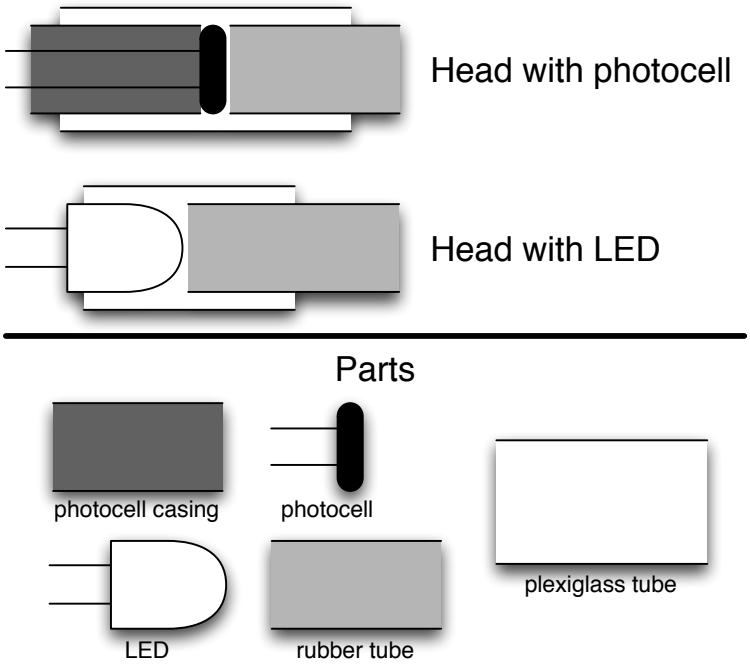

Figure 4.3: Sensor Heads Schematic

they were designed to leave the inside of the device anyway, precision was not that important, they would be cut to fit at a later time. Of course their layout inside the device was done in a way so that they have enough space to move when the prototype is bent and use as few cables as possible.
Next they were soldered to the sensor heads and stabilized with tape. Now the final thickness of each sensor head was reached and an appropriate portion of foam could be cut out from the middle layer of the device. It was important to leave a little space for the heads move a little when the device was bent. Once they were put into the foam, the missing fiber cable could be cut to the correct length, abraded at the section to measure bending at and put between the heads.
A picture showing how the final interior of the device looks like is shown in figure 4.4. Some of the sensors lie in a bit different angle than was originally intended (see figure 3.6 in 3—"Design"). This was due to the thickness of the sensor heads and the characteristics of the foam.

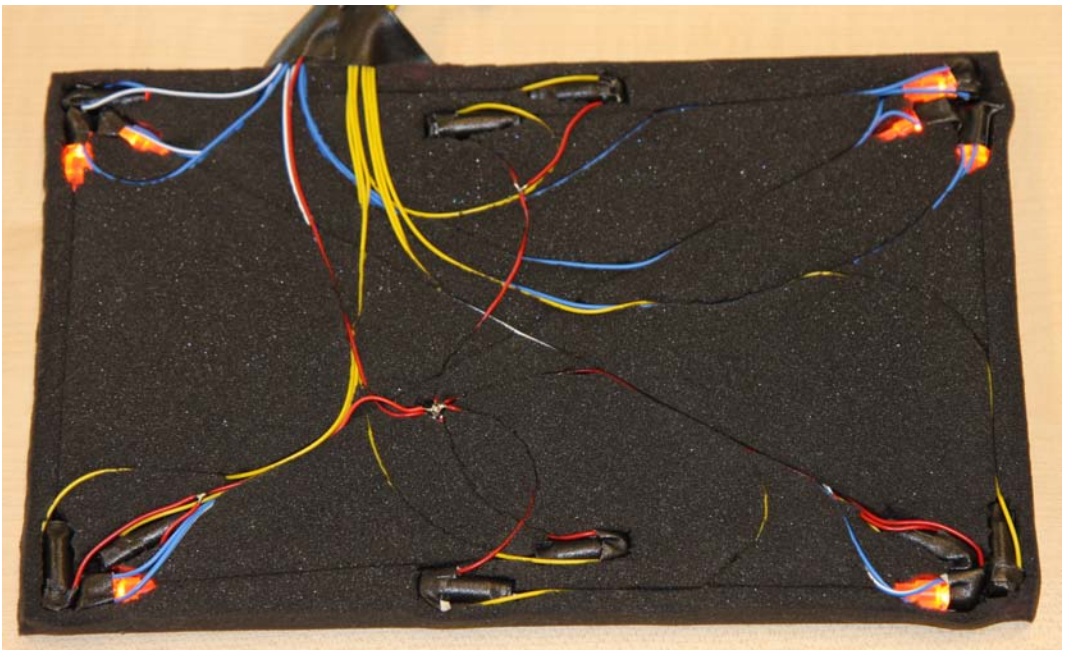

Figure 4.4: View of the larger prototype's interior

At this point it became clear that glueing the upper plastic plate onto the device would probably not be a good idea for two reasons. First, this would make later maintenance of the sensors impossible should they break. Second, the device could perhaps become too rigid. So the initial idea to completely seal the prototypes was changed, leaving the upper plate loosely on the device. To fix it on the device, the larger prototype got a bag of stretchable cloth in which it is put with the plate laying on it. This has proven to be ideal not only to effectively close the device, but also to provide a good grip and feel to the users holding the device.
Figure 4.5 shows the interior, the protective upper plastic plate and the bag holding the different parts together. The Arduino board can be seen in the upper right of the prototype.

To complete the sensors and allow measuring, resistors needed to be connected to the photocells to build a voltage divider. Common knowledge from electrical engineering dictates that these pull-down resistors should be about

Figure 4.5: Picture showing the different parts of the prototype

equally large as the average resistance of the resistor they are connected to, i. e. the photocell in this case. Additional resistors were put in front of the LEDs first.
Since photocells vary in their resistance due to their construction process and the abraded sections in the fiber cables were produced by hand, this average resistance, or better the resistance in a neutral state of the device varied quite a bit. So to determine the needed pull-down resistor for each of the eight sensors, the resistance of each photocell was measured with the LEDs on and the device not being bent, lying on a table. The voltage supply for the LEDs is 5V, their difference in light intensity is compensated by this pre-measurement for the pull-down resistors as well. To avert feedback influences, the LEDs run on a separate power circuit than the photocells and the Arduino board. The circuit and resistor layout for one sensor is examplified

in figure 4.6.

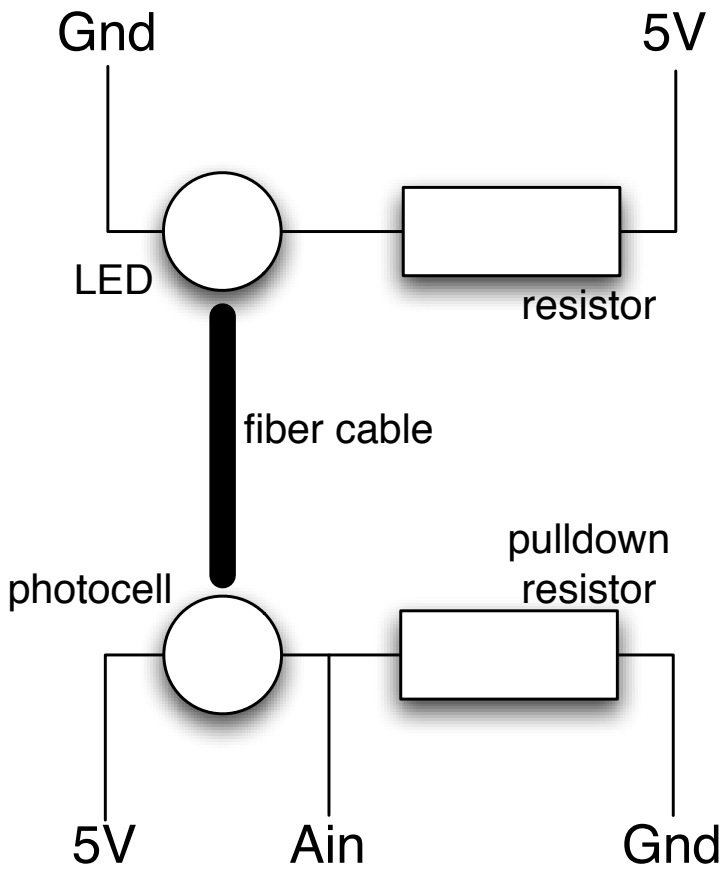

Figure 4.6: Circuit for one sensor

The last change of the hardware was done after the device along with the software were thought to be already finished. Both had been in informal use for a few days, i. e. the device had been bent several times to check if everything was working. It turned out that although there was some space left in each cutout section holding the sensors to allow them moving a bit, constant bending slowly pulled the fiber cable out of the heads of the sensors, probably due to scissoring forces. The foam was obviously strong enough to hold one head in place while the other one along with the fiber cable was pulled at due to the bending. To overcome this, the heads were glued to the fiber cable and the rubber tubes in the heads were glued to the plexiglass tube

as well. The LEDs and photocells did not need to be glued because they were irrevocably fixed to the plexiglass tube. This of course would make it impossible to replace a damaged fiber cable, but this turned out to be no problem, since none of the sensors got damaged so far.

4.2 Software Architecture

Like stated in 3—"Design" the software concept did not change in general and the code developed along with the hardware devices. The interface between the hardware, i. e. the Arduino Bluetooth board, and the software is a standard serial port, placing the whole program at a relatively low level of the operating system. Once the Bluetooth connection is set up using the normal system settings it can be started from a command line with the serial port as a parameter.

Of course the Arduino board runs a small program as well sending the measured data from the eight sensors to the Macintosh as a constant stream of colon separated integer values. The whole code can be found in the linked archive or on CD ROM.

Sources[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/mainsources.zip

Please check these files for specific code details. They contain an XCode project, but the header and source files can be viewed by any editor. The also contained Arduino code is plain C, the main program C++.

The main program on the Macintosh consists of several units refining the incoming data and ultimately converting them into standard system events. The rest of this chapter discusses the purpose of each of these units and provides information when and how it should be extended or altered.

The most basic level of the code is the `main()` routine.

It basically consists of an infinite loop constantly calling methods to poll incoming data to refine. This refinement consists of the following steps:

- Poll data from the serial port, i. e. the Arduino board
- Check whether the new measurements result in a new system event and if so, send it
- Check for control input from the keyboard to influence the behaviour of the program, e. g. quit it

Unlike the first two, the last step is not put into a separate class, because the current different control mechanisms either rely on methods that have to do with the first two steps in some way or are so simple that they do not need an additional class. Before this constant polling and checking stage is reached, the routine of course sets up the serial port and calls some other initialization methods.

The first of these, together with the polling method `readArduino()`, is contained in the class `TWDataReader`. It basically holds all functionality to interpret the strings of colon separated integers coming from the Arduino into a more abstract scale, including a calibration routine, `calibrate()`. This is very important so at later stages the data refinement may operate with more device independent values. A reasonable scale for the degree of bending seemed to be from -10.0 to 10.0 in floating point values. The calibration routine is called at program start, unless a previously saved file with calibration data is passed to the command line utility as a parameter, or by a direct control input character (c). It requires the user to do a measurement with the device in neutral position and then bending it in a way that all sensors are bent to their maximum and minimum bending degree. This way, offset, minimum and maximum values for all sensors are collected, allowing the class to convert all future readings into values between -10.0 and 10.0. Usually this calibration is done by the experimenter when setting up a new experiment and not by users themselves. The rest of this class's methods should be self-explanatory, please refer to the source files themselves.

Checking new input for changes and deciding whether they result in a new system event and if so, in which event, is the purpose of the `TWEventDriver` class. Like the `TWDataReader` class it has a method that periodically gets called by the run loop, `sendNecessaryEvents()`. This method gets the most current values from the `TWDataReader` instance and compares them with the last values, deciding whether any system event has to be created. This decision is one of the more complex parts of the program and will be described in the next paragraph.
The class holds an array of objects each representing an action that is performed on one of the specific gestures the device is designed for. Currently, these actions are mutually exclusive and the objects performing them in the end invoke different kinds of standard system events like scroll wheel events or keystrokes. An additional method, called during the setup at program start, parses a configuration file and passes different of its sections to each of these 18 objects to set up their behavior. The classes defining these objects are explained later.

Before any of the action representing objects' methods can be invoked, the `TWEventDriver` has to decide which gesture, if any, was performed. To understand how this is done, one needs to know how a bending gesture, or rather a bending state, is internally represented in the program. Since the device delivers eight sensor values and each sensor linearly changes its value with the degree it is bent to, any of the 18 different bending gestures in any degree of bending is represented by a vector of eight values. After these values have been pre-processed by `TWDataReader`, they lie between -10.0 and 10.0, thus:

Definition:
bending state

BENDING STATE:
An eight-dimensional vector of floats between -10.0 and 10.0 representing the state of bending of the hardware device.

Each of the different bending gestures defines a sub-space of the vector space representing all possible bending states. E. g. the zero element in this space represents the "neutral gesture", i. e. the neutral state of the device when it is not bent at all. Note that each of the 18 different gestures does not simply equal one point in this space, but a whole area. Which point exactly it is depends on the degree of bending

of that gesture. Thus, the problem to decide which of the eighteen gestures a given input vector represents is a clustering task.

To solve this problem the project employes a k-means clustering algorithm (see e. g. Wikipedia[1] for a brief introduction of this algorithm and links to visualizations). This algorithm clusters n objects into k clusters with k a given number and $k ¡ n$. Starting with initial center vectors for each cluster, it performs the following steps:

1. For each object, calculate its distance to each center and assign it to the one it is nearest to, effectively clustering the objects.
2. Calculate the mean of all objects in each cluster and use it as new center.
3. If the distance between old and new center is not below a given bound, repeat the procedure with the new centers.

Figure 4.7 depicts how it works. The initial centers of the clusters are usually randomized, but this is not necessary in this case. Due to the layout of the sensors it is obvious which coordinates of the bending state change with which gesture, allowing to estimate a good initial center for each cluster. In fact, the initial centers used in the software are hardcoded in the header `TWIncludesAndCalibrators.h`. Note that internally, there are 19 and not just 18 clusters. This is to ensure that when the device is not bend, no gesture gets accidentally recognized as being performed. The rest of the algorithms implementation is contained in the files with the `TWKMeans` prefix. They define a data structure for holding vectors representing the space of bending gestures as well as the actual functions to calculate the final cluster centers. At program start, right after the calibration, the user or experimenter has to perform all 18 different gestures to provide data to cluster for the k-means algorithm. The result of running it then is an array of 19 cluster centers, each representing one of the 18 bending gestures plus the neutral device state.

[1] http://en.wikipedia.org/wiki/K-means_algorithm

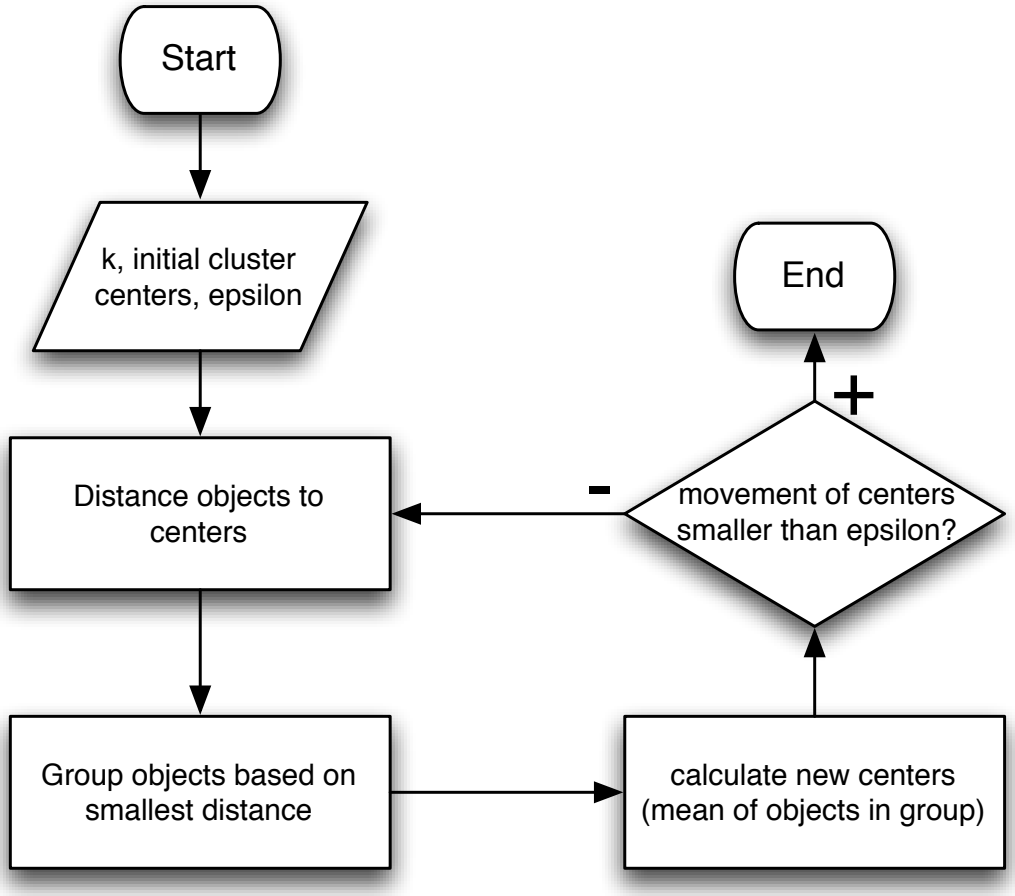

Figure 4.7: K-Means Algorithm Chart Diagram

These can be saved at program end in a file to be used in future program runs. The methods for this are also provided in the `TWEventDriver` class. The file has to be passed as a command line parameter to the utility to be used.

On program execution, a new input is assigned to a gesture by simply determining which center is the nearest. The neutral cluster, however is specially treated. If it were assigned to be the correct cluster for a given input, users would have to bend more than half of the maximum degree of bending for each gesture to actually invoke an action, because only then the distance to an actual gesture cluster would be smaller than the zero cluster. This is overcome by simply looking at the second smallest distance to a cluster.

To prohibit false positives, the distance to the zero cluster must be larger than a predefined minimum, hardcoded in the header `TWIncludesAndCalibrators.h`. This effectively reduces the "size" of the zero cluster. Practical use has shown that this is the best way to solve the problem.

Another issue was that some gestures seemed to be very near to two clusters at the same time, especially gestures with a low bending degree. Two facts made the system then jump between recognizing each gesture after some passes of the run loop. First, humans can probably not hold a bending state absolutely still. Second, even if they could, the material itself would move on a very low scale, because the foam as well as the plastic and the sensors slowly adapt to the form they are bent to and relax. To solve this issue a final state machine with a Schmitt Trigger was realized in the `TWGestureFSM` class (`TWEventDriver` integrates an instance of this class). Since it was impossible to reach certain states without first passing at least the neutral state, this eliminated impossible state transitions. The Schmitt Trigger further improved this, because it basically weights a transition resulting in the following behavior: Once the system had decided on a certain cluster, it would not change to another one, unless the distance was not just smaller than the distance to the current cluster, but smaller by a certain factor.

Up to here, the whole software system is more or less platform independant, i. e. although it has not yet been tested on a different operating system than Mac OS X it does not use any immanently platform dependent libraries. The last part of the program in contrast to this is of course very platform dependent, because it is responsible for system event dispatch. To allow an easy integration with the rest of the program and extensibility of the different kinds of actions, it is realized via a special purpose class structure. Any kind of action that is desired to be associated with one of the 18 different bending gestures needs its own class, implemented so far are keyboard strokes, scroll wheel events, Apple Script events and mouse movements in four directions.
Each of the responsible classes is a subclass of the abstract class TWEvent, adapting a provided interface. This interface consists of the methods `sendSysEvent()`,

`resetToNeutral()` and `setUpEventFrom()`. The first method for an instance of the `TWEvent` subclass associated with a gesture is called every time the `TWEventDriver` recognizes the current input as belonging to that gesture. It gets passed an always positive degree of bending which `TWEventDriver` calculates from the eight vector coordinates. Note that it gets called on each pass of the run loop and not just once so continuous events can be implemented. It is up to the `TWEvent` subclass to make sure the method acts only once if a non continuous action is desired. The `resetToNeutral()` method is called on each pass of the run loop for each object associated with a gesture that is currently not recognized as being performed. The last method, `setUpEventFrom()` gets a string passed by `TWEventDriver` that was parsed from the event setup file to initialize the concrete instance of a `TWEvent` subclass.

Currently the `TWEventDriver` class still needs to know all the classes that are to be used for actions, because it has to initialize its member variables of the type `TWEvent*` with concrete objects. Nevertheless, this setup allows the class to then call the three methods described above on these without actually knowing of what type they are. It also makes design of new `TWEvent` subclasses very easy.

As was mentioned above, `TWEventDriver` expects to parse a configuration file to set up the instances of the different `TWEvent` subclasses. This file, currently expected to be named `eventsetup.txt`, contains sections for each of the 18 gestures such a subclass can be associated with. It is supposed to be properly set up by an experimenter before the command line utility is executed. Each of the 18 sections consists of one line describing the name of the event type and a number of additional lines to configure this specific event. `TWEventDriver` needs to know the name of the event type to initialize the correct subclass of `TWEvent`. The other lines then are passed to the new instance via the `setUpEventFrom()` method.

Chapter 5

Evaluation

So far this work has described a prototype framework to research bending as interaction gesture for mobile devices. The following chapter shows two user studies using this framework to evaluate certain bending gestures in concrete tasks. The first user study mainly focuses on the usefulness of the software and hardware itself, but already gives insight into how users experience bending in general as well. The second study explores how bending can improve interaction in tasks commonly encountered on mobile devices by comparing it to traditional input methods in a formal efficiency analysis.
Finally this chapter will look at the requirements identified in 3—"Design" and evaluate how well the prototype satisfies them.

5.1 The First User Study

The first user study done with the prototype framework served the purpose to check whether the system works as planned and to give a first insight on how users experience bending as interaction gesture.

5.1.1 Task & Setup

Because it was designed to examine bending gestures in the context of mobile devices, the task users should have to complete was supposed to be one also found on mobile devices. One area that is becoming more important in this context is map navigation, or rather navigation in a virtual space in general. Schwesig et al. [2004] examined this as well, but they mainly focused on using the bending for zooming. The study described here finally chose to let users navigate a character in the 3D-environment of a computer game, World of Warcraft[1] . The decision was made because the game controls are very easy to configure and thus allow a very quick adaption for use with the prototype. Moving a character through a 3D environment was assumed to be similar enough to navigating a map or any kind of view, because it basically just moves the viewport over a planar field, the ground in a 3D space. The game controls and the prototype system were configured to allow the following gesture to action mapping:

- Bending the device away from the user, i. e. into a lens-like form along the horizontal axis, moved the character forward.
- Bending it the other way round moved the character backward.
- Bending the devices upper left edge towards the user turned the character to the right (the camera moved to the left around him).
- Bending the same edge away from the user turned the character to the left (the camera moved to the right around him).
- The upper right edge functioned accordingly (bending it away turned the character right, etc.).
- Bending the device into a "wave" form along its horizontal axis, with the "peak of the wave" on the right device side made the character run a left curve.

[1] http://www.worldofwarcraft.com

- Bending it into a mirrored form made the character run a right curve.

On the game's side, this equaled pressing the arrow keys on the keyboard: "Up" to move forward, "Down" to move backward, "Left" to turn left, "Right" to turn right and the combinations of "Up" and "Left" respectively "Right" to move in a left or right curve. The gestures were directly mapped to these key presses, i. e. no continuous gestures were used in this experiment.

The rationale behind using the "wave" bending gestures in the described way was that when bending the device into these gestures one turns the hands into the direction the character moves, like figure 5.1 shows.

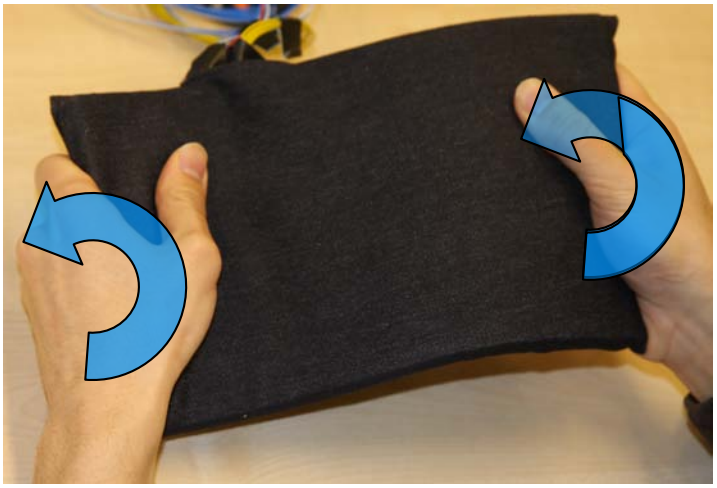

Figure 5.1: Hand movement for the "wave" gesture

5.1.2 Methodology & Hypotheses

The actual goal users had to accomplish in the experiment was to maneuver the character along a certain route in the game world as fast as they could. They had to do so using the prototype in one run and using the keyboard in another. The route was the same for both input methods and

for all users; it was chosen in a way that they had to use left and right movements and could easily orient themselves from surrounding structures in the game.The experimenter stopped the time they needed with both input devices.

Before both runs users had the chance to get familiar with each device. Because the interaction with the prototype was something completely new to each user, the experimenter gave a demonstration how to use the device and explained the different gestures and how they controlled the game. He also showed them the path to follow in the task beforehand in this demonstration.

Other than to see how the prototype works in regards of the goals described in 3—"Design", there were the following hypotheses hopefully to be proven in the experiment:

H1: There would be a statistical significance between the average time users needed to accomplish the task with the prototype and the keyboard and the keyboard would perform better.

H2: Users would prefer the input method that lets them accomplish the task quicker.

To verify hypothesis H2, users were asked to mark what input device they had liked better to accomplish the given task with on a questionnaire after the experiment. They were also asked to answer the question "I could imagine that bendable devices will be used in the future" on a five-point Likert scale and they were encouraged to give comments on their own.

For a better analysis of the experiment later on, the users were filmed from an angle that allowed to see the screen and the prototype. None of the participants felt uncomfortable with this, nevertheless they were assured that this footage would not be published in written form.

To counter learning effects in regards to the task, six users started with the prototype and five with the keyboard. Time was stopped manually with a stop watch.

While conducting the experiment users were seated in an office chair approximately two and a half meters away from a 40" display where they could see the game environment. A table was positioned in front of them where they could rest their hands while holding the prototype or using the keyboard if they wished to do so.

After the experiment, each user was given a piece of candy worth about 50 Eurocents.

5.1.3 Participants

Eleven users took part in the experiment. Nine of them were male, two female. Eight worked at the chair and except one, all had heard of the project before, but none had ever used it. The average age was around 32, but nine users were in the age group 20 to 29. The reason for the high average age is that the other two were aged between 50 and 59 respectively between 60 and 69. These two were also the females taking part in the experiment and they were the ones with the least experience with computers in general (they were briefly asked before the experiment); the others had much experience, since they work with computers every day.

5.1.4 Results

As expected, almost all users were faster with the keyboard. The measured times, preference and answer on the Likert scale can be found in table 5.1. Regarding the Likert column, a 1 here represents the answer "does not apply at all", a 2 represents "applies a little" and so on.

The mean time over all users for the prototype was approximately 56.3 seconds, for the keyboard approximately 37.8 seconds. The user slowest with the keyboard was the one older than 60, the user slowest with the prototype was the one older than 50. Both of them explained they had trouble with the orientation in the 3D environment of the game

Preferred	Likert	Time Prototype	Time Keyboard
Prototype	4	51.6 s	25.6 s
Prototype	5	75.4 s	125 s
Prototype	5	152.6 s	31.8 s
Keyboard	5	26.5 s	49.6 s
Keyboard	4	56.1 s	26 s
Prototype	3	75.5 s	26.8 s
Keyboard	5	36.5 s	26.4 s
Keyboard	4	34.6 s	26.4 s
Keyboard	4	33.9 s	25 s
Keyboard	4	43.5 s	26.9 s
Keyboard	5	32.8 s	26.2 s

Table 5.1: Data measured in the first user study

during the task, although they had an extra long familiarization phase.

5.1.5 Conclusions

It was surprising to see that three of the four users who had preferred the prototype over the keyboard did so in spite of being slower when controlling the game by bending. One possible explanation could be that they tried to be friendly and gave the answer they believed the experimenter wanted. Their comments, however, hint in another direction. One said it was more intuitive and one it was interesting because it was new. In fact, the oral feedback from all users was very positive. Those who preferred the keyboard said they mainly did so because of drawbacks in the prototype. Most thought it was a bit too stiff and that they wished for a continuous input when using a continuous gesture like bending. Nobody stated to have preferred one input method simply because it was faster to achieve the goal. One user mentioned he would like the idea very much to have something to "squish" when being engaged in a game. A possible explanation for this apparently lacking connection between preference and speed could be that neither the game, nor the experiment somehow encouraged being quick, e. g. by rewarding the user for quickness.

All in all, it seems that hypothesis H2 can not be accepted. It appears that the task itself also has an influence on what input method is preferred, but this would have to be explored in future experiments.

Regarding hypothesis H1, the results were surprising as well. A Student's t-test was done and resulted in a p-value of approximately 0.0914, so there was no statistical significance between the measured values and hypothesis H1 had to be rejected. If one takes into account that the two users with extremely high times in their first run stated to have problems with orientation in the 3D environment it seems justified to ignore their values and eliminate them from the study. The remaining nine users result in a p-value of approximately 0.0281, which means hypothesis H1 can be accepted in this case. Of course this influences the study quite a bit, because all remaining users are male, between 20 and 29 years old and have much experience with computers in general.

Other valuable conclusions were made about the prototype itself. All in all, it worked just the way it was supposed to. Most users said it was a bit too stiff and thus hard to bend, but due to limitations in time it was not possible to build a second, easier bendable one before the last study. Using keystrokes, i. e. non-continuous gestures, as system events induced by bending the prototype appeared to be problematical. The reason for this was that usually users tended to bend the prototype slightly more than was needed to "press the key". When they then wanted to "release the key", it took them unexpectedly longer to do so than compared to when they do the same on a keyboard, because they usually slowly released the gesture on the device instead of letting it "snap" back to its neutral position. This probably means that keystrokes would better be used in a manner equal to a "keypress" instead of a continuous "keydown". The prototype framework is already able to do this by setting up the configuration file differently, but a more specialized subclass for an event that implements a threshold, e. g. , would be an even better way.
One thing that can not be easily done with the current system is a merging of gestures. Unlike with the keyboard, where running a curve is basically the merging of the forward and left or right gesture, i. e. pressing both keys at

once, the prototype is not able to allow a combination like this. E. g. if it is bent along its horizontal axis already, bending one of its corners is not recognized by the software. Of course this is currently only a software and not a hardware problem, the sensors do deliver other values if a corner is bent as well. Until this is changed, experiments that might suggest users to try to combine gestures should be avoided.

5.2 The Second User Study

The second user study was meant to give new insight on two specific bending gestures used for a common and important task in any mobile device: scrolling. Since information is often represented in a list and the screen size on a mobile device is limited, this is something that has to be addressed. Users spend much time in navigating text or menus in search of a specific paragraph or item, so scrolling should be realized in a convenient way. Although there are other methods to order or represent information, an unordered list is used in many cases, sometimes it is an inherent quality of the information in the first place. Text is, in the end, an unordered list of characters respectively lines or pages and people frequently skim it to get an overview or find a specific information like a chapter or image on a page.
The study's other goal was to find out how intuitive the gestures and the prototype itself were and to get first hints for natural mappings of gestures and certain actions encountered on a mobile device.

5.2.1 Task & Setup

In the experiment to examine how well it was possible to use a bending gesture for scrolling, the prototype framework was set up to directly invoke mouse wheel scrolling events in the system. To compare it to other input methods, a Wacom[2] tablet display was connected to the host

[2] http://www.wacom.com

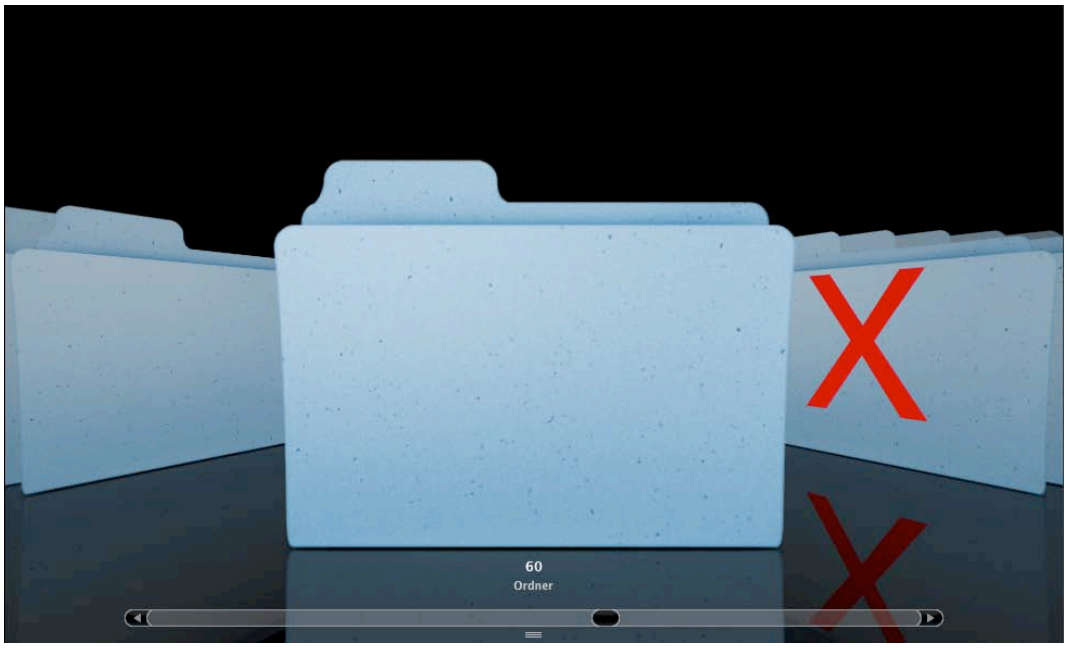

Figure 5.2: Cover Flow view of the Mac OSX Finder

computer that provided a pen to navigate the mouse cursor. Such a pen is often found in mobile devices, e. g. in PDAs. A second comparison was to be made to a jog & shuttle control device, the ShuttleXpress[3] (in the following referred to as shuttle). The pen is a position based input device where the shuttle is a rate based input device that influences the velocity of the scrolling. The prototype was set up in a similar way, i. e. the degree of bending a gesture was directly changing the velocity of the scrolling.

The concrete task users had to accomplish was scrolling vertically through a list of 100 icons displayed on the screen until they encountered one icon that was marked with a big, red x. The application responsible for displaying the icons was the MacOS X Finder. It was set up to display 100 folder icons in the cover flow view. This view looks like shown in figure 5.2. The currently selected icon is the frontmost. If the user scrolls to another position, the icons "flip" to the left or right, depending on the direction of the scroll (the Finder supports vertical scrolling devices, but up also equals left and down equals right in this case). The flipping

[3] http://www.contourdesign.com

icons are smoothly animated and it is easily possible to spot the one folder icon marked with an x while they pass by, of course it gets harder with increasing scrolling speed. There are other parts of the cover flow view of a Finder window that are not displayed in figure 5.2, but they were covered with paper attached to the screen in the experiment. The visible area only showed the animated part and was of the same size as the prototype (15 cm x 15 cm or 6" x 10"). As can be seen, there is a scroll bar under the icons. This was used with the pen.

The gestures used in the experiment were the two "waves" to scroll to the left and right and bending the upper two edges to the front described in 3—"Design". The "wave" with the peak on the right was set up to scroll to the right, the other one to scroll to the left. The edges were supposed to flip a single icon (right edge next icon, left edge previous icon).
The rationale behind this was that this mimics the way in which users would flip pages in a book, either several at once or one by one. Harrison et al. [1998] proposed a similar approach regarding the edges in their system. The shuttle was configured in a similar way, its outer ring did the scrolling and the most left and most right buttons flipped to he previous respectively next icon. The pen also provided a mechanism to flip a single icon by clicking on the softbuttons at the right and left sides of the scrollbar.

There occurred one problem when setting up the edges of the prototype. Originally, it was intended to mimic an arrow down or arrow up key press, like the shuttle was configured to. Unfortunately, it turned out that the prototype could send this event very quickly after the last scroll event it produced. This lead to a strange behavior, because of how the cover flow works. Under the actual animated icons, the window displays a second, vertical list of smaller versions of the icons. If a user scrolls through the cover flow and stops at a certain icon, the frontmost icon in the cover flow gets selected in the list below. The arrow keys, however, basically operate on the lower list alone. If a key-press switches the selection to a neighbor icon, the cover flow does switch as well, but if this key-press happens right after a scroll event, but before the cover flow could update the lower list, not the icon next to the one the scrolling stopped

at, but the one next to the currently selected icon gets selected. This then makes the cover flow move back to highlight the currently selected icon from the lower list. This happened only with the prototype, because it could send events very quickly after a scroll event, if a user alters from the "wave" bending gesture into an edge bending gesture to be exact.
To counter this phenomenon, the edges were set up to also invoke scroll events, but their amount of scrolled units and their latency was scaled to such a low degree, that it resulted in switching only one icon. The drawback was that there now was a relatively long time between two edge bending gestures, i. e. it was not possible anymore to send several edge gesture events after another rapidly like with the shuttle.

5.2.2 Methodology & Hypotheses

Before the experiment started, the experimenter prepared three groups of ten different runs. The positions of the x in each of these runs was randomly determined and five of them were supposed to be scrolled to from the left and five from the right. None of the determined positions was in the first or last 20 icons. Users were supposed to each test one device per group of runs and the users were grouped as well so that an equal number of users would test the same device in a given group of runs.

The rationale behind this on the first view weird setup was to eliminate factors like individual scrolling abilities of users and learning effects. In the end, the different distances should be regarded and compared between the devices, see [Hinckley et al., 2002].

Before users started with the task the experimenter welcomed them and explained how the experiment would be like in general. He did not tell them about the specific devices yet, only that the users would have to accomplish a task with different input devices and that the time they needed would be measured. Before these measurements would start, he would have a short informal interview with them and take notes.

The idea behind this was to see whether the users would naturally associate certain actions with bending gestures. Because the project had been known around the chair and the experimenter's friends, it was unclear who had heard of the different ideas for bending gestures already. Thus, the first questions aimed to find out what the users had heard already. If they were too familiar with the experimenter's own ideas, the rest of questions would be skipped and the actual task would begin.

During this task, the users were instructed to scroll as fast as possible to the icon marked with the red x. The pen and the shuttle were explained to them beforehand. Depending on how the first questions of the interview were answered, they were told to try and figure out how to scroll with the prototype on their own while the experimenter took notes. Users were allowed to get familiar and train with each device in a demonstration run before the actual measurements began.

After the experiments the participants were thanked and offered a small candy.

Several hypotheses were made that the experiment could hopefully verify:

H1: The shuttle would perform best of all three devices, i. e. the times measured with the users using the shuttle would be significantly smaller than the other two devices.

H2: The prototype would be significantly better than the pen, but worse than the shuttle for long distances

H3: The users would find the edge bending gestures natural and in most cases figure them out on their own.

H4: The users would find the "wave" bending gestures natural and in most cases figure them out on their own.

The first hypothesis was backed up by the works of Hinckley et al. [2002] that suggest rate based devices to perform better on long and position based devices to perform better on short scrolling distances. Since the shuttle was set up with two buttons allowing a non-rate based correction

of overshooting the target, it was assumed to perform good even for short distances. The prototype was set up like this as well, but the above described problems suggested not to expect it to do better as the pen on short distances.

5.2.3 Participants

This time 15 people were recruited, mostly from outside the chair. All but three were in the age group between 20 and all but one worked with a computer on a daily basis. One user was in the age group of 30 to 39, one in the group 40 to 49 and one user in the group 60 to 69. Four were female, the rest male.

5.2.4 Results

The complete measurements of the study can be found in B—"Results of User Studies" and on CD ROM. In general, the measured data appears to be very noisy. The times for the shuttle on average do not become bigger with increasing distance and there are a lot of outliers. Several users repeatedly managed to be faster at longer then at shorter distances. Even looking at the times with the pen and the prototype this phenomenon can be found, although the average times over the five users testing the different runs with these devices slightly increase.

The mean times and variances for each distance with the different devices can be found in table 5.2. Keep in mind that not all users delivered values for each distance. Also, during evaluation of the experiment it was found that the distances 69, 73 and 70 accidentally were used more than once (73 three times, the others two times).

Concerning the interview, 13 of the 15 persons were unfamiliar enough to let them try to find out which bending gestures would lead to the scrolling and single flipping actions in the task. Only one did not manage to find the edge bending gestures in a couple of seconds. One of the 12 that did find it mentioned he would have expected the lower

edges to work the same way, but not the upper ones.
Only three participants found out how to scroll with the prototype on their own. Five of the others said they had expected a complete bending along the horizontal axis to do the scrolling, bending the prototype away from them to scroll up respectively to the left and bending it towards them to scroll in the other direction. The rest also had this idea when pointed to the book metaphor, i. e. when the experimenter asked them to imagine how they would flip several pages in a book. After explaining people also sometimes flip pages while bending a book or magazine into a "wave" form instead of just bending it completely along one axis all participants understood the mapping between this gesture and the scrolling action.

5.2.5 Conclusions

The interviews respectively observations of the users that had not heard of the project in detail strongly suggest that the chosen gestures to flip a single icon or element from the list are intuitive and natural. Almost all of the participants anticipated this single gesture on their own. It was not always clear whether users found the gesture because of them having in mind the metaphor of a book or magazine or whether it was just a gesture so simple it was tried out as one of the first. All of them agreed that it was easy to remember and fitting the book metaphor after the experimenter mentioned it. Due to the problems of the experimental setup concerning the edges these gestures were not used very frequently or effectively, but these first observations suggest to use this gesture to action mapping when designing an interface for a bendable mobile device.
The "wave" gesture showed to be less intuitive. Although users understood the rationale behind it after it was explained to them, most of them did not find it out on their own. A possible explanation for this might be that the prototype was too stiff to create the impression of even being meant to be bent in this specific way, i. e. its material just did not afford to be bent like this. It could also be that the whole concept of bending was too foreign for users to think about more complex bending gestures or that they were afraid to break the device in spite of being told that it

is robust. The fact that many users believed the horizontal bending to better fit the book metaphor suggests to explore this gesture in the same context as well. See 6—"Summary and future work" for details on this.

The interviews and observations in the end lead to the acceptance of hypothesis H3, but the rejection of hypothesis H4.

Regarding the other two hypotheses the measurements are of interest. Unfortunately, there is almost never a statistical significance between the devices in the different distance runs. For the prototype and the pen, only the distance 42 delivers statistically significant values with a p-value of approximately 0.012. For the prototype and the shuttle, only the distances 55, 62, 75 and 79 (the first listed in table 5.2) result in p-values below 0.05 (approximately 0.032, 0.017, 0.033 and 0.039). The explanation for this is most probably the strong randomization of other independent variables like the knowledge of the users and the learning effects. It appears that the number of participants was too small to properly counter these effects in such a way.

The graph of the measurements is shown in figure 5.3. The distances that occurred more than once are averaged. A red circle marks statistically significant runs for the prototype and the pen, black circles denote statistically significant runs for the prototype and the shuttle.
The graph illustrates the high noise in the data, it is hard to see a trend. For this reason, several linear regressions were tested (linear, logarithmic, exponential and polynomial of second degree), the ones with the best regression factor, the polynomial regressions, are printed in the graph as well. These lines hint that the shuttle is indeed the device performing the best, but they also show that the prototype is not better than the pen but worse. However, it is also visible that the prototype almost linearly increases with growing distances. In contrast, the pen appears to grow non linearly and in a way that it would cross the graph of the prototype at a distance larger than the largest depicted in the graph. This could mean that the chosen distances are too small in general to deliver good results, although the scrolling was done over several screen sizes already.

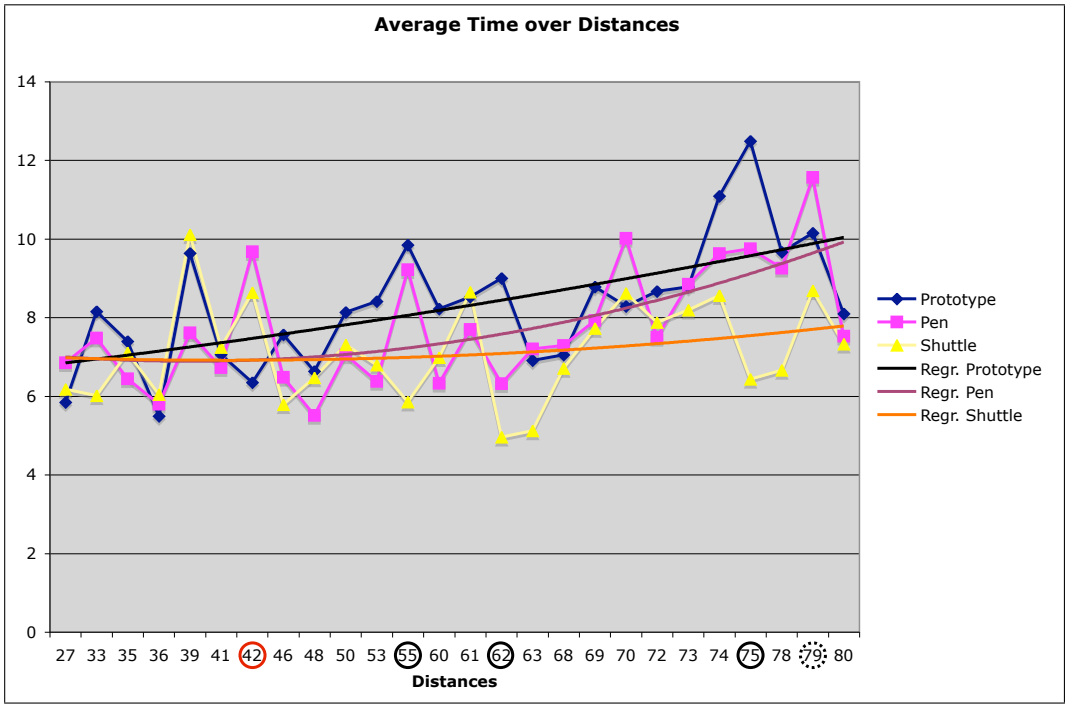

Figure 5.3: Graph of user study two measurements

As was said it turned out that the strong randomization of other variables than the devices did not turn out well. It is very well possible that for a given set of runs only very skilled users coincidentally were assigned to one device and users less skilled to another. To try to eliminate this influence of a participants skill in scrolling, the measured values were normalized according to the following formula:

```
normalized Time = (Time - Average Time) /
                  Variance
```

The thought behind this is that a user's skill in scrolling anti-proportionally correlates with the variance in that user's data set. The resulting graph is shown in figure 5.4. The regressions for each device still suggest that the distances were not long enough for the prototype to deliver better results than the pen. It is interesting to see that the runs with statistically significant values are not the same as in the not normalized value set (significant values of

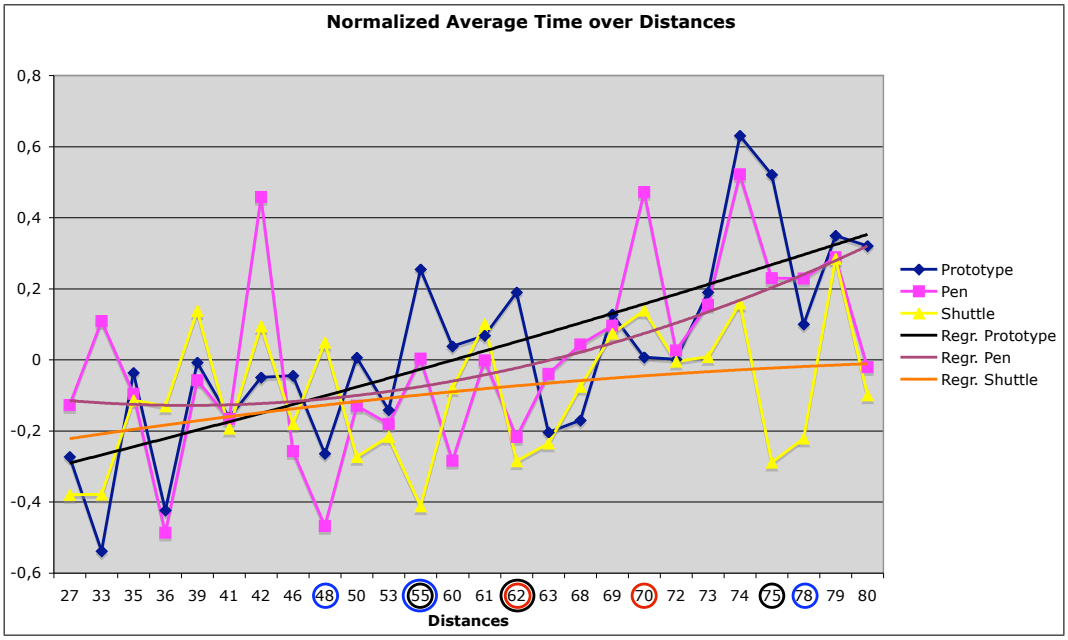

Figure 5.4: Normalized graph of user study two measurements

the pen and shuttle are marked with blue circles). This suggests that even the normalized data contains too much noise from other variables than the devices. A last try was to eliminate the randomization of the scrolling direction and only look at those runs in which users would scroll from left to right to find the target. These are illustrated in figure 5.5. The conclusions are not very different from the ones already found. It is interesting to see that in all graphs there are runs with short distances in which the prototype outperforms the pen and sometimes even the shuttle.

Nevertheless, the lack of many statistically significant times suggest that the study was not successful in regards to hypotheses H1 and H2. Both have to be rejected. The strong randomization of so many factors showed to be not applicable to such a small number of users per device.
The most valuable learnings were found in the interviews with the users. The edge gestures were learned very well and appear to be applicable to the context of navigating in lists. A concrete example where this could be put to praxis would be an eBook reader, a mobile device to display digital books. The information learned about the “wave” form bending gesture was also inspiring. Since many users told

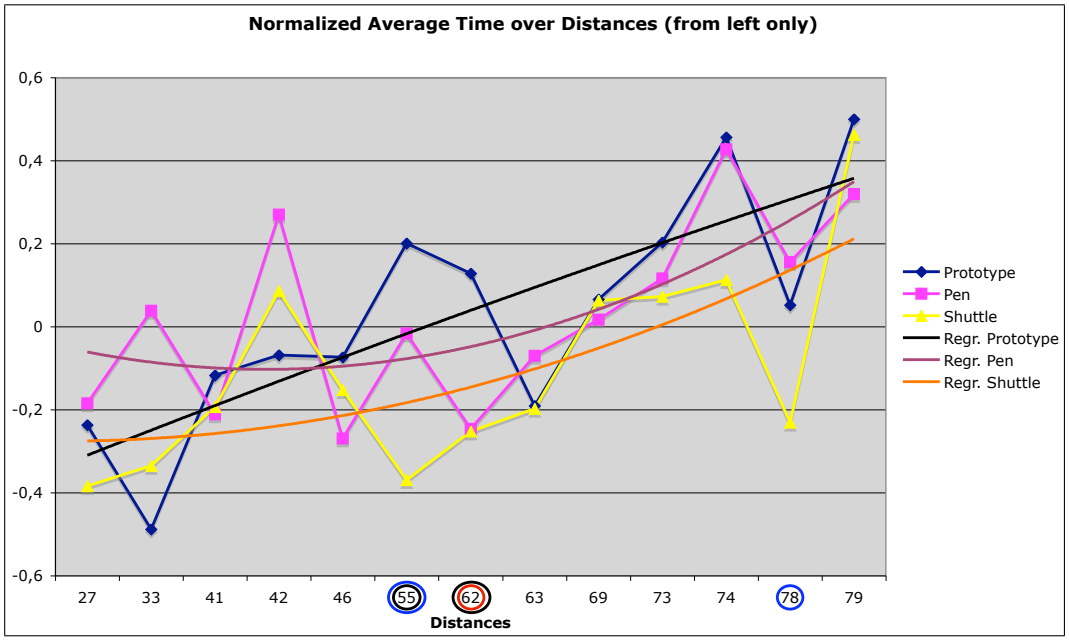

Figure 5.5: Normalized graph of user study two measurements (left)

they would expect the simpler gesture of bending the device along the horizontal axis to be more intuitive, future work should concentrate on this gesture, since it seems it can be as intuitive for scrolling as the edges were intuitive for progressing item after item in a list.

5.3 Prototype Framework Evaluation

Regarding the different requirements a prototype for a bendable mobile device should satisfy (see 3—"Design"), the final prototype was a success. The following table 5.3 sums up briefly which demands were fulfilled in which way.

The size of the prototype was the only aspect of its form factor that perhaps does not equal a typical mobile device's shape. It more resembles a tablet PC or an ultra-mobile PC. Nevertheless, there are mobile devices of this size and it is very likely that some gestures that can be examined with the prototype are also useful on smaller devices. Concern-

ing the grip and feel nobody who saw or held the device during its development complained. During user tests, it was no problem for the participants to hold it. How extensible the software really is will be seen in the future, especially if other people work with it, but so far the idea to use an abstract class hierarchy for the event classes already proved to be useful. One kind of events, the mouse movements was implemented after the first user study, i. e. not together with the other event classes, and this was a matter of minutes.
The problems that occurred with setting up the edges in the second user study show that just using standard applications on Mac OS X can be harder than originally thought. The configuration does work, but its quality can obviously not be good enough for sophisticated experiments. It might be a better idea to write special purpose software in future experiments that interacts with the prototype framework more directly. This does not really negate the original intention to provide a whole framework to easily set up new experiments if one takes into account that it is much easier these days to program a small application for a desktop PC's operating system, especially Mac OS X, compared to developing an application for custom hardware.
Really surprising was how robust the system was, especially the hardware. It ran even with an older laptop and could be carried around and used without having to worry about damaging it. It was even taken to a conference and withstand the trouble of repeatedly being set up and transported on plane.

	Prototype		Pen		Shuttle	
Distance	Mean	Variance	Mean	Variance	Mean	Variance
27	5,8	4,5	6,9	5,8	6,2	1,6
33	8,2	39,8	7,5	17,0	6,0	3,2
35	7,4	2,6	6,4	2,2	7,1	7,9
36	5,5	1,3	5,8	2,2	6,0	13,2
39	9,6	38,0	7,6	5,8	10,1	39,6
41	7,1	17,7	6,7	2,0	7,2	3,3
42	6,4	2,8	9,7	4,1	8,6	7,7
46	7,6	8,5	6,5	1,9	5,8	5,0
48	6,6	9,1	5,5	1,3	6,5	7,3
50	8,1	28,1	7,1	0,7	7,3	5,5
53	8,4	37,8	6,4	2,0	6,8	1,6
55	9,8	12,7	9,2	29,0	5,9	0,8
60	8,2	8,8	6,3	2,2	7,0	5,9
61	8,5	5,5	7,7	0,6	8,7	9,4
62	9,0	8,5	6,3	2,0	5,0	0,6
63	6,9	6,1	7,2	2,6	5,1	4,0
68	7,1	2,5	7,3	1,1	6,7	3,7
69	8,1	5,1	7,0	3,7	7,1	0,7
69	9,5	8,7	8,9	0,5	8,4	6,8
70	8,3	10,3	10,0	4,6	8,6	7,9
72	8,7	19,3	7,5	6,4	7,9	1,5
73	9,4	3,7	10,1	13,1	9,2	7,7
73	8,5	8,0	8,2	1,1	7,8	7,6
73	8,5	11,5	8,2	0,8	7,6	6,8
74	11,1	8,7	9,6	4,2	8,6	1,9
75	12,5	29,4	9,8	18,4	6,4	2,7
78	9,7	22,2	9,3	9,3	6,7	1,8
79	8,7	4,2	8,5	5,8	6,4	1,5
79	11,6	14,4	14,6	114,9	10,9	1,5
80	8,1	1,9	7,5	3,4	7,3	6,0

Table 5.2: Excerpt from the data measured in the second user study

Common shape	Rectangular, a little too big
Many ways to bend	18 gestures, tested ones worked well, no gesture merging yet
Configurable actions	A bit uncomfortable to configure, but easily improvable
Continuous input	Worked well, improvement of fine-tuning possible
Nice Feel	Never slipped, nobody complained
Mobility	Medium, still needs a computer, but runs with a laptop
Software extensibility	Not used very much, one event subclass added later
Robustness	High, survived even flight trip

Table 5.3: Table to check demands for a bending prototype

Chapter 6

Summary and future work

This chapter provides a retrospection of the project as a whole, pointing out problems and summing up the contributions for the field of human computer interaction. Following this will be some ideas of future work that are meant to give and idea of what following experiments to expect or to serve as inspiration for the interested reader to do own research.

6.1 Summary and contributions

The main contribution of this work lies in providing a starting point for research on bendable mobile devices. When looking at similar projects it becomes clear that there is a great need for this kind of research. Several other works described in 2—"Related work" deal with this issue, but compared to other fields and input techniques in human computer action there is relatively little knowledge about paradigms or design guidelines for bending.

After this became clear, the author thought of what would be a good way to allow quickly getting base data about bending gestures. 3—"Design" illustrates what ideas came up dealing with this problem and shows what plans were

made in constructing a prototype device that would allow this research. Of course the whole concept was never meant to be the one and only method to achieve findings on bending gestures, but it tried to differ in a way from other projects. Instead of focusing on one or two precise aspects of bending, in the context of this work one could say one or two bending gestures alone, the prototype framework plan arose.

4—"Implementation" then describes how the demands and constraints noted in the previous chapter were put into a physical device. In the end, this development took up most of the time and it became clear, that the goal of this work was very high given the amount of time the author had. Nevertheless, the concept turned out in a usable way.
Something that showed to be very problematic were the materials the prototype was built of. Although each of them on their own first seemed to be just as flexible and deformable as was intended, it turned out that putting them together resulted in a slightly too stiff construction.

5—"Evaluation" described two user studies, one that mainly served to provide feedback on the prototype itself and one to actually provide a first example of how the system can be used to explore bending gestures. Due to the short amount of time that was left, they unfortunately do not provide very reliable numeric results, but the insight in people's perception of bending as input method should be very valuable for future studies. The second experiment also hints that bendable devices could serve very well as rate controlling input devices that outperform positional input methods like pens or touchscreens in certain scenarios, although this has to be proven by additional studies. This study also suggests that there is a natural mapping between page flipping or advancing in a list of items one by one and bending an edge of a mobile device. A negative aspect of the prototype framework that this study also shows is how complicated integration with standard applications can be, but it is still easier to set up new experiments with the system than building a completely new prototype, because any new custom application for experimental user studies can be developed under the host operating system, i. e. Mac OS X.

6.2 Future work

Of course the most imminent work to be done would be to repeat the second user study with a differently set up task and perhaps a special purpose application instead of the Mac OS X Finder. One could also use a different gesture for the scrolling, since most users did not find out how to execute such a simple and important action on their own with the "wave" gesture.

Other important aspects would include changing the core software to allow merging of gestures or improve the calculation of the bending degree of a gesture.
The K-Means clustering algorithm is perhaps not the best solution to detect which gesture is executed by a user, something that includes a complete virtual representation of the device would probably lead to a much better and more robust gesture recognition.

These aspects basically concern the software only, but changing the hardware itself would probably also be interesting. One has to consider that the current prototype is, not taking into account its smaller version, a very early try to build a bendable input device. Although the sensors used in it did well so far, there might be other solutions. The same goes for the material, it would be very good if the prototype was a bit more soft. Utilizing a tracking system instead of using a sensor based hardware device would allow testing different materials for their usefulness as bending devices. This would of course reduce the mobility of the prototype a lot, but opens options to do research on aspects that are not currently within technological reach for mobile devices, like including complete detection of the users posture.
Another idea is to include additional sensors in the hardware, like tilting sensors. In the second study, users mentioned to use the horizontal bending towards their face to scroll forward, since they do so with a book or magazine. The problem with this gesture is that from the bending alone it is not possible to determine whether a user would scroll forward or backward, since a book can be flipped in both directions when it is bend like this. What might usually be different between flipping pages forward or back-

ward in a book is the angle in which the book is held, skimming forward usually also tilts the book to the left so that its back points at the floor and the skimming pages to the users face and vice versa. This could be detected with a tilting sensor and allow a more natural mapping between scrolling and bending if relying on the book metaphor.

The last big issue with the hardware and something the author would like to try out as well is the inclusion of a flexible display. Either by attaching a real flexible display to the current or some future prototype device or by going for the tracking method and projecting an image on the bent device. Something that was left out completely from this and other works so far is how a user's interaction with a device is influenced if its main information display, the screen, is bend. Since bending a surface influences things like reflection and viewing angle, i. e. visibility, this is definitely an issue that has to be explored. It would be no use if a certain bending gesture alone is good for a certain action, but bending a device with screen in this way at the same time destroys all visual feedback from the device.

Appendix A

Instructions for the Prototype Framework

This appendix documents how to configure and use the software and hardware of the prototype framework described in this thesis. It refers to the original version of the software when this thesis was published. The sources needed to compile and run can be downloaded with the following link.

Sources[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/mainsources.zip

In addition, a pre-compiled version of the command line tool necessary to run the system is available:

Command line tool: twend[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/twend.zip

It contains a universal binary of the command line utility and an example configuration file for the event setup.

There is also a tool used for debugging the hardware device. It displays the measured sensor values in a graph and is contained in a separate file. Unlike the command line tool, it is a full application written in Cocoa. There is no bi-

nary version of it available, just an XCode project with the sources.

Debugging graph tool: twendgraph[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/twendgraph.zip

Since the usage of this tool should be self explanatory after the instructions for the command line tool have been read, an in detail discussion of it is omitted.

The hardware referenced throughout this description will also be the exact same as was used for the whole project. If an interested reader constructed a new prototype it is up to him or her to skip the parts that differ.

Setting up the Hardware

The hardware setup is very simple. The Arduino board and the prototype just have to be connected to a power supply of 5V. Both have the same type of plug, a standard multi-purpose power adapter should do fine. Next, the prototype has to be connected to the board. There are labels attached to the yellow and black cables coming out of the upper side of the device. These are numbered from zero to seven and have to be connected to the according analog input pins on the board. Note that the Arduino board originally did not have plugs for pins six and seven. These were soldered to the board manually by the author. They are not positioned next to the other pins, but a little above them, like the following figure A.1 illustrates:

After the wiring, the device should look something akin to figure A.2:

The also marked plugs for five volt and ground are meant to be connected to the red respectively yellow cable coming out of the device. It is important not to mix them, red is meant to be plugged into the 5 volt port.

The last thing to do is to set up the Macintosh to connect to

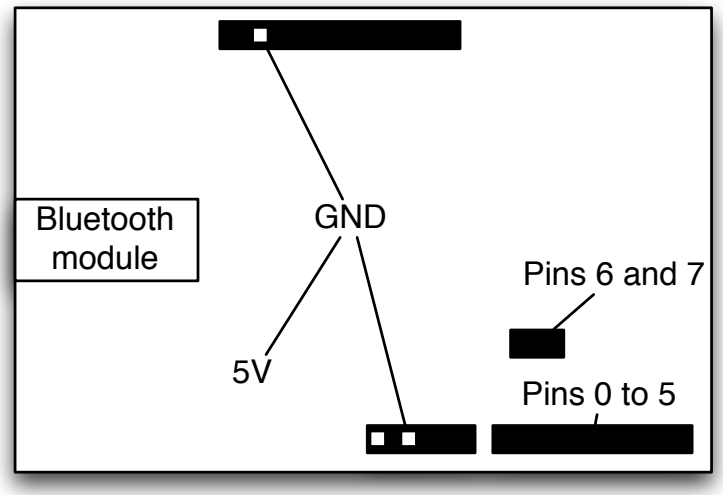

Figure A.1: Sketch of the plug layout on the Arduino board

the Arduino board. This can be done in the Bluetooth pane of the system preferences, please refer to the Mac OS X help files if having trouble to configure it. The Arduino should be set up so that a serial port is assigned to the Arduino. Remember the exact port name for later.

Configuring the Software

If running the software for the first time, there are no files to be parsed by the command line tool other than the event setup file. It contains a section describing how it can be configured, but here is a brief example assuming the upper right edge of the device is to be configured to invoke a scroll wheel event.

The `eventsetup.txt` file contains the following lines:

```
#upRBW:
NoEvent
0
```

This means that bending the upper right edge does not in-

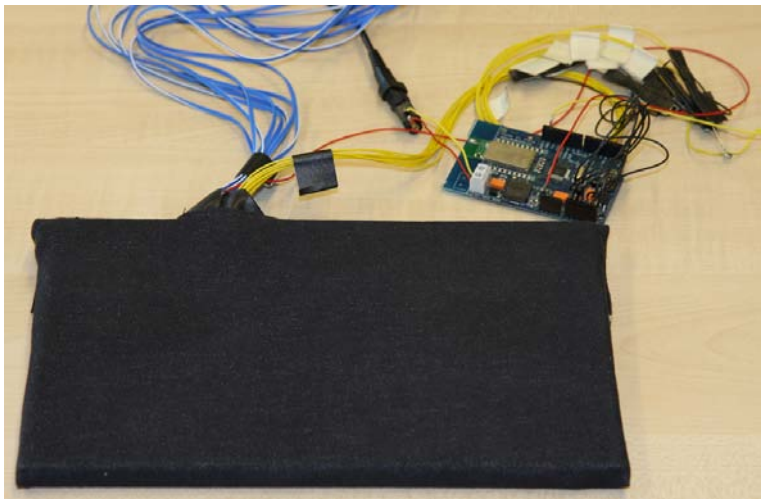

Figure A.2: Image of the finished hardware setup

voke any event so far. To change that, the second and third line have to be replaced by the following code:

```
ScrollEvent
1
0
0
100
0
```

Like the description of the `eventsetup.txt` file says, this stands for a scroll event. The second line represents a scaler that is applied to the bending degree before it is converted into scrolling units (use -1 to scroll in the opposite direction). The third line specifies which of the three possible axes (x, y or z) an apple scroll event can contain to associate with the gesture and has to be zero, one or two. After this comes a boolean, i. e. the next line has to be either zero or one. A zero means that the units to be scrolled in is lines, a one stands for pixels. The 100 configures the software to have at least 100 milliseconds between each scroll event that is sent. This, as well as the scaler, is one of the main values to tweak how the device acts. The last value has the effect that the degree of bending influences the delay between the scroll events if set to one. In this setting, the

stronger the device is bent, the shorter the delay between scroll events. Usually this setting is not necessary, but some applications might work better when this is activated.

For other types of event configurations please refer to the description at the beginning of the `eventsetup.txt` file.

Running the System

Now that the file is properly configured the software can be started. In the command line, type `./twend -p <port>` where `<port>` has to be substituted with the path to the serial port the Arduino uses. Usually this is something like `/dev/tty.ARDUINO_609-Bluetooth-1`. It is important that the Arduino is reset as soon as the command line utility is started, i. e. it is best to press enter and the reset button on the Arduino at the same time.

After a small while, the connection will be made and the system runs. The first thing that has to be done is to calibrate the sensors. The tool prints instructions on the command line, please follow them. The calibration values can be saved in a file as soon as the utility quits. Such a file can be parsed in future runs by attaching the parameter `-c <calibration file name>` to the `twend` command.

The next thing is the measurement of example values for the clustering that is necessary to recognize gestures. The utility again prints instructions for this on the command line, to sum it up all 18 gestures have to be performed with the device. It is important to be very careful at this stage to provide good clustering data. It has shown to be best to lie the device on a table and bend the different gestures in a way so they distinguish from each other as best as possible. At program end this calibration data can also be saved to a file, the parameter to attach to the `twend` command to parse it in future runs is `-C <cluster file name>`.

Now the system is running and one can invoke Mac OS X system events with the configured gestures. The gestures in use do not have to be bent in a way as careful as in the clus-

tering stage. Depending on how well the clustering went, they are recognized even if performed in a slightly different way.

There are some controling features of the command line utility that are probably useful during execution. These are invoked by tying single characters to the command line and listed in table A.1.

Character	Effect
x	Terminate the tool
v	Print the last read values to the command line
c	Print the calibration values to the command line
V	Continuously print the read values to the command line
C	Continuously print the distances for each read value vector to each 19 clusters to the command line. An xxx denotes values that did not lead to an event invocation
k	Re-calibrate the device
K	Repeat the clustering procedure
z	Re-calibrate only the offsets, i. e. partially re-calibrate the device

Table A.1: Command line utility control characters

This concludes this small introduction on the use of the prototype framework described in this thesis. The main programs XCode project contains some other executables respectively the source code for them that are meant to be used for development. This includes a very simple emulator for the hardware device utilizing a virtual serial port and an executable to integrate and test new `TWEvent` subclasses. Their exact functionality is not described here, because they require familiarity with the sourcecode of the main tool. If this familiarity is given, the other tools should be self explanatory just like the sensor value visualization tool described earlier.

Appendix B

Results of User Studies

Results of the first user study

User No.	Age Group	Gender	Likert*	Preferred	started with	Time (sec) Prototype	Time (sec) Keyboard
1	20-29	m	4	Prototype	Prototype	51,6	25,6
2	60-69	w	5	Prototype	Keyboard	75,4	125
3	50-59	w	5	Prototype	Prototype	152,6	31,8
4	20-29	m	5	Keyboard	Keyboard	26,5	49,6
5	20-29	m	4	Keyboard	Prototype	56,1	26
6	20-29	m	3	Prototype	Keyboard	75,5	26,8
7	20-29	m	5	Keyboard	Prototype	36,5	26,4
8	20-29	m	4	Keyboard	Keyboard	34,6	26,4
9	20-29	m	4	Keyboard	Prototype	33,9	25
10	20-29	m	4	Keyboard	Keyboard	43,5	26,9
11	20-29	m	5	Keyboard	Prototype	32,8	26,2
Mean			4,4			56,3	37,8
p-value						0,0915	
p-value	(without users 2 and 3)					0,0281	

* 1 = does not apply
2 = applies a little
3 = applies
4 = strongly applies
5 = fully applies

File: Excel version of the results[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/userstudy1results.xls

Distance	started from	User No. 1		2		3		4		5		6		7		8		9		10		11		12		13		14		15	
		Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device	Time (sec)	Device
74		13,21		12,61		6,39		10,00		8,00		7,62		10,76		9,53		7,60		8,97		11,68		7,84		11,71		13,27		7,27	
69		9,32		11,32		5,57		6,71		5,90		6,00		6,60		10,28		5,53		7,59		6,00		7,20		6,94		7,53		8,00	
46	left	3,90		6,21		7,55		11,82		3,64		6,12		6,37		8,57		4,72		9,39		6,69		4,54		6,31		8,34		5,02	
63		5,11		4,70		5,71		8,70		2,56		6,03		7,30		9,68		5,56		6,28		7,30		5,96		7,45		10,31		3,53	
62		9,06	Pr	8,80	Pr	5,96	Pr	7,45	Pr	4,81	S	5,90	Pe	6,09	S	8,51	Pe	4,94	Pe	5,36	S	5,42	Pe	3,99	S	6,85	Pe	13,74	Pr	4,54	S
79		7,71		10,84		5,70		10,23		5,95		8,34		6,97		12,54		6,72		6,88		8,60		7,78		6,54		8,82		4,60	
68		7,47		6,61		5,70		5,90		5,91		7,56		9,71		8,57		6,37		7,22		6,06		6,09		7,90		9,59		4,63	
36	right	6,60		4,40		6,78		4,51		3,21		5,23		5,36		6,54		4,03		6,25		7,90		12,12		5,30		5,17		3,30	
60		8,91		8,52		6,03		5,00		5,59		7,08		8,37		8,37		4,57		9,96		5,36		7,33		6,25		12,63		3,68	
48		6,30		4,80		6,19		4,10		2,57		5,99		7,59		7,23		4,46		6,40		5,23		5,81		4,63		11,78		10,00	
79		10,45		10,50		10,61		13,12		32,39		15,78		10,25		10,12		7,99		11,59		15,65		14,85		8,66		10,00		3,87	
73		8,10		11,35		6,80		6,75		9,17		7,99		11,14		9,30		8,97		15,78		12,63		8,21		7,87		12,85		6,28	
55	left	6,35		6,25		4,70		5,11		18,63		14,15		7,87		8,03		8,06		6,60		13,05		7,90		5,93		6,85		5,11	
78		6,77		4,78		5,91		7,75		9,08		17,53		14,34		8,40		6,28		9,11		10,15		7,17		5,96		8,09		6,61	
33		3,99	S	8,81	S	6,45	S	5,58	S	9,78	Pe	13,33	Pr	5,71	Pe	3,56	Pr	3,78	Pr	3,90	Pe	16,57	Pr	4,43	Pe	3,53	Pr	5,27	S	13,59	Pe
35		5,52		10,80		9,42		5,24		5,08		6,03		5,99		6,94		6,00		8,94		9,84		5,78		8,15		4,57		6,43	
53		6,28		6,64		6,50		5,63		7,84		5,08		6,28		5,71		6,25		7,14		19,39		6,51		5,65		8,91		4,12	
80	right	7,70		4,44		6,00		7,50		8,88		6,63		9,62		7,87		7,65		6,85		7,96		7,36		10,40		11,01		4,91	
72		7,43		8,53		5,92		8,70		7,81		16,18		11,39		8,91		5,74		5,27		6,60		7,84		5,90		8,82		5,20	
75		5,90		4,81		5,21		7,61		9,99		14,43		12,40		11,71		6,34		8,46		20,60		14,60		9,36		8,63		3,33	
42		7,99		9,81		13,12		8,78		4,79		6,72		9,05		10,00		5,65		5,39		12,66		5,71		8,18		8,66		6,82	
27		5,87		5,39		10,65		4,65		9,20		7,14		4,46		7,45		4,69		6,69		6,69		4,09		4,94		7,71		4,79	
69	left	9,68		9,57		8,64		8,37		10,76		7,36		13,90		12,51		6,31		8,81		9,36		7,39		6,40		8,25		6,54	
41		6,49		6,50		9,10		5,20		7,36		7,71		5,23		8,97		5,23		14,18		8,85		5,02		5,45		6,37		3,53	
73		9,24	Pe	7,48	Pe	6,90	Pe	9,18	Pe	9,56	Pr	8,60	S	6,69	Pr	12,00	S	6,18	S	12,88	Pr	7,42	S	7,84	Pr	4,76	S	8,28	Pe	5,64	Pr
39		6,93		11,87		6,84		6,45		17,27		6,45		8,09		9,65		5,78		14,82		21,05		3,42		7,62		5,96		4,57	
61		6,90		8,80		8,15		7,66		10,60		7,90		9,28		12,82		4,28		10,57		9,23		5,39		9,02		6,97		6,79	
70	right	11,15		8,24		9,36		8,13		13,05		7,00		10,03		12,40		6,06		6,28		10,85		6,82		6,79		13,20		5,27	
50		6,67		5,81		7,42		7,29		5,23		7,39		6,45		7,42		4,00		17,23		10,60		7,90		7,14		8,09		3,87	
73		8,07		7,26		7,50		8,84		7,02		7,62		9,71		10,79		3,84		13,84		9,08		5,56		6,76		9,45		6,25	

Pr = Prototype, Pe = Pen, S = Shuttle

Raw results of the second user study, ordered by person

Results of significance analysis of the second user study

Distance	p-value			
	Prototype & Pen	Prototype & Shuttle	Shuttle & Pen	
74	0,195	0,068	0,184	**bold entries denote statistically significant values**
69	0,209	0,191	0,456	
46	0,242	0,157	0,289	
63	0,414	0,123	0,054	
62	0,058	**0,017**	0,053	
79	0,470	**0,039**	0,066	
68	0,394	0,383	0,287	
36	0,361	0,379	0,446	
60	0,125	0,247	0,311	
48	0,235	0,466	0,247	
79	0,294	0,355	0,245	
73	0,346	0,454	0,328	
55	0,418	**0,032**	0,118	
78	0,439	0,117	0,068	
33	0,424	0,251	0,248	
35	0,182	0,426	0,327	
53	0,253	0,296	0,317	
80	0,296	0,280	0,445	
72	0,312	0,358	0,387	
75	0,202	**0,033**	0,082	
42	**0,012**	0,081	0,262	
27	0,251	0,386	0,300	
69	0,344	0,276	0,345	
41	0,437	0,467	0,317	
73	0,415	0,345	0,380	
39	0,262	0,453	0,222	
61	0,244	0,472	0,267	
70	0,175	0,434	0,203	
50	0,337	0,381	0,414	
73	0,440	0,333	0,322	

File: Excel version of the results & significances[a]

[a] http://hci.rwth-aachen.de/~herkenrath/thesis/downloads/userstudy1results.xls

Glossary

A/D converter

An electronic circuit to convert an analog signal, i. e. a certain voltage, into a digital signal.

eBook reader

A device or software application to view electronic texts. Usually has a graphical interface respectively form that is meant to allow convenient reading like when reading a novel. Commercial eBook readers are available.

Likert scale

Psychometric response scale often used in questionnaires. Instead of just agreeing or disagreeing, subjects formulating an answer according to this scale can specify their level of agreement. In this thesis the five levels were presented to the subjects as “does not apply”, “applies a little”, “applies”, “applies much” and “totally applies”.

point-and-click interface

A graphical user interface in which the user can control different responsive icons by pointing at them with an input device, in some cases in addition pressing or "clicking" a button associated with the device. The desktop metaphor found integrated in the interface of modern operating systems is an example for this.

Schmitt Trigger

In the context of this work a necessary threshold that has to be reached to initiate a state change in e. g. a final state machine.

softbutton

Usually a button that can change its associated effect on an interactive system depending on the context. Today the button is often nothing more than a specified area on a touchscreen and sometimes even vanishes to make space for other data to be displayed on the screen. In this case the softbutton is usually designed in a way that resembles traditional buttons.

strain-gauge

A device or sensor to measure the strain of an object. Usually a flat, flexible backing supporting a metallic foil pattern. Deformation to it causes its electrical resistance to change.

voltage divider

Literally a circuit that divides a electrical voltage between two resistors connected in parallel to each other. Often used to construct a circuit to measure voltage falling on over an adjustable resistor.

Bibliography

Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and Karan Singh. Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 111–118, New York, NY, USA, 1999. ACM. ISBN 1-58113-082-1. doi: http://doi.acm.org/10.1145/300523.300536.

Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-30987-4. doi: http://doi.acm.org/10.1145/274644.274647.

Ken Hinckley, Edward Cutrell, Steve Bathiche, and Tim Muss. Quantitative analysis of scrolling techniques. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 65–72, New York, NY, USA, 2002. ACM. ISBN 1-58113-453-3. doi: http://doi.acm.org/10.1145/503376.503389.

K. S. C. Kuang, W. J. Cantwell, and P.J. Scully. An evaluation of a novel plastic optical fibre sensor for axial strain and bend measurements. *Measurement Science and Technology*, 13:1523–1534(12), 2002. URL `http://www.ingentaconnect.com/content/iop/mst/2002/00000013/00000010/art00303`.

Jun Rekimoto. Tilting operations for small screen interfaces. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 167–168,

New York, NY, USA, 1996. ACM. ISBN 0-89791-798-7. doi: http://doi.acm.org/10.1145/237091.237115.

Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. Gummi: user interface for deformable computers. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 954–955, New York, NY, USA, 2003. ACM. ISBN 1-58113-637-4. doi: http://doi.acm.org/10.1145/765891.766091.

Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. Gummi: a bendable computer. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 263–270, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: http://doi.acm.org/10.1145/985692.985726.

James Scott, Lorna M. Brown, and Mike Molloy. I sense a disturbance in the force: Mobile device interaction with force sensing, May 2008. URL `ftp://ftp.research.microsoft.com/pub/tr/TR-2008-57.pdf`.

Index

Typeset May 27, 2008