DIPLOMA THESIS

User-oriented development of an universal hardware controller for graphical user interfaces of audio software

Benedikt Bengler

Institute of Electronic Music and Acoustics

University of Music and Performing Arts Graz, Austria
Graz University of Technology, Austria

Advisor: Prof. Dr. Gerhard Eckel
Co-Advisors: Dr. Werner Magnes and Prof. Dr. Gunter Winkler
Assessor: Prof. Dr. Gerhard Eckel

Graz, February 2011

Institut für Elektronische Musik und Akustik

Abstract

This project aims at developing a touchscreen-based device which enables the user to operate all virtual control elements of graphical user interfaces by handling a physical hardware controller. A fully functional prototype will be built being available at the IEM afterwards.
The user shall be enabled to access every virtual on-screen control via the touchscreen and to control it precisely using an on-board hardware encoder. Furthermore, no additional software will be required while the controller works regardless whether an external hardware control is supported or not.
In this way the all-purpose controller shall enable a task-oriented and intuitive handling of audio software applications.
An important aspect is an user-oriented development process which incorporates the future users already in the system design. By this means it should be ensured, that the development is significantly driven by the requirements of its users. Also the user interaction and the interface are going to be developed incorporating methods of the fields Human Computer Interaction respectively Interaction Design.

Kurzfassung

Ziel dieser Arbeit ist es, einen touchscreenbasierten Controller zur Steuerung von Audioanwendungen mit grafischer Bedienoberfläche zu entwickeln. Im Rahmen dieser Arbeit soll ein voll funktionsfähiger Prototyp entstehen, der in weiterer Folge dem IEM zur Verfügung steht.
Er soll dem Benutzer ermöglichen, jedes virtuelle, grafische Steuerelement einer Softwareanwendung ber den Touchscreen auszuwählen, um dieses dann direkt und präzise mit einem controllereigenen Hardwareregler zu bedienen. Hierbei soll der Hardware Controller jedoch nicht auf die Unterstützung der entsprechenden Softwareanwendung angewiesen sein und unabhängig davon funktionieren, ob diese für einen gewünschten Parameter die Kontrolle durch externe Hardware vorsieht oder nicht.
Auf diese Weise soll ein flexibel einsetzbarer Hardwarecontroller entstehen, der eine intuitive und aufgabenangemesse Steuerung von Audioanwendungen ermöglicht.
Ein besonderes Augenmerk liegt auf einem benutzerorientierten Entwicklungsprozess, der die zukünftigen Nutzer bereits in die Systemgestaltung miteinbezieht. Dadurch soll gewährleistet werden, dass die Entwicklung maßgeblich durch die realen Anforderungen der zukünftigen Nutzer bestimmt wird. Auch Interaktion und User Interface sollen mit Hilfe von Methoden aus den Disziplinen Human Computer Interaction bzw. Interaction Design entwickelt werden.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz,am ....

(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date (signature)

DANKSAGUNG

Ich möchte mich herzlich bei meinem Betreuer Prof. Dr. Gerhard Eckel bedanken. Er ist von Beginn an für das Projekt eingetreten, hat mich unterstützt und die optimale Rahmenbedingungen geschaffen, das Projekt im Rahmen einer Diplomarbeit am IEM zu realisiern.

Desweiteren möchte ich Prof. Dr. Gunter Winkler und Dr. Werner Magnes danken, die durch Übernahme der beratenden Kontrolle auf dem Gebiet der Hardware- und Geräteentwicklung die interdisziplinäre Ausrichtung der Arbeit durch ihre Bereitschaft zu interuniversitärer Zusammenarbeit maßgeblich unterstüzt haben.

Dank geht an Gerriet, Peter und Roland für ihr Engagement und ihr konstruktives Feedback bei den User Tests, Stefan für sein ständig offenes Ohr und seine Idee, ein zu scheitern drohendes Projekt als Diplomarbeit weiterzufüren, Dean für seinen großartigen Support der LUFA Library, sowie Marco für die Herstellung der Drehteile.

Weiters möchte ich allen Teilnehmern der Focusgruppendiskussion am IEM, Frau Bergner und allen Freunden danken, die mich während der Arbeit unterstützt und ermutigt haben.

Ein besonderer Dank gilt meine Eltern, die mir dieses Studium ermöglicht haben. Für Ihr Vertrauen und Ihre rückhaltlose Unterstützung.

“Technology, like art, is a soaring exercise of the human imagination.”
Daniel Bell

Contents

List of Figures

Chapter 1

Introduction

1.1 Basic idea

The aim of this work is to develop a hardware controller for computer music applications. The basic idea is to realize a control concept allowing to use the hardware immediately with any desired software without configuration. It should not rely on the support of the particular application and provide access to all of its available parameters.
The hardware controller - named Zeus Control - features a touchscreen as a central component and gets connected to a computer via VGA and USB. Around the screen several physical controls are grouped.
The Zeus Control enables its user to select any virtual control element via tapping it on the screen and regulate it with one of the Zeus's physical controls immediately. Virtual control elements can be control dials, sliders, X/Y pads as well as number boxes[1].
The universal functionality is given by the fact that operating systems having a graphical user interface (GUI) provide the mouse pointer as a standardized input device. Therefore all applications share the same control concept to handle user input. The core functionality of the Zeus control is to emulate these control data appropriately. The software application interprets the incoming data as usual mouse data and the desired parameter is controlled in relation to the user's input via the physical controls.

[1] A number box is a control element holding a parameter value that can be changed by mouse or keyboard input

1.2 Related work: “Mouse for Intuitive Media Software Control”

This basic functional principle has already been utilized in a previous project: The goal was to re-design the artifact “computer mouse” in a way that allows the user to control all on-screen parameters in a tactile and intuitive way. The user just has to move the mouse to a virtual knob or fader and is immediately able to control it with the hardware encoder placed on the mouse. As no configuration or mapping procedure is needed to make use of this additional feature, it perfectly fits into the plug & play concept of a computer mouse. Due to its special shape, the knob can be handled from the hand’s basic position while the mouse can be used as usual. Even if the Zeus concept shares the same basic principle, the context of use as well as the technical realization differs significantly from the mouse project: Instead of enhancing a “standardized” input device, the Zeus approach aims at providing an interaction *environment* for musical computing.

Figure 1.1: Mouse for Intuitive Software Control - Functional prototype

1.3 Combining touchscreen and physical controls for musical computing – a motivation

In the early 1980's a paradigm shift occurred in how to interact with computers: Direct-manipulation[2] interfaces replaced complex command-line interaction by manipulating visible objects directly on the screen. This new interaction style supported rapid, incremental operations providing instantaneous feedback. Especially the use of real-world metaphors (e.g. "desktop", "trash can" or "window") made it much easier for users to learn and handle software applications.
For many years the mouse remained the main interaction tool for desktop computer systems. But over the last years the touchscreen became more and more relevant as an input device for different kinds of applications. Isn't the touchscreen the most suitable interaction tool for direct manipulation anyway?
Instead of the relative control behavior of a mouse or a track pad, a touchscreen provides a direct relationship between the physical input, its caused action, and the graphical feedback: Graphical and physical interface become the same. This enables a direct and intuitive interaction as well as a rapid performance. It takes less to select a target on the screen as compared with a mouse – at least if the touch target exceeds a minimum measure [Sears and Shneiderman, 1991].
When during the 1990's the emerging computational power made real-time audio processing possible using desktop computers, studio technology has increasingly been transferred into the digital domain. Also the established user-interfaces were reproduced in every detail: Previous physical controls like buttons, rotary encoders, sliders or joysticks became the main control elements of audio application's graphical user interfaces constrained to be operated by a mouse. So why not resuming the direct access via touchscreen control?
Even if touchscreen interaction is very suitable for discrete control tasks like selecting items or actuating buttons, virtual controls for continuous tasks are a rather poor match compared to their physical archetypes. Their lack of precision and tactile feedback makes fine-grained parameter manipulations quite difficult and prone to error – an unsatisfying fact, especially with respect to to musical tasks.
Therefore, it seems promising to combine the best of both input strategies: The

[2]The term direct manipulation was coined by Ben Shneiderman in his seminal publication: Direct manipulation: a step beyond programming languages, published in IEEE Computer in 1983 [Shneiderman, 1983]

direct and intuitive access of a touchscreen and the accuracy and the tactile feedback of a physical control.
Even if the Zeus concept has been developed intuitively, its basic idea is nicely illustrated by an user study of Fiebrink, Morris and Morris [Fiebrink et al., 2009]. They conducted an user evaluation of an interactive tabletop system for simple audio editing tasks. The users were free to use the direct-touch controls of the tabletop interface or to map them freely to a set of physical controls (knobs and buttons). As a result, most continuous parameters were re-mapped to physical controls while almost all discrete task were executed directly on the screen.

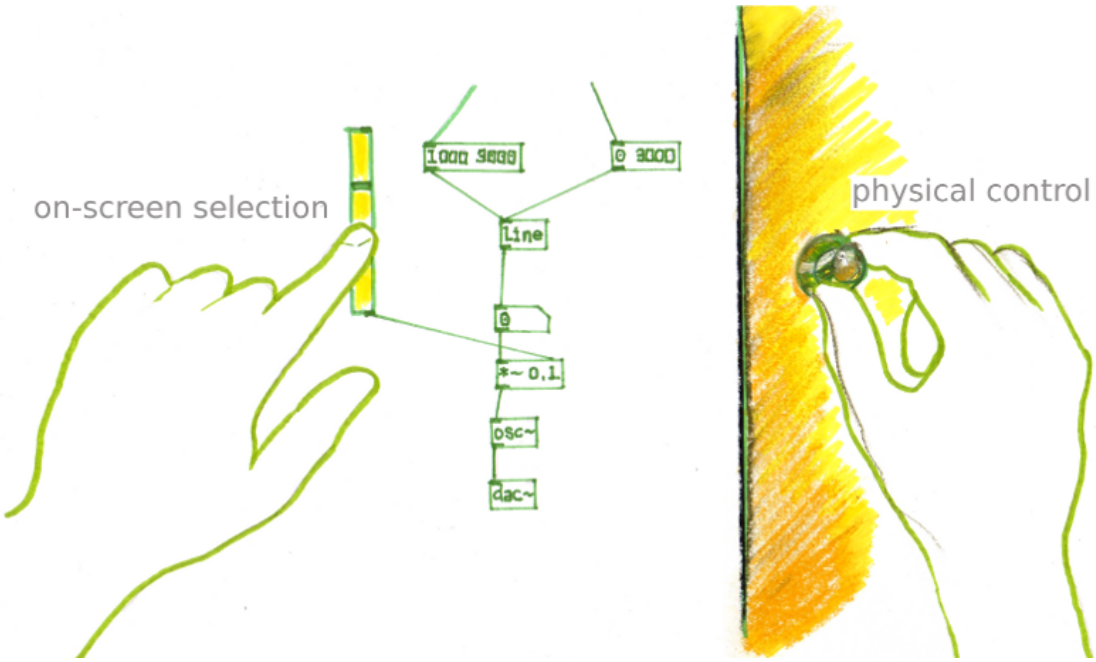

Figure 1.2: Scenario of use

1.4 A multidisciplinary development approach

In terms of its research context, the Zeus Control development can be assigned to the field of Sound and Music Computing (SMC). By combining scientific, technological and artistic methodologies SMC is a multidisciplinary approach to computer-based music and multimedia technology [3]. One of the key research issues relates to the improvement of human interaction with sound and music – a topic that directly refers to the development of new musical interaction devices like the Zeus Control.

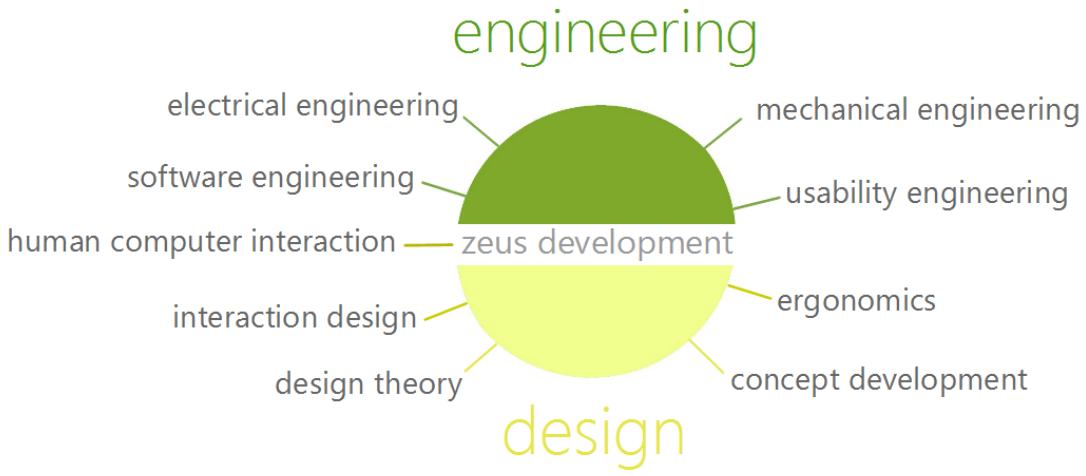

Figure 1.3: Related fields of research

The design and development of such a control device is a highly multidisciplinary task. In order to implement a novel, fully functional hardware controller utilizing an user-oriented approach, knowledge, methods, and techniques of several different disciplines are incorporated into its design process. The technical aspects of the development relate to the fields of software engineering, electrical engineering as well as mechanical engineering. The user-oriented design process is based on strategies of usability engineering and interaction design. The design of the user interaction and the interface is driven by methodologies of human computer interaction and interaction design while considering design theory and ergonomics.

[3] A comprehensive overview of the research field can be found in the “Roadmap for Sound and Music Computing” by the S2S[2] Consortium. The document can be downloaded at: http://smcnetwork.org/roadmap (accessed 30.01.2011)

This multidisciplinarity becomes a key issue of the Zeus Control development aiming to combine knowledge and methods of diverse research fields in terms of an overall goal – creating an innovative and practical controller for music computing.

Chapter 2

Methodological framework for a user-centered design process

The user-centered design approach is motivated by the challenge to develop a basic idea into a fully functional device that really suits its intended context of use. Involving users in the design process ensures a development significantly driven by the real needs of its future users. This approach relates to the entire development process and serves as an organizing principle. Therefore its structure, the used methods and their goals have to be formulated in advance.

2.1 Defining the process model

The process model provides the basic structure for the development progress (fig. 2.1).
I utilized the process model “Usability-Engineering” of Sardonick and Brau as a guideline for developing a basic project schedule [Sardonick and Brau, 2006]. My process model consists of three phases: *Analysis phase, concept phase and implementation phase*. All of them contain specific activities to involve the user in the design process. In the analysis phase general conditions like the fields of application and the user requirements are to be detected. Beside that, currently used systems are examined. The concept phase is meant to combine the results of the analysis with the technical possibilities to develop the conceptual framework. In the implementation phase the concept is realized step by step accompanied by user testing.

Moreover, I want to incorporate three terms into my process model which were coined by Löwgren and Stolteman [Löwgren and Stolteman, 2004]: *Vision, operative image* and *specification*. I consider them as very helpful to describe and communicate the creative aspects of the design process. These terms can be understood as three levels of abstraction in the design process. The vision is the initial idea that activates the whole process – in this case combining touchscreen interaction with physical controls. The operative image is the first externalization of the vision: A documentation of the basic idea being concrete enough to share the vision but flexible enough to provoke new ideas and diverse solutions. Therefore the operative image becomes the essential communication tool in the analysis and concept phase. Once the operative image of a particular system component is elaborated enough it can serve as specification for its implementation. Theses terms are considered when describing the design process in the following chapters.

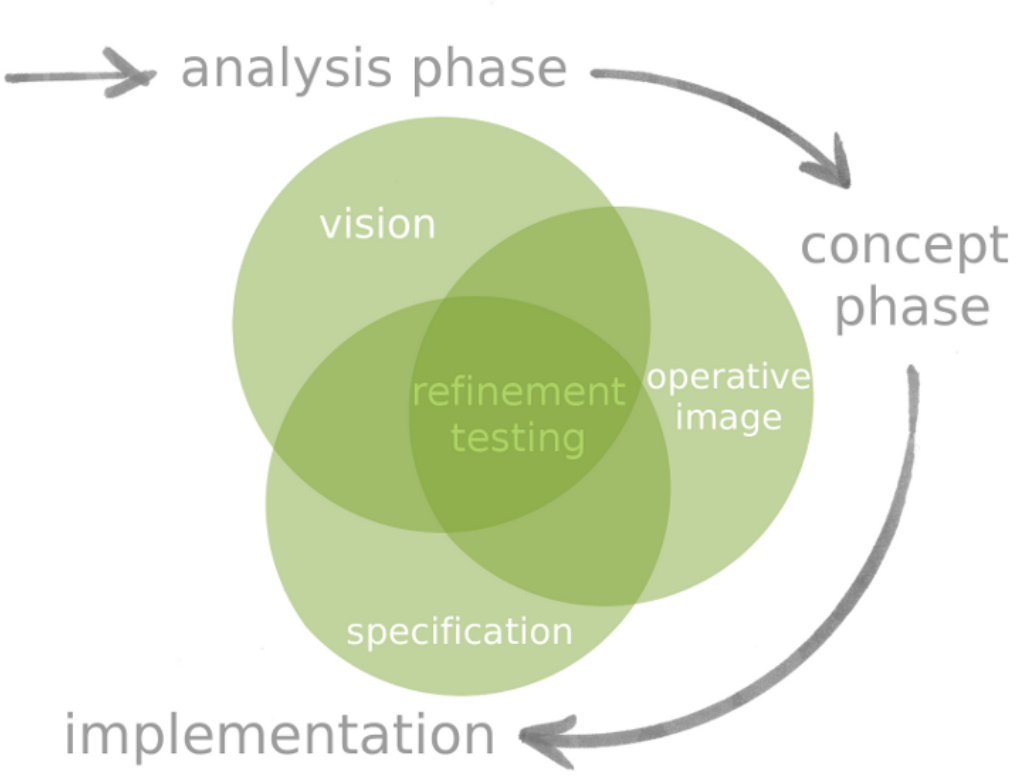

Figure 2.1: Process model for the Zeus development

2.2 A dynamic design approach

The process model of the Zeus development incorporates the very structured approach of having different consecutive phases as well as Löwgren and Stolteman's approach who perceive the design process as a dialectical process. For them the way from the vision to the final implementation can neither be described as a linear nor as an iterative process. Vision, operative image and specifications influence each other continuously in a fully dynamic dialectical process.
The way I combined both approaches – seeming contradictory at first – is to apply them on different layers: The clearly defined phases are used to structure all practical and organizational issues of the design process. They serve as an *organizational principle* helping to time specific activities like user tests or intermediate presentations. Beside that, the well-structured approach allows to communicate and document the design process in a clear and reasonable manner.
Within this basic framework I utilize the approach of Löwgren and Stolteman as a principle concerning the *creative issues* of the design process. Generating new ideas and finding solutions for particular problems is an overall process that can hardly be restricted to a certain phase. Moreover, they result from a dynamic interplay between previous ideas, new findings and the current design situation. For instance, a newly implemented sub-system enables a specified user testing which can evoke entirely new ideas that affect and change the previously defined concept. Therefore the dynamic dialectical approach is far more realistic to characterize the creative decision making. Being responsive to the current situation and open to re-think and question previous ideas throughout the whole design process allows to tap its creative potential. This attitude can be characterized by the term "*reflection-in-action*", coined by Donald Schön, describing a continuous reflection and adaption while being actively engaged in a practical process [Schön, 1987].
As this dynamic approach is difficult to describe adequately in the following chapters, which correspond to the three phase model, I try to illustrate the creative reasoning as part of the design process in chapter 6 presenting several concrete examples.

Chapter 3

Analysis phase

3.1 Aim and approach

The aim of the analysis phase is to lay the foundations for a concept development that meets the user requirements. These are related to the users' intended tasks as well as to their general working conditions.
Before entering into a dialog with the users I created a first operative image containing a functional description and a first conceptual sketch to communicate my ideas. The analysis phase was carried out using a focus group approach: I invited the staff of the IEM – the main users of the future system – to a meeting at the institute. The focus group consisted of about 15 people. In the first part of the meeting I presented the idea and related the Zeus Control to other controller concepts in order to illustrate and classify its functional range. This discourse is transcribed in the following section.
The presentation and classification served as a basis for the elicitation of the user requirements. These were identified with the aid of guiding questions as well as in an open discussion. This progress is documented in section 3.3.

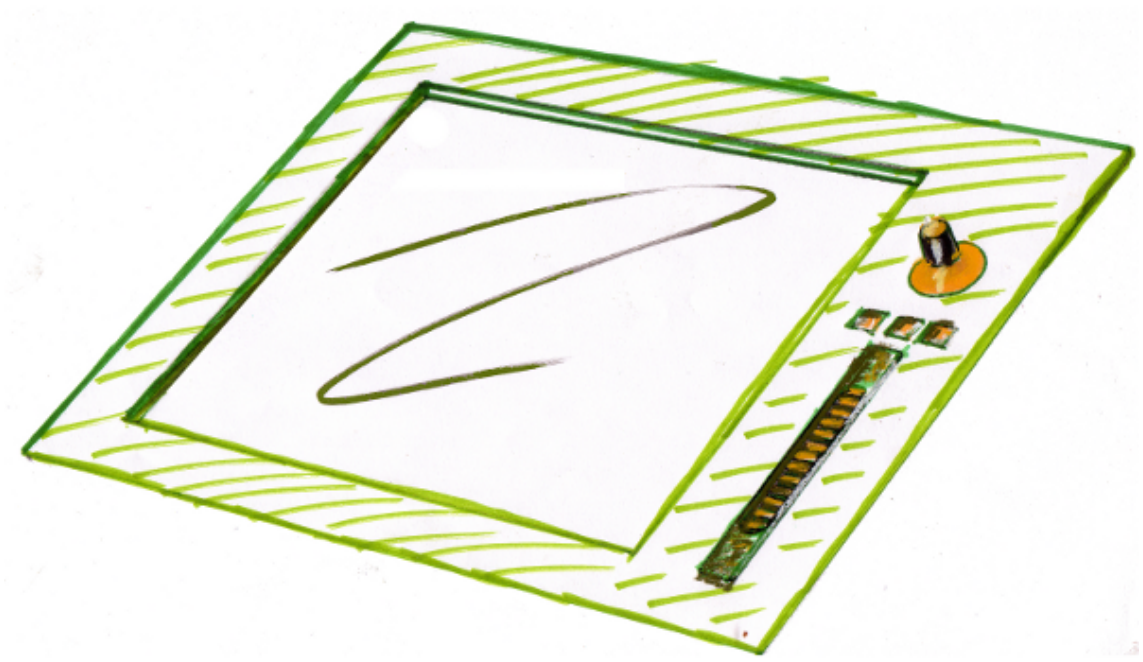

Figure 3.1: Zeus Control - First conceptual sketch

3.2 Classification of the concept with regard to other solutions

3.2.1 Established controllers: Mouse, MIDI and DAW Control

Established control devices for computer music applications range from the mouse to various kinds of MIDI[1] controllers up to extensively equipped hardware surfaces for Digital Audio Workstations (DAW).

The mouse is such a integral part of today's graphic-based computer systems that it is not considered as a separate controller. Being our standardized input device its superordinated control mechanism is supported by all software applications providing a graphical user interface. In the field of audio software most GUIs still refer to real hardware devices. Therefore, they are mostly dominated by established control elements like potentiometers or sliders. Operating these with a mouse, the handling in terms of precision and smoothness is significantly

[1] MIDI is the abbreviation for Musical Instrument Digital Interface and is the most common protocol for communication between electronic musical instruments, computers, and control devices. The original specification for the serial MIDI protocol was defined in 1982.

poorer compared to their realization in hardware.
Using one of the many available MIDI controllers allows the user to map several parameters to their hardware controls as long as the parameters are supported by the application's MIDI implementation. In a configuration progress selected parameters get explicitly assigned to the available controls of the MIDI device.
Large hardware controllers are mostly characterized by a high degree of integration into the software system they are built for. This provides comprehensive interaction possibilities implemented using extensive input and feedback devices like motor faders, LED ring-equipped rotary controls or embedded displays. Such a system is most efficient if utilized for established audio engineering tasks like multi-track production. However, due to their preassigned functionality they are only suitable to a limited extent for other musical applications.

3.2.2 New technologies: Tangible and Multi-touch interfaces

In the recent past also new technologies started to appear in the field of musical computing: Tangible user interfaces[2](TUI) or multi touch technology expand the possibilities of how to interact with audio software. The main advantage of both approaches is that they enable bimanual control of several parameters accompanied by rich graphical feedback. In this way entirely new interfaces for musical interaction can be realized. The downside of both technologies is, however, that at present these kind of input devices are not supported natively by any common music software application. Even if some technical preconditions like integrated multi-touch capabilities found their way into common operating systems, a standardization of these input devices is not yet in sight[3]. Therefore a multi-touch or TUI control interface has to be implemented individually for every application. Even if rewarding for special applications, not every musician is able or willing to first create the tool he or she wants to use.

[2]A tangible user interface enables to control digital information via a physical representation that can be directly manipulated by the user. A well-know, musical TUI application is the tangible synthesizer *reactable* developed by the Music Technology Group (MTG) of the Universitat Pompeu Fabra in Barcelona (http://mtg.upf.edu/ (accessed 20.12.2010)).

[3]Multi-touch and physical controls: Even if multi-touch technology will be standard in the near future, a basic problem of touchscreen interaction – the lack of precision and feedback regarding continuous tasks – will remain. Therefore, the combination of touchscreen interaction and physical controls is equally relevant to multi-touch systems

3.2.3 Defining the Zeus control's profile

A design avoiding all drawbacks of the other solutions while combining all their advantages is unrealistic to a high extent. Hence, I think it is important that a newly developed artifact is clearly positioned within its related field. Thus, important questions are: What are its crucial advantages? What are potential weaknesses compared to other solutions? How can these be resolved? This reflecting process serves as a crucial tool to provoke new ideas how to use, extend and improve the device.

As there is no *'swiss army knife'* solution, I think it is important to consider potential flaws as well as to reflect how these can be bridged in future use.

A key consideration is that since the Zeus Control is based on the control mechanism of a computer mouse, all virtual controls can be operated immediately. But in contrast to the mouse, it provides appropriate haptics for musical tasks: The rotary encoder and the sliders enable a smooth and fine-grained control of musical parameters. However, compared to a MIDI controller no configuration or a restriction to MIDI-compatible parameters is needed.

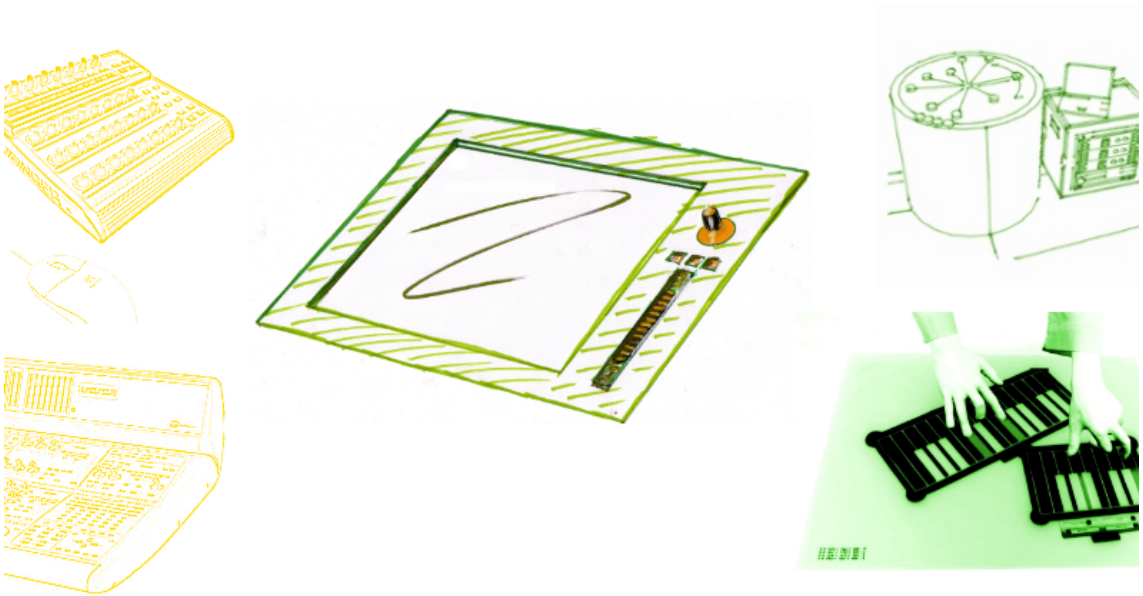

Figure 3.2: Defining the controller's profile

At large controllers not just the controls for continuous parameters, but many other functions of the software interface are duplicated in hardware: One thinks of all kinds of buttons, displays or metering. This redundancy should be avoided by the touchscreen setting the focus on the software's graphical interface itself as the center of interaction. Beside continuous parameters, also discrete ones like function buttons can be directly controlled via the touch interface. In this way the spatial and structural separation of the graphical and the haptic interface disappears. The close relation of haptic input and graphical feedback supports an intuitive and coherent interaction causing a low cognitive workload.
But as the immediate and universal functionality is based on the mouse mechanism, a simultaneous control of several parameters may is inherently impossible. However, dual parameter input is available for two-dimensional, virtual controls like X-Y pads often used for panners, equalizer curves or effect parameters.
So how significant is this limitation in terms of the context of use? The Zeus Control can be considered as an *interaction environment* rather than a separate controller like e.g. a MIDI box. The essential computer peripherals monitor and mouse are directly integrated into the system and enhanced with new possibilities of how to interact with one's familiar software. Therefore, the Zeus Control does not compete with established solutions like a MIDI box or a DAW control. It is a kind of enhanced computer peripheral combining display and universal input device – both components that can hardly be replaced by an external controller.
Due to these reasons, the limitation regarding multiple parameter input can be balanced considering the Zeus Control as an universal peripheral for musical computing that enables immediate control of every off-the-shelf application without configuration. Furthermore, the system can easily be combined with established concepts like MIDI – to their mutual benefit. An appropriate scenario could be a DAW-based mixing process. Using a MIDI fader bank for simultaneous level adjustments and the Zeus Control for all other parameters of the DAW and the used plug-ins would result in a compact but comprehensive mixing system.

3.3 Elicitation of user requirements

In order to incorporate the specific demands of the IEM into the development from the very beginning, a user survey was conducted right after the project kick-off presentation. Therefore several guiding questions have been prepared in advance. These were meant to discern important demands of the future users while structuring the focus group discussion. The five guiding questions were:

- Which operating system would you prefer to use the Zeus Control with?
- What kind of software do you like to control?
- Where do you see potential fields of application?
- What technical features would you appreciate?
- What are your ideas and suggestions?

Everybody in the group had the possibility to answer these questions which also served as basis for a collective brainstorming session in the following. The individual responses were instantly compiled and visualized with a digital mind map tool (fig. 3.3). Beside that, the meeting was videotaped to document the discussion and capture details.
Using a moderated focus group survey as an evaluation method for the analysis phase turned out to be a good choice. Beside exposing the most important requirements, ideas were generated that strongly influenced the further development process. The most striking example is the idea to realize the touchscreen driver entirely in hardware in order to avoid a driver implementation for several operating systems. Even if the focus group and I discarded this suggestion as not being feasible, it remained in my memory. Reconsidering this idea several times during the design process in context of newly acquired, technical knowledge made it possible to finally realize this lasting idea yielding a unique solution.

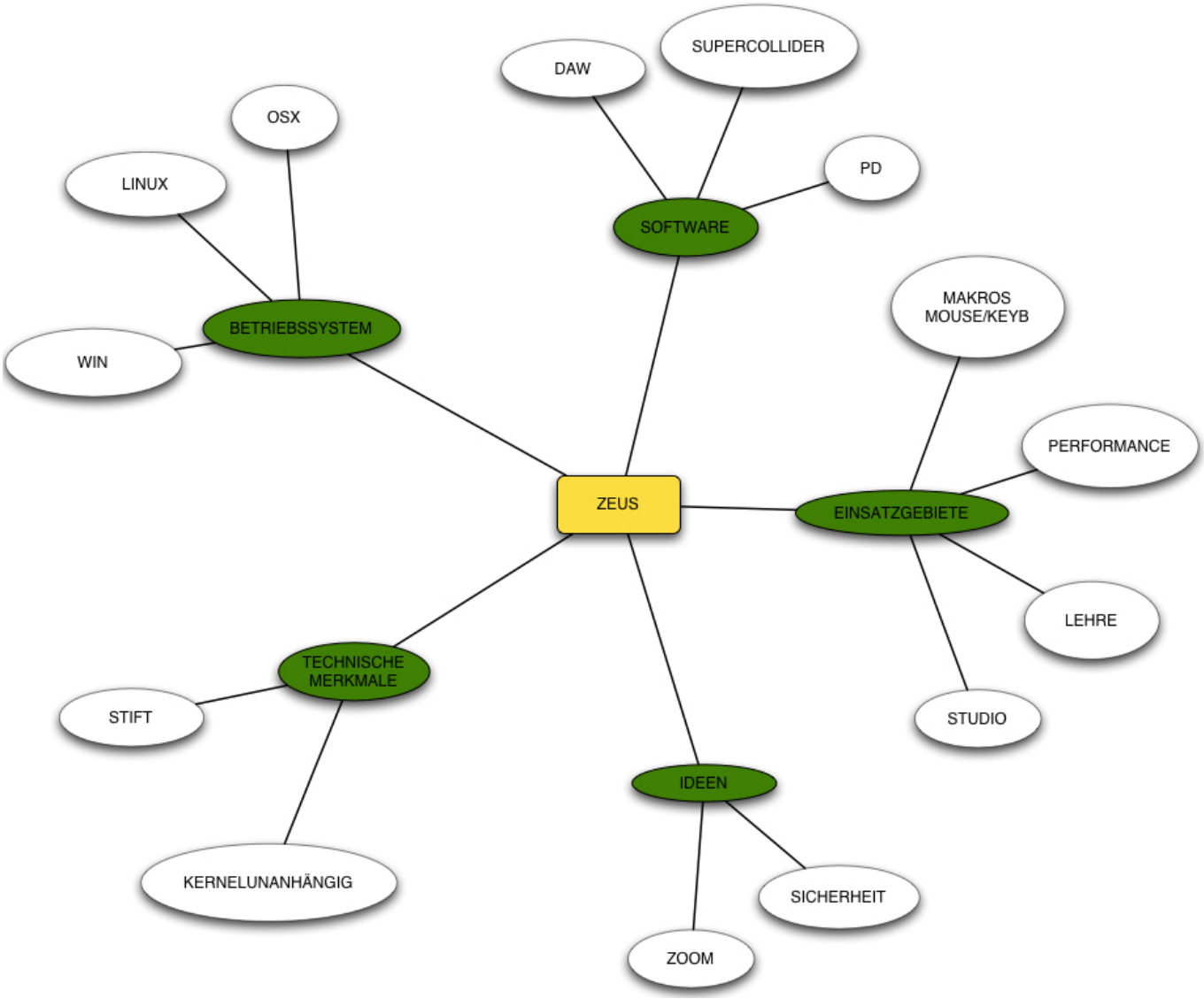

Figure 3.3: Mindmap of the focus group discussion

Chapter 4

Concept phase

sectionAim and approach Based on the results of the analysis phase, a concept is developed combining the technical possibilities with the actual user requirements. Guided by the results of the focus group discussion, appropriate solutions for the particular requirements are found and integrated into the overall concept.

4.1 Evaluation of the focus group survey

In the following I want to point out especially two user requirements as they have the most significant influence on the technical concept: Cross-platform compatibility and operating reliability. The remaining demands will be covered in section 4.1.3.

4.1.1 Cross-platform compatibility

It appeared in the discussion that the Zeus Control should work under all operating systems used at the institute. These are Windows, Linux and OS X.
With regard to the conventional way to develop external hardware for personal computers this requirement is quite demanding: Normally a separate software driver for each operating system is needed according to their different specifications. In order to avoid an individual driver development for all three systems, the idea was to implement the Zeus Control as a Human Interface Device [1] (HID) as

[1] A HID is a device that belongs to the USB human interface device class describing devices such as mice, keyboards, joysticks or game controllers. The USB HID class is defined in the

the corresponding HID class drivers are already integrated in the common operating systems. If this would succeed, the implementation of the Zeus Control as an universal peripheral for musical computing could be realized in the most elegant way: Without the need to install drivers, the controller would work as a universal, cross-platform, plug & play device.
This strategy is highly suitable for generating appropriate mouse data according to the user's input at the physical controls: The Zeus Control is recognized as a standard, HID compliant mouse by the host computer. During a control process relative mouse data are sent to the host in order to increment or decrement the actual cursor position while the status of a pressed left mouse button is reported. In this way all virtual controls of a GUI can be operated via the physical ones.
The main difficulty of this solution scenario is the implementation of the touchscreen: Instead of relative control data, the absolute screen position of the finger is required. But as the touchscreen's sensor overlay is detecting and transmitting the touch position related to its physical, spatial coordinates (touch coordinates), a further processing step is needed in order to convert the touch coordinates into absolute screen coordinates dependent on the chosen display resolution. This task is normally accomplished by a software driver.
Therefore, the question arose whether this coordinate mapping is at all possible in hardware – especially if the device should be plug & play. The main problem here is that in contrast to a software driver the actual screen resolution is unknown.
The first potential solution I aimed to realize was to find a way to determine the screen resolution with the hardware in order to convert the touch coordinates into absolute screen coordinates. By reasons of the user requirement *Operating reliability*, which will be addressed in the next section, the Zeus Control has to be able to serve as a host for a computer mouse. The idea was to implement a small, Java-based, cross-platform application that is able to detect the current screen resolution and generate a scaled, graphical calibration mask. In a calibration procedure the hardware could determine the ratio between the absolute touch position and the relative displacement of the mouse cursor by matching the absolute data of several defined calibration points with the relative data of the connected USB mouse. In the following the controller could send every touch screen position as a relative cursor displacement using the conventional mouse proto-

Device Class Definition for HID 1.11 by the USB Implementers' Forum [USB-IF, 2001a].

col. Even if this solution avoids the problem of the unknown screen resolution, several problems occur: The standard mouse protocol specifies the maximum displacement per mouse report as 127 (7-bit) in x and y direction. In order to send displacements greater than 127, the controller had to decompose the value and send it using several, consecutive mouse reports. Another problem is that the sent values just result in same displacement in pixels if the cursor behavior is not further modified by the operating system: Common mouse modifications like "enhanced acceleration" or adjusted speed settings have to be deactivated to ensure a correct execution. Struggling with these mentioned drawbacks during the implementation process I reconsidered the USB mouse protocol specifications when discovering a feature that seemed worth to change the entire strategy:
It is possible to modify the HID Mouse Report Descriptor that describes the functionality and report data structure of the device. In general, this information is needed by the host to communicate with every HID device correctly. Modifying the Mouse Report Descriptor allows to indicate that the sent data are absolute values. In addition, the resolution of the touch coordinates has to be transmitted to the host. The detailed modifications of the descriptor are shown in appendix A.3. Knowing the range of values the HID Mouse Driver is able to convert the touch coordinates into appropriate screen coordinates in consideration of the current screen resolution.
In this way the standard mouse driver can also be used as a driver for the controller's touchscreen. This solution is a perfect fit for the intended cross-platform, plug & play approach.

4.1.2 Operating reliability

Operating reliability refers to the avoidance of unintended input. This requirement is highly relevant, in particular for live performance situations.
If controlling graphical interfaces initially designed for mouse input via a touchscreen, unintended input can occur easily. For example, if the user touches a number box in Pure Data[2] multiple touch events are generated caused by the size of the finger tip (coll.: *"fat finger problem"*). This results in an unintended

[2]Pure Data is a graphical, open-source, real-time programming environment for computer music, audio, video and graphic processing. The programming language has initially been developed by Miller Puckette at the IRCAM: It is comprehensively used at the IEM and has also been extended by the institute. (http://puredata.info/ (accessed 01.02.2011))

parameter change: The value of the touched number box fluctuates erratically around its previous value. In this way the user already changes the parameter unintentionally during its selection.
However, unintended input can also occur during the control process:
The operating system is determining the current cursor position superposing all incoming data of relevant devices like mouse, touch pad or touch screen. Therefore erroneous data can occur if the control data generated by the Zeus Control are not synchronized with the data of the touchscreen or a connected mouse. A possible example is: A new position on the touchscreen is selected while the controller transmits the status of a pressed left mouse button as one of the physical controls is used. This would correspond to the input of a mouse that is moved all across the screen while the left button is pressed causing various unintended input events which have to be avoided in any case.
To enable an error-free on-screen selection of virtual control elements, two strategies can be considered: One is to resemble the *"hover mode"* of a mouse: Instead of sending successive touch events as long as the screen is touched just the position data are sent to the host. In this way the user is able to safely navigate the cursor across the screen. In order not to lose the directness of touchscreen interaction, a single touch event is sent for the initial touch position. Hence, the user can operate discrete tasks like clicking buttons directly. To enable a smooth and intuitive transition between this save *"initial touch mode"* and a mode that allows continuous tasks like dragging objects across the screen, a specific solution emerged during the development process: Beside serving as physical controls for continuous input, the two neoprene sliders act also as activators for touch events. Due to their size and position being easily accessible, the user can touch them at any position while dragging her/his finger on the screen to trigger continuous touch input.
The second approach is a touchscreen mode which transmits the initial touch event while further touch data are ignored for a certain amount of time relating to the typical duration of an on-screen selection task. In this way the user can select a virtual control by tapping without causing unintended input. But if the user's finger is placed on the screen longer than this certain duration, touch status as well as position data are transmitted to the host enabling continuous touch input.
Based on the feedback of a user testing session I incorporate both touch modes into the final implementation allowing the user to select the one that suits best

her/his actual needs.
To avoid unintended input caused by simultaneous use of mouse, touchscreen and physical controls as mentioned previously, all control data have to be coordinated by the controller's hardware before sent to the host computer. Therefore, the Zeus Control must serve as a host for an external computer mouse in order to align its input with the data of the physical controls and the touchscreen. Then the user input can be transmitted to the host computer as a matched data stream.

4.1.3 Further user requirements

The user feedback concerning the fields of application and the considered software applications shows that the controller is intended to be used for live performances as well as in the studio. For studio use a flexible integration into the familiar working environment is favorable. Therefore, the controller features a high-quality, 19-inch touchscreen that can be used as a standard system monitor. The inclination angle of the device should be adjustable in order to take account of the user's position and the lighting conditions. Regarding live performances all functions should be obvious and easily accessible. The main controls will be illuminated to ensure a safe handling at dark next to the bright display of the touchscreen.
The request for a *"zoom"* functionality of the physical controls' input behavior will be incorporated as follows: The user can switch between three modes to set the relation of physical input and resulting cursor speed.
Using a pen instead of the finger as an alternative input device for fine-grained editing tasks is enabled by utilizing five-wire resistive touch technology. A detailed description of the touchscreen can be found in section 5.1.3.
Another application area mentioned in the open discussion with the focus group was that the controller should serve as a tool for guest musicians and composers to experiment with the institute's sound facilities. To keep the training period as short as possible, the interface should be clearly structured and self-explanatory. A Quick Start Guide will be directly accessible from the USB flash drive integrated in the controller which also contains all other relevant files and documents.

4.2 Defining the technical concept

In the following I will mainly focus on the technical aspects derived from the user survey. Demands that address the look-and-feel and the user interface will be covered in chapter 6.

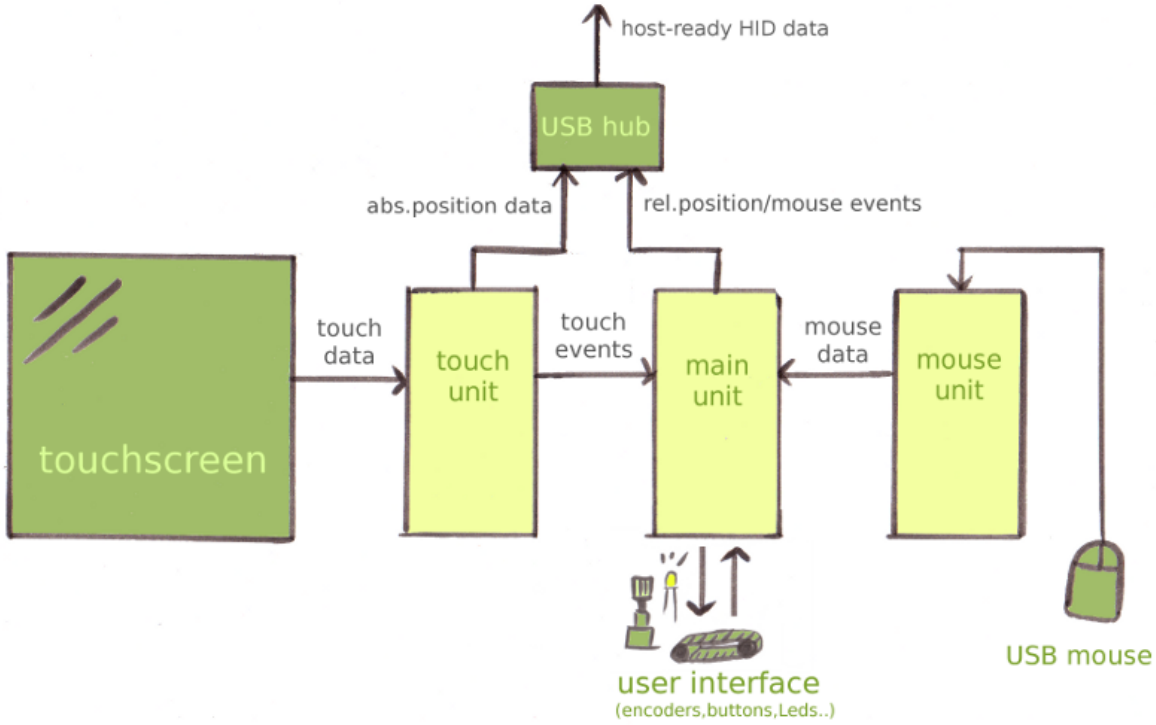

Figure 4.1: Schematic block diagram

By combining the found solutions for the particular user requirements, a functional representation of the Zeus control can be created which serves as the specification for the implementation. Therefore, the overall system is structured into functional blocks. A useful representation referring to the specific needs determined during the concept phase consists of three units: A main unit, a mouse unit and a touchscreen unit. The mouse unit has to serve as a host for an external USB mouse being able to decompose its HID report and send the data to the main unit. The touch unit has to interpret the user's input via the touchscreen and generate absolute position values according to the solution presented in section 4.2.3. The

absolute position data are sent directly to an internal USB Hub whereas the touch status is reported to the main unit. The main unit acts as a central processing unit controlling the user interface and coordinates the input events of physical controls, touchscreen and external mouse. In this way the main unit is able to generate the appropriate mouse events for the host computer. Beside that, it outputs relative position data derived from the input of the physical controls and the connected mouse. As the touch unit, the main unit is connected to the internal USB Hub in order to realize a single USB connection to the host computer. The Hub can also host the USB flash drive used to provide all controller related documents.

Chapter 5

Implementation phase

5.1 The hardware platform

The choice of an appropriate hardware platform was mainly driven by the basic technical requirements. In particular, demands that were decided at the very beginning, like the intended USB functionality, were most significant for the selection. As due to practical reasons the decision had to be made quite early in the development process, not all potential demands could be predicted in detail. Therefore, I aimed at choosing a hardware platform that allows for flexibility in terms of its broad technical specifications. I also tried to consider the concept of modularity throughout the implementation process in order to react quicker to new or modified demands. An example showing the reward of the modular approach is the integration of the touch unit: Even if the opportunity to implement the touchscreen functionality as a separate HID device appeared quite late in the development process, it could be easily integrated into the existing hardware environment.

5.1.1 Atmel AT90USB1287 microcontroller

The main component of the Zeus hardware is the Atmel AT90USB1287 – a 8-bit RISC microcontroller with in-system programmable flash memory [Atmel, 2009]. However, the most important selection criterion was its integrated, on-chip USB controller. This allows to realize HID compatible USB devices without the need of additional hardware for setting up an USB connection. The AT90USB1287 also

complies with the USB On-The-Go (OTG) specification which enables to implement applications that operate as a USB host for other devices [USB-IF, 2001b]. In this way, USB devices can communicate directly with each other without a PC serving as a host. This feature is crucial for the implementation of the mouse unit which acts as a embedded host for a standard USB mouse.
Beside that, the controller offers the common hardware interfaces USART, SPI and I2C. As the chip provides 48 programmable I/Os, its is possible to connect all interface-related components (LEDs, buttons, encoder) directly without further multiplexing.

5.1.2 Atmel AT90USBKey development board

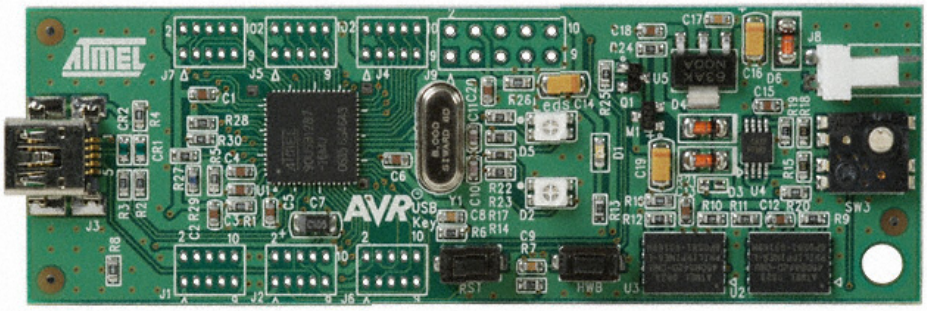

Figure 5.1: Atmel development board

In order to enable rapid prototyping I choose the Atmel AT90USBKey – a development board built around the introduced Atmel controller [Atmel, 2006]. This enables a quick start benefiting from the on-board peripherals while avoiding the need for SMD soldering. A main advantage is the regulated 3.3V power supply supporting input voltages within a range from 8 up to 15V DC. When using the microcontroller in USB host mode, the AT90USBKEY provides a 5V power supply for the connected device over the VBUS pin of its USB mini connector. Beside that, the bootloader execution can be forced via two hardware buttons allowing to reprogram the chip directly over the USB interface. A problem was the quite small footprint of the I/Os (1.27mm pitch) with no headers mounted. In order to use female-to-female jumper cables for prototyping, I extended the board with

PCB adapters converting the port pins to standard headers (2.54mm pitch).
The entire hardware environment is based on three linked AT90USBKey boards, one for each functional unit.

5.1.3 Elo1939L touchmonitor

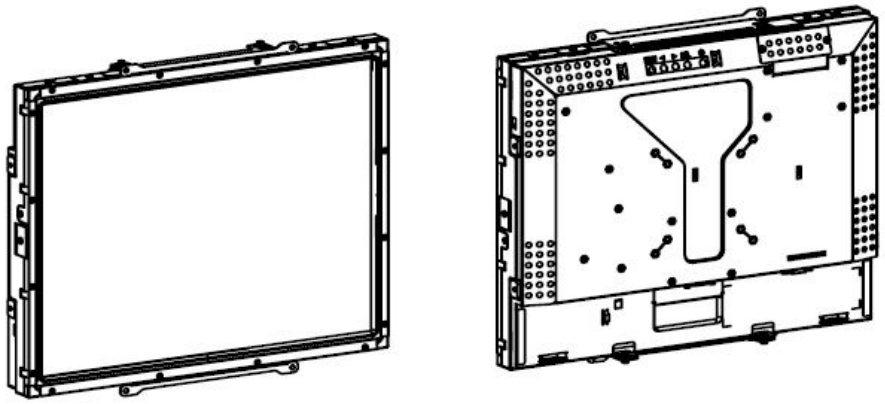

Figure 5.2: ELO open-frame touchmonitor

The used Elo1939L touchmonitor features a 19” TFT display with a maximum resolution of 1280 x 1024 (SXGA) combined with 5-wire resistive touch technology. A 5-wire resistive touch sensor consists of a flexible and a rigid layer separated by spacer dots. Both layers are coated with a thin, conductive film. If a finger, or an other object, touches the surface the layers become connected and serve as a voltage divider. The X-position is determined by applying a voltage gradient in X-direction across the rigid layer and using the flexible layer to measure the touch voltage corresponding to the X-position. Accordingly, the Y-position is determined by applying a voltage gradient in Y-direction and using the flexible layer to measure the touch voltage. The signals are controlled and analyzed by a touch controller deriving the touch positions.
The main reason choosing a 5-wire resistive touchscreen for the Zeus control was that it offers a accurate and responsive performance while fulfilling the user requirement of input flexibility (finger, soft-/hard-stylus). Beside that, the integrated dual Serial/USB controller was very useful during the prototyping process. Using the screen in USB mode with a modified driver for Windows, I was able to

simulate most of its later functionality. In this way realistic user testing could be conducted long before the touch unit was implemented.
In order to transmit the touch data to the touch unit of the Zeus Control, the device is now used in serial mode. In addition, the touchmonitor has an internal On Screen Display (OSD) control for picture quality adjustment, which can be accessed via a mini-DIN 5 connector. The open-frame metal housing allows for a compact form factor while providing an effective shielding.

5.2 The hardware communication

As the hardware environment evolved into a distributed system with three different processing units, the communication between the single units becomes a major issue of the development process.
The most important communication links exist between mouse- and main unit as well as between touchscreen controller and touch unit. The mouse unit has to transmit the decoded data of the hosted USB mouse to the main unit while the data from the touchscreen have to be reported to the touch unit in an appropriate way. For both connections reliability and real time capability is required.

5.2.1 Serial Peripheral Interface

The connection between the mouse unit and the main unit is realized using Serial Peripheral Interface (SPI), a synchronous serial bus system originally initiated by Motorola. However, there is no explicit definition of the software protocol, just specifications for the hardware interface.
SPI follows a master/slave concept where the transmission is always initiated by the master device. It also provides the clock and selects the slave via a chip select line. The data are transmitted in full duplex mode using two signal lines: Master-Out Slave-In (MOSI) and Master-In Slave-Out (MISO). The word length is restricted to 8 bit as the SPI hardware is realized by two 8 bit shift registers, each on every chip.
The SPI communication between the mouse unit and the main unit corresponds to a single master/slave relation. The mouse unit serves as a master initiating the transmission as soon as a new mouse input is detected. The main unit serves as

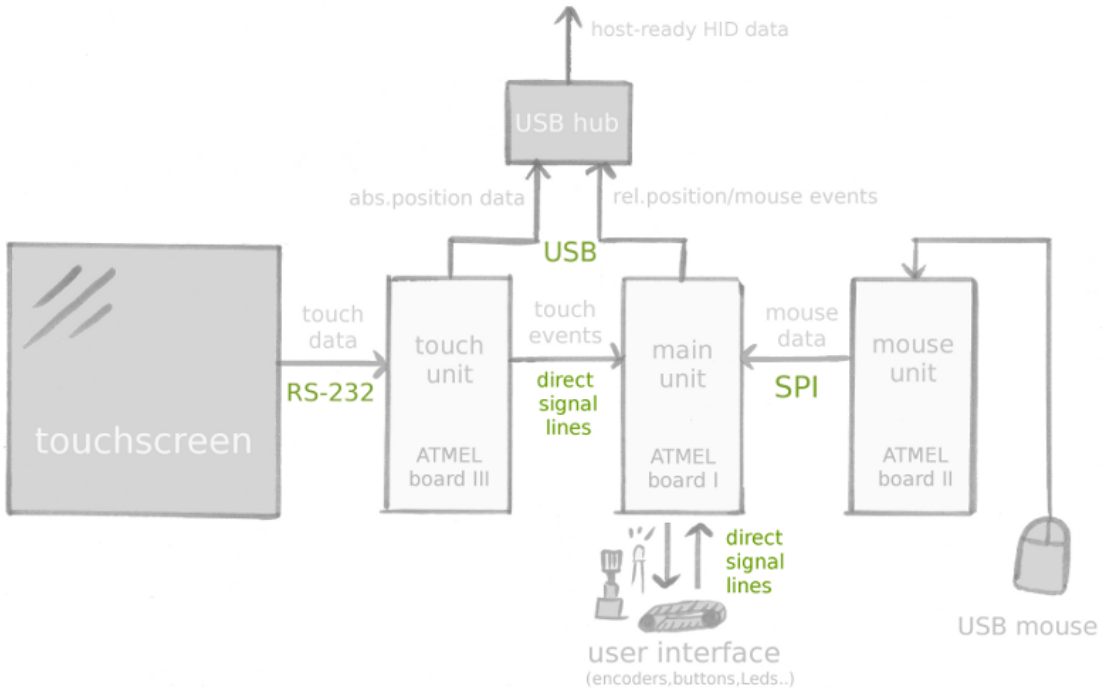

Figure 5.3: Hardware communication schematic

a slave that has to react to newly received data immediately. Therefore, its SPI routine is interrupt-driven.
For the transmission of the mouse data I defined a simple protocol consisting of three consecutive bytes containing the mouse's current delta values of X,Y and of the mouse wheel. They describe the displacement relating to the last reported value. The SPI mouse protocol looks as follows:

[DeltaMouseX][DeltaMouseY][DeltaWheel]

The interrupt-driven slave routine writes the received values into an array. As soon a new data set is transmitted, it is passed to the main routine.

5.2.2 RS-232

For transmitting the touch data to the touch unit the RS-232 interface of the touch-monitor's internal Serial/USB controller is used. Expanding the USART interface

of the development board with an RS-232 receiver, as described in section 5.4.3, enables to connect the ELO touchmonitor directly to the touch unit via a DB9 cable. The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a flexible and very common serial communication bus. As it is used in asynchronus mode, the data are transmitted using two signal lines: TxD ("Transmit Data") and RxD ("Receive Data").
For the transmission the ELO specific "SmartSet" protocol is used [Elo, 2003]. Each touch data frame consist of ten bytes: A lead-in byte, eight data bytes and a checksum byte.

[Lead-in byte][eight data bytes][Checksum byte]

Via the "Smart-Set" protocol the internal controller reports touch coordinate packets to the touch unit. A touch coordinate packet refers to the eight data bytes of every "Smart-Set" frame and looks as follows:

['T'][status][X][X][Y][Y][Z][Z]

The 'T' command indicates a data frame as touch coordinate packet. The status-byte reports the selected touch mode (Initial Touch, Stream, Untouch). It describes if touch packet belongs to an initial touch, to a coordinate transmitted continuously while the touchscreen is being touched (Stream) or to a point where the finger is lifted. In the following, X and Y position are reported using two consecutive bytes each. The Z axis values are set to a default value as no Z data (pressure) can be obtained using resistive 5-wire technology.
The data are received via an interrupt-driven USART routine which writes the incoming data into an array. As soon as a new touch coordinate packet is complete, it is passed to the main routine of the touch unit.

5.3 The system software

5.3.1 Development environment

For the software development I use Atmel's official development environment "AVR Studio 4" in connection with the free AVR-GCC compiler for C which is part of the open-source WinAVR suite[1]. For programming the hardware Atmel's In-System Programmer software "Flip" [2] is used allowing to transfer the compiled hex-files to the microcontroller board directly via USB.

5.3.2 USB HID concept

The main idea of USB HID was to simplify the use of PC input devices while enabling opportunities for innovation as new devices require neither custom drivers nor protocols. The HID concept bases on a extensible, self-describing protocol allowing a single HID driver to determine the I/O data structure and the functionality of a device by parsing its HID descriptor. In this way the driver is able to communicate with all devices that conform to the USB HID class specification [USB-IF, 2001a].
Building an USB-compatible HID device requires an hardware driver to manage the USB communication in combination with a set of USB descriptors accordingly to its intended use. The USB descriptors are data structures describing the specific properties of the device such as configuration, interface, endpoints, or its report structure.
Due to the complexity of USB HID firmware development, I implemented the HID functionality using an existing USB framework – the LUFA library.

5.3.3 LUFA USB library

The Lightweight USB Framework for AVRs (LUFA)[3] is an open-source USB library for USB-enabled Atmel AVR microcontrollers. The library is adapted for the free

[1]The WinAVR suite including the GNU AVR-GCC compiler can be downloaded at:http://winavr.sourceforge.net/index.html (accessed 27.12.2010)

[2]Atmel's AVR Studio 4 and the In-System Programmer Flip can be downloaded at: http://www.atmel.com/dyn/products/tools.asp?family_id=607 (accessed 27.12.2010)

[3]The LUFA framework, written by Dean Camera, can be downloaded at:http://www.fourwalledcubicle.com/LUFA.php (accessed 10.07.2010)

AVR-GCC compiler of the WinAVR suite. Beside the library source, the framework contains several applications demonstrating the use of the library. The fact that the author also uses an Atmel AT90USBKey development board enabled a rapid entry into USB programming.

5.3.4 Main unit

The Main unit is the central processing system of the Zeus Control. The software is based on the mouse device application of the LUFA USB library in order to recognize the unit as a HID USB mouse by the host. It generates a HID-conform mouse report containing the relative position data and the appropriate mouse events. The position data are derived from the input of physical controls and the connected mouse. The mouse events are synthesized by coordinating the input events of physical controls, touchscreen and external mouse according to the considerations regarding the operating reliability as mentioned in section 4.2.2.
The position data from the Mouse unit are received via an interrupt-driven SPI slave routine.
Beside that, the main unit is responsible for controlling the user interface and adopts its output according to the chosen interface settings. It contains all required I/O drivers for LEDs, buttons and external signals as well as an quadrature encoder reader/driver to interface the optical encoders of the sliders and the rotary knob. A detailed description of the Main unit's file structure can be found in appendix A.1.

5.3.5 Mouse unit

The Mouse unit serves as an USB host to integrate a standard USB mouse into the Zeus environment. The software is based on the mouse host application of the LUFA USB library using a HID report parser. The HID parser is necessary as a basic approach referring to the standard mouse "boot protocol" is not able to extract the wheel data from the mouse report. Via the parser the HID report of the connected mouse is decomposed and the extracted X,Y, and wheel values are transmitted to the Main unit using an SPI master routine. The button status (left, middle, right) is reported directly via three signal lines. A detailed description of the Mouse unit's file structure can be found in appendix A.2.

5.3.6 Touch unit

The Touch unit implements an USB HID touchscreen driver. The software is based on the mouse device application of the LUFA USB library. It is modified in a way, that the Touch unit is recognized as an “absolute-coordinate” device by the host computer. The HID report includes the received position data of the touchscreen as well as their general dynamic range. Based on this information the operating system’s HID driver scales the position data according to the current screen resolution.
The position data of the touchscreen are received by an interrupt-driven USART slave routine. Beside the position data, which are directly sent to the internal USB hub, the touch status (onset, duration) is reported to the Main unit via two signal lines. A detailed description of the Touch unit’s file structure can be found in appendix A.3.

5.4 Additional circuits

Even if the use of development boards reduced the need for external circuitry significantly, several circuits had to be designed to implement the needed, additonal functionality. In order to ensure a quick and flexible implementation these are assembled on perfboard providing connections for female-to-female jumper cables.

5.4.1 Touch sensors

The touch sensors are used to determine if the user touches one of the neoprene sliders. Therefore, an additional function of the sliders is realized. They become tactile, easy accessible actuator bars for triggering touch events in the *secure mode*. The touch sensitivity is implemented via charge-transfer touch sensors capable to detect touch through an isolator. The used Quantum Research[4] QT113 touch ICs are connected to aluminium plates mounted under the neoprene belts in their entire length. These serve as sensing electrodes whose capacitance is continuously measured by the ICs in order to detect relative changes due to

[4]The Quantum Research Group Ltd. was acquired by the Amtel Corporation in 2008. An alternative to the used QT113 is the Atmel AT42QT1011 (just available as SOT packages (SMD))

touch.
The corresponding circuits contain an integration capacitor providing the reference voltage for the measurement. Beside that, some external wiring is used to set the sensitivity level to “high” and the maximal on-duration to “infinite” via the IC’s option pins. The circuits are connected to the 3.3V supply of the Main unit.

5.4.2 Level converter

As the logic level of the Atmel boards is 3.3V, a level shift is needed as the optical encoders of the rotary knob and the neoprene sliders require a 5V supply. In order to adapt their 0V/5V quadrature patterns to the board level, 74VHc04 Hex inverters are used. Having 5V-tolerant inputs they provide the appropriate 3.3V output levels when powered at 3.3V. The fact that the logic levels are inverted is irrelevant as the rotation is encoded in the phase relation of the signals. Beside that, the output signals of the touch circuits, which are active-low, are inverted in order to use the same input driver for all external signals.

5.4.3 RS-232 Receiver

To receive the touchscreen data via the USART of the Touch unit a MAX3232 IC is used. It is similar to the common MAX232 but suited for 3.3V systems. The IC reduces the RS-232 voltage levels (+3V to +15V “high”, -3V to -15V “low”) to 3.3V TTL levels (+3.3V “high”, 0V “low”). The circuit offers a DB9 connector to directly plug the serial cable of the touchscreen. The external wiring consists of four charge pump capacitors.

5.4.4 Power supply

The sub-systems of the Zeus control operate at different voltage levels: The touchscreen requires 12V, the optical encoders 5V and the Atmel boards operate at 3.3V. As all boards offer an on-board voltage regulation they can be supplied with 12V. The external logic circuits are supplied with 3.3V via the several Vcc/GnD pins of the development boards. The voltage supply for the optical encoders was realized using an 7805 integrated circuit – a linear voltage regulator having 5V output. In order to dissipate the emerging heat the regulator will

be directly attached to the metal housing. To avoid a short-circuit between the TO220 package of the regulator and the housing an isolating thermal pad is inserted while the used screw is covered with an isolating ring.
For reasons of simplicity, safety and space constraints an external desktop power supply was chosen as main 12V supply. Since just the LCD touchscreen has a power consumption of about 50W, the selected supply is able to provide up to 5.5A at 12V.

Chapter 6

How an idea takes shape - the design process

6.1 Sketching - Refining the vision on paper

> *"A sketch is created from current knowledge. Reading, or interpreting the resulting representation, creates new knowledge."*
> Bill Buxton in *Sketching User Experiences* [Buxton, 2007]

Beside helping me to communicate and share my vision as mentioned in chapter 3.1, the practice of sketching had another important function during the process: I tried to use sketching as a thinking tool for externalizing my ideas. It is capable to unveil flaws that were unseen before as well as to reveal new features and possibilities. In this way it helps to refine current ideas and generates new ones. I think sketching allows a better "conversation" with his own thoughts or – as Gabriela Goldschmidt puts is – *"introduces some kind of dialectic into the design process that is indeed rather unique"* [Goldschmidt, 1991]. Putting ideas down on paper as a concrete sketch allows a much closer examination, while the representation can still be provocative and explorative. During the Zeus development several ideas evolved from sketching: Especially the layout of the interface has been refined several times on paper. But also the idea to use the neoprene sliders as touch-buttons to trigger a left mouse click in "save-mode" arose from sketching.

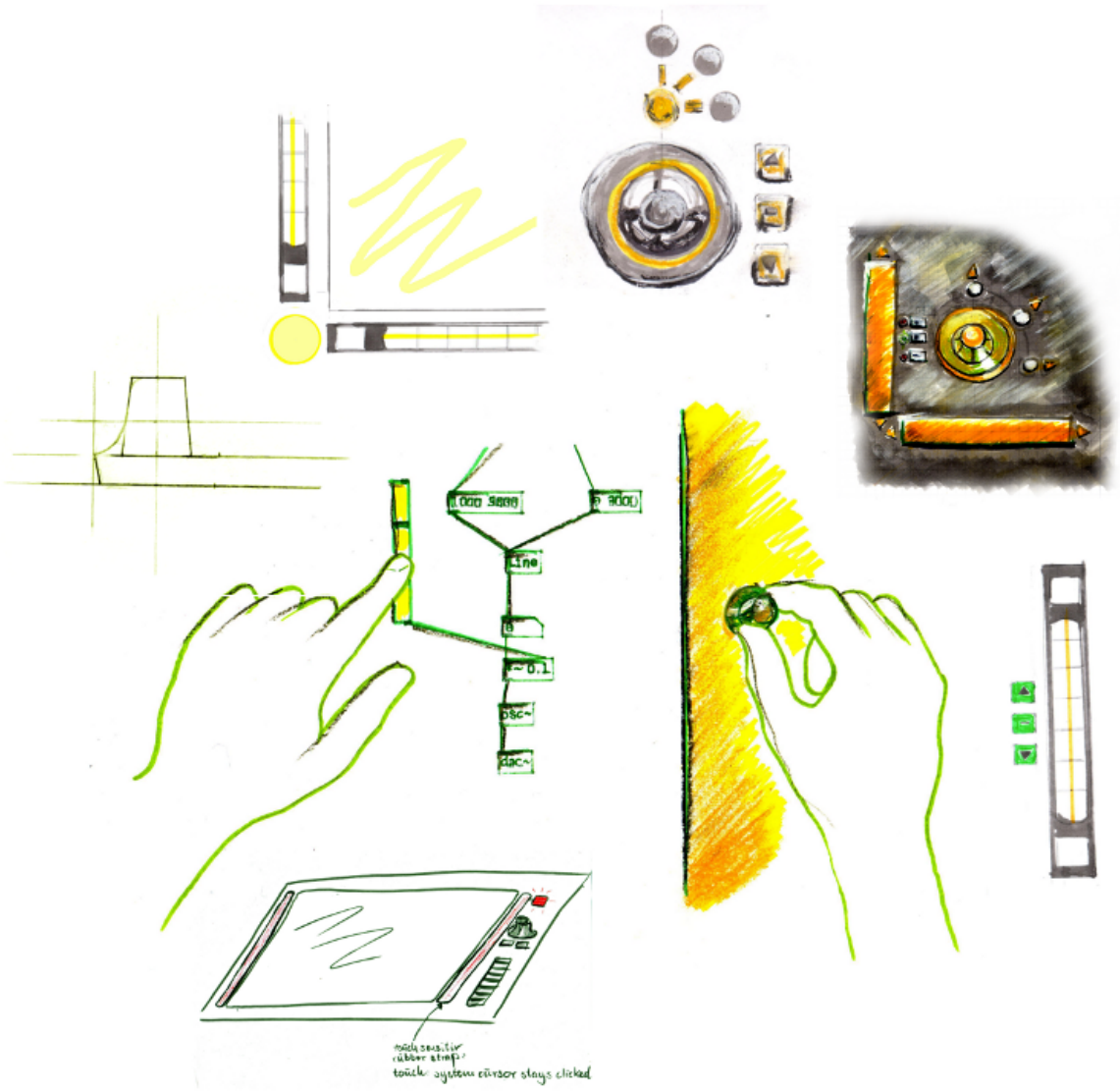

Figure 6.1: Design sketches

6.2 Prototyping

One of the most challenging aspects of the practical work is that the final implementation should be a fully functional device which directly evolves from series of prototypes. Due to the “one-man” development situation of my diploma thesis it is not possible to implement several different prototypes in hardware as it may be done in a company. Therefore, I apply the strategy of *evolutionary prototyping* [1] where the initial prototype is developed, evaluated and refined continuously into the final system. I aimed at a successive implementation of functional subsystems without restricting the overall system to early. This enables to incorporate the user feedback into the next revision of the prototype. The intention is to balance between the prototype as basis for constructive feedback and its progress towards becoming a fully functional device.
Throughout the design process the prototype has gone through several stages having different objectives. In the following, I outline the most important stages.

6.2.1 The initial prototyping environment

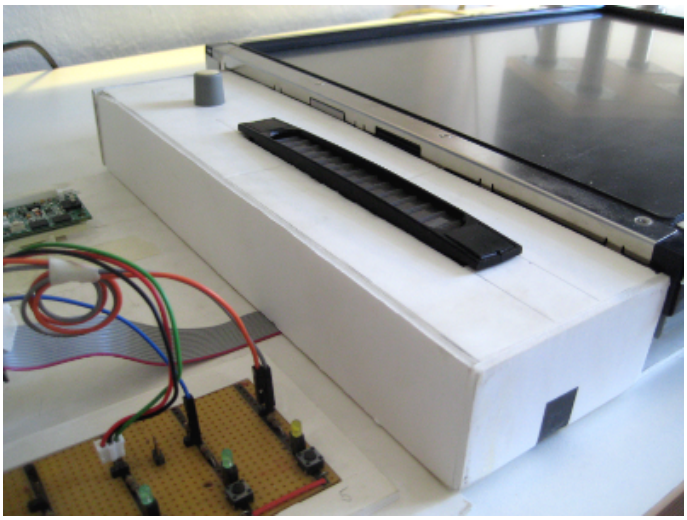

Figure 6.2: The initial prototyping environment

As a starting point for the practical development served a basic hardware environment (fig. 6.2) consisting of one ATMEL board, an optical encoder, an endless-

[1] Even if this term is more common in the area of software development, it provides an accurate description for the chosen prototyping approach

belt controller and several buttons and LEDs assembled on perfboard. The primary objective of this setup was to get used to the hardware platform, the programming environment as well as the USB library. Beside that, essential functions of the firmware have been implemented or adapted: Generic drivers for external buttons and LEDs as well as for the quadrature encoders have been integrated and tested. In this way technical foundations have been laid while facilitating the further development as the basic I/O functionality could be implemented in the following by adapting the generic drivers.

6.2.2 The first prototype

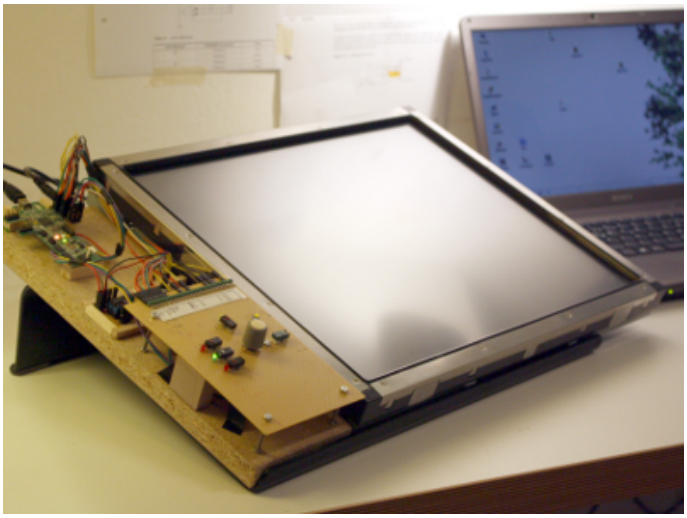

Figure 6.3: The first prototype

The first environment arranged as a real prototype (fig. 6.3) featured an interim user interface providing a rotary encoder as well as buttons and status LEDs for direction and cursor speed. This could be placed freely to both sides of the touchscreen. Via the LUFA library the basic USB HID functionality has been implemented: The ATMEL board, which should become the Main unit of the later system, was able to “mimic” a HID mouse and send the decoded rotation data as relative mouse movements to the host.
In this way, a first hands-on experience became possible while the touchscreen was operated by a modified Windows driver, realizing the intended functionality on operating system level. This basic functional setup provided a first fundamental clue concerning the user interface: It became clear that if a mouse should be

integrated into the controller's work flow as intended, it is beneficial if the physical controls and the mouse do not need to be operated by the same hand. From the perspective of a right hander, the work flow is more efficient if the right hand is able to switch between touchscreen and mouse while the left one operates the physical controls. This allows the user to combine the physical controls with either touchscreen or mouse in a pleasant way.
As a consequence the user interface has been placed to the left side of the touchscreen inducing the first major modification of the current conceptual representation (fig. 3.1, p.19).

6.2.3 The user testing environment

Figure 6.4: The user testing environment

In the next revision of the prototype the second ATMEL unit has been integrated into the system providing USB mouse host functionality to incorporate mouse data into the Zeus environment. From the user's perspective, the major novelty has been the introduction of neoprene sliders running along two sides of the screen. The motivation for the development of these specific physical controls is described in the following section. As now all relevant components of the user interface were available, a prototype setup could be arranged suitable for comprehensive user testing. In order to provide a flexible test environment, the rotary knob and the sliders were not attached to the screen enabling different arrangements. The related user test is described in section 6.4.

6.2.4 The first plug & play version

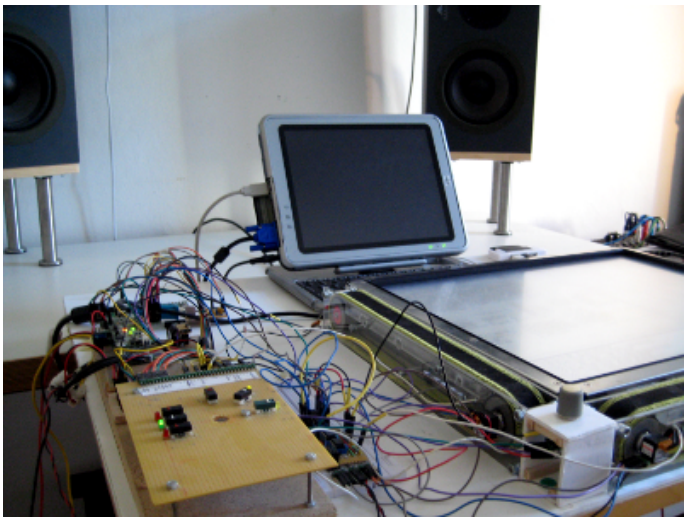

Figure 6.5: The first plug & play version with slate PC running Linux

While during the user testing the touchscreen still was controlled by a modified driver application for Windows, the prototype shown in figure 6.5 was able to be used without a specific OS driver. As the focus of this revision was entirely put on the technical aspects, the user interface has not yet been adapted to the findings of the testing session. However, from the technical point of view this version was a major step forward. Due to the integration of the third ATMEL board into the system implemented as an absolute-coordinate HID transmitting the touchscreen data, the controller became a true plug & play, cross-platform compatible device.

6.2.5 The alpha prototype

The last revision that is presented within this work is the alpha prototype: It is a fully functional version that has been refined according to the user feedback concerning functionality and lock & feel. The final user interface is implemented by reference to the findings as described in section 6.5. Beside that, all electrical components are grouped and installed on aluminum profiles mounted around the display. As a result, the alpha prototype is a ready-to-use, open-frame implementation of the Zeus Control taking into account the results of the entire design process. Remaining tasks are the design of an appropriate metal enclosure as well as the fine-tuning of the control settings in collaboration with the users.

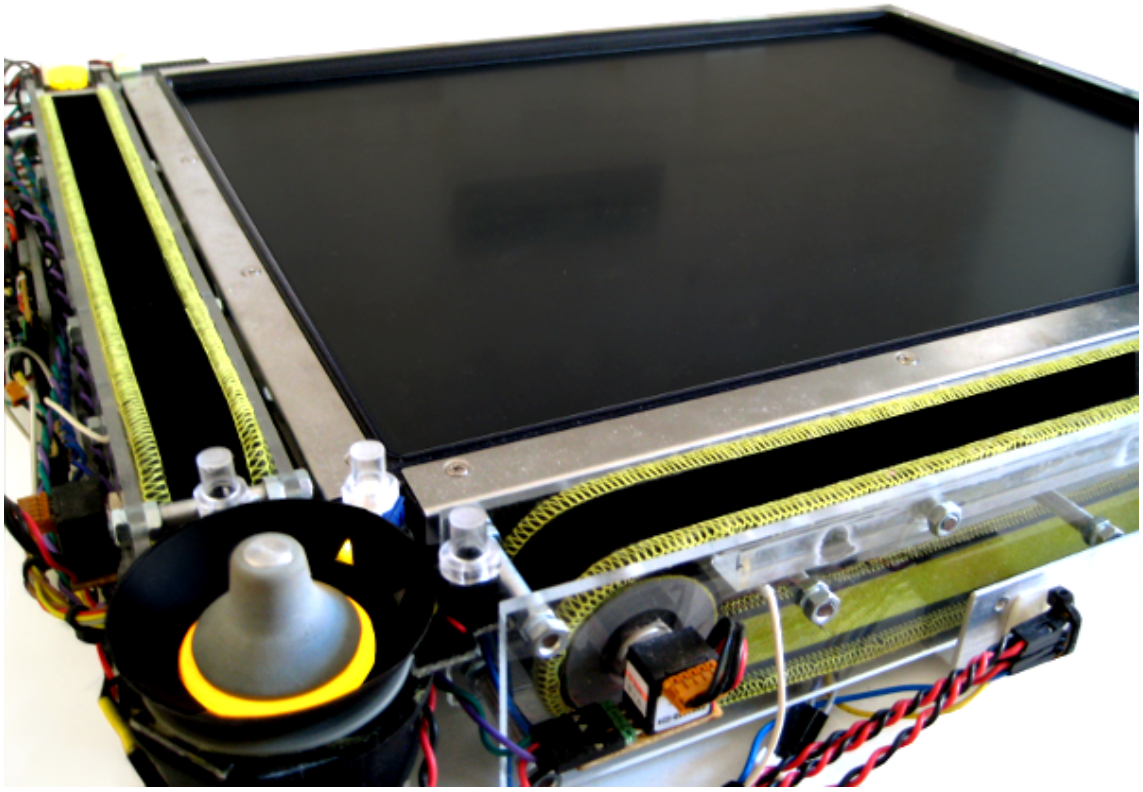

Figure 6.6: The alpha prototype

6.3 Case study: The neoprene sliders

The development of the neoprene sliders is intended to serve as a representative example of how a certain idea can evolve throughout the design process.
The basic idea was to provide an input device that combines a relative control behavior with the operating feel of a fader – an input device used for precision demanding musical tasks since almost the beginnings of music technology.
Looking at the first conceptual sketch, it becomes obvious that the initial vision of the controller was significantly influenced by the most common fader application – a mixing desk. The physical controls resemble the “classical” arrangement of a mixing desk’s channel strip. At this point it seemed to be a self-evident and logic layout.
Intending to implement the controller’s user interface in such or a similar way, I searched for a commercial endless-belt controller. Surprisingly, there was just one implementation available on the market: The Penny+Giles PGF7000. As this company has a very good reputation in the field of audio technology as a manufacturer of high-quality faders for mixing desks, I was looking forward to use it for the controller. Unfortunately, the product did not match my expectations: The

Figure 6.7: Penny+Giles PGF7000 Digital Belt Controller

belt was made of stiff plastic instead of a soft, rubber-like material I expected on account of the product images. This results in an unpleasant tactile feel when handling the belt, intensified by its non-uniform friction and noisy operation.

Even if the controller provides important clues like *resistance* and *guidance* – both basic requirements for precise, spatial control tasks [MacLean, 2008] – these are significantly diminished by its unpleasant tactile and operational characteristics.
As one of the main reasons for using physical controls was to bridge the touch screen's lack of tactile feedback, I was not willing to accept these shortcomings in terms of haptics.
Therefore I decided to develop an endless-belt controller by myself with the aim to enhance the haptic characteristics in comparison with the purchasable implementation. Since the belt properties seemed to be the key element regarding the haptic properties of such a control, the selection of its material was an important aspect: After testing several different synthetic rubber materials, I found that neoprene is very well suited for realizing the belt: The material itself provides a strong *affordance*[2] to touch it. Due to its soft and responsive surface, it provides a pleasantly tactile counterpart to the rigid display of touchscreen. Beside that, its "stickiness" enables precise, fine-grained control movements while supporting the spatial guidance of the control. The user *feels* in control.

Figure 6.8: Neoprene belt prototypes

The belts are made out of two neoprene fabric strips sticked together using a special adhesive. The sides are hemmed with a nylon band reinforcing the neoprene belts while reducing their stretchability. This was necessary to reduce slippage between the guidance pulleys and the belt. The guidance pulleys are made out

[2]The term affordance refers to the physical characteristics of an object: If the affordance of an object corresponds to its intended function, efficiency and ease of use can be increased significantly. The term "affordance" was originally coined by the perceptual psychologist J. J. Gibson in 1977. It was introduced to the field of design by Donald Norman in his seminal book "The Psychology of Everyday Things" in 1988 (Reprint: "The Design of Everyday Things"[Norman, 2002])

Figure 6.9: Final belt version with pulleys

of solid plastic turned on a lathe. To increase the friction between pulleys and the slick bottom side of the belt (jersey-fabric laminated), they are covered with an outer neoprene coating. Between the pulleys, the belt is running on a satined aluminum track serving also as an electrode for the touch sensor ICs. To ensure an appropriate input resolution, high-quality optical encoders were used providing 64 pulses per revolution. In contrast to the commercial solution, which size

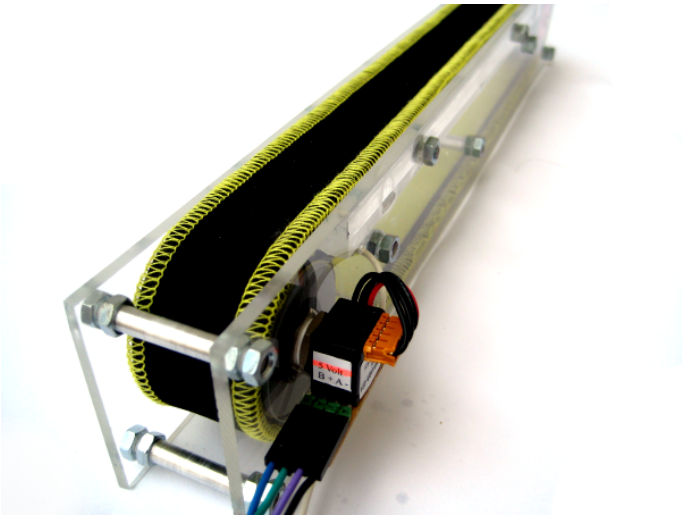

Figure 6.10: Optical encoder attached to pulley

corresponds to a conventional audio fader (100mm fader-length), the custom belt controller has been increased in size (220mm length) according to its function as

assignable “master” control: Due to the long fader path at high resolution, very precise control tasks can be performed. The decision to integrate two endless-belt controller along the edges of the screen, will be addressed in section 6.5 relating to the user interface.

Figure 6.11: Finished endless-belt control

6.4 In dialog with the user: Feedback & Testing

Figure 6.12: User testing

Based on the user testing environment shown in section 6.2, I conducted a feedback session to clarify important functional and ergonomic aspects with regard to the final revision. It was carried out as a qualitative, “hands-on” feedback session with users corresponding to the controller’s fields of application being computer music/electronic composition, live electronics and the studio. The user testing environment was set up in the institute’s production studio in order to control relevant software applications (Samplitude, Max/Msp, Pure Data). I prepared a checklist containing aspects I aimed to ascertain within this user session. One of the most important goals was to determine the specific layout of the user interface through hands-on experience. Moreover, setting options, (e.g. cursor speed/direction) and additional technical features (pedal, casing) were to be defined. The correspondent feedback was gained by observing and interviewing the users while handling the controller. Beside that, the session evolved into a intensive discussion providing constructive feedback. The most important findings being essential for the final revision of the system will be covered in section 6.6.

6.5 Case study: The knob

Similar to the endless-belt controller, the knob was developed in an iterative, experimental process. Driven by the question how to design the shape of a knob in order to *afford* its usage, i created several prototypes. Throughout the iterations, it became apparent that I prefer a bell-shaped form which is adapted to the grip positions of thumb and forefinger. Even if the knob was designed entirely according to my personal conception, several people favored its design when using it during informal tests. In order to “compete” with the tactile affordance of

Figure 6.13: The knob prototypes

the endless belt controls, the knob was turned on a lathe out of solid rubber. A lathe-turned aluminum rod was inserted as a shaft. The knob is attached on an illuminated plastic cylinder which is connected to the axis of the rotary encoder.

Figure 6.14: Assembled rotary encoder module

6.6 Refining the user interface

The final revision of the user interface was driven by findings of the feedback session while applying several design principles.
Since the first conceptual sketch, where the arrangement of the controls resembled a mixing desk's channel strip (fig. 3.1, p.19), the interface layout has been fundamentally changed: Beside relocating the controls to the left side as reasoned in section 6.2, a second endless-belt control has been added alongside the lower edge of the touchscreen. This ensures that the slider positions correspond directly to their function: Considering a touchscreen as a two-dimensional field, the control behavior of each endless belts is mapped to X and Y axis, respectively. In this way a good *mapping* [3] can be achieved as the controllers' effect is consistent with its spatial properties according to natural expectation.
As a rotary knob has no defined direction property, a natural 'spatial' mapping can not be used to indicate the direction of its control data. As a direction assignment is necessary anyway, the rotary knob enables to choose control in X,Y and diagonal direction. To support the user handling this unavoidable one-control-multiple-effect relationship, the active direction is indicated using LED arrows providing a strongly visual, direction-dependent clue.
In the feedback session, the general need for the rotary encoder when having X and Y belt controllers and the mentioned multiple-effect relationship was questioned and discussed: The users agreed to keep the rotary knob as well as its multiple-direction functionality indicated by LED arrows providing a visual, direction-dependent clue. The default direction of the rotary knob's control movement is diagonal as this is appropriate for most virtual controls: In this way, a virtual knob or vertical slider - reacting on y-movements - can be controlled just as a horizontal slider reacting on x-movements.
Also in terms of the knob's position, the users were in agreement: Trying out different layouts, the intersection point between the two endless-belt controls appeared as the most suitable position for the rotary knob. In the following – due to terms of better handling – the need to lower the knob a bit in relation to the control panel has been identified.

[3] The seminal publication on mapping is Donald Norman's book "The Design of Everyday Things" [Norman, 2002]. It also covers the principles of *feedback* and *visibility* being addressed in the following.

Another important issue of the user interface is providing appropriate feedback about the user's action and the current system status. Concerning this, the most crucial information is the current mouse-event state. As mouse-events can be triggered by touchscreen interaction as well as via the physical controls, visualizing the current state contributes to the user's mental model of how the system works and her/his feeling of being in control. Therefore the mouse-state is clearly indicated by a red LED field in the left upper corner. In the testing session, the users evaluated this visual feedback as very helpful being apparent without being distractive.
Beside the LED field, the different touch modes can be selected via illuminated buttons visualizing the current state of the device (green = "save-mode", red = "stream-mode" [continuous input]). The touch mode buttons were placed clearly visible next to the LED field. In order to avoid that the user covers up some part of the screen while accessing them (*screen coverage*), they are shifted to the left side.

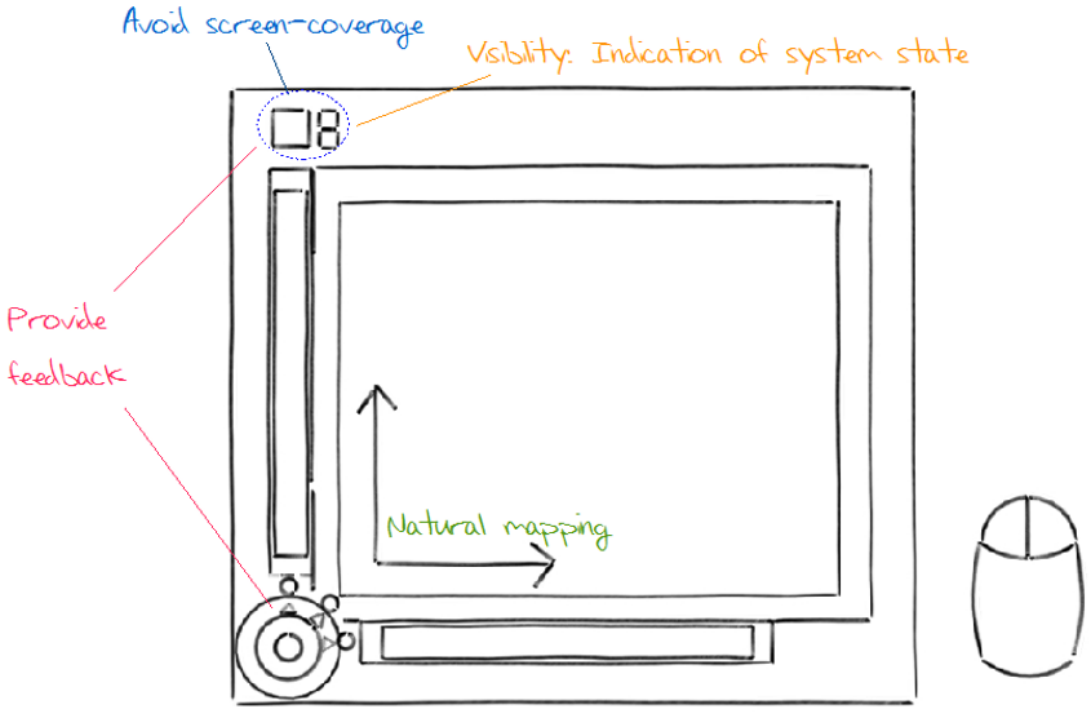

Figure 6.15: The revised user interface

Chapter 7

Conclusion

7.1 Prospects

The scope of this work covers the development of the Zeus Control from the initial idea up to the alpha prototype – being a ready-to-use, open-frame implementation taking into account the results of the entire design process. In order to make the controller ready for use at the IEM, the alpha prototype has to be assembled into an appropriate casing. It is intended to built a custom aluminum enclosure with a combined support/carry handle allowing an adjustable inclination.

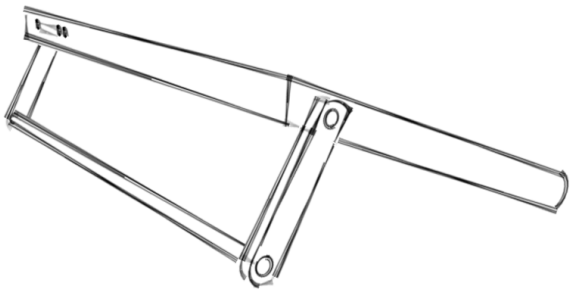

Figure 7.1: Enclosure concept

A final step will be to establish the controller as a new alternative for sound control within the institute: In a close cooperation with the system's future users control settings like the touch modes or the physical controls' behavior will be adapted to their intended field of application.
Beside that, the on-board documentation on the controller's internal flash drive, which contains all the technical documents relating to the device, will be extended with a Quick Start Guide for novice users.
The way it looks right now, the first application using the Zeus Control will be a newly designed graphical user-interface for the CUBEmixer software[1] at the IEM CUBE[2] being customized to the controller's properties.

7.2 Final remark

Looking at the first sketch being a simple outline of the basic idea and the final prototype vividly illustrates how a vision can evolve in such a process.
It was rewarding to experience what a creative potential can be released sharing one's ideas with others while taking their feedback seriously. Especially as the apparent constraints, which resulted from the real users' demands, became the main actuators for creativity.
Also the impact of using methods in order to continuously refine one's conceptions was quite remarkable to experience.
In conclusion, generating new ideas and finding solutions for problems – some even dismissed as unfeasible before – within an overall process driven by existing and newly gained knowledge, constructive feedback and applied methodology was a very challenging and exiting overall experience.
I hope that the final result contributes to the available music technology at the IEM with regard to intuitive sound control.

[1] The CUBEmixer software is a open-source, real-time multipurpose mixing and mastering tool for multichannel loudspeaker systems written in PureData.
Project page: http://ambisonics.iem.at/xchange/products/cubemixer
[2] The CUBE is a medium-sized concert hall at the IEM enabling the reproduction of ambisonic soundfields of at least 3rd order.

Bibliography

[Atmel, 2006] Atmel (2006). *AT90USBKey Hardware User Guide*. Atmel Cooperation.

[Atmel, 2009] Atmel (2009). *AT90USB646/647/1286/1287 Datasheet, Revision K*. Atmel Cooperation.

[Buxton, 2007] Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann Publishers.

[Elo, 2003] Elo (1993-2003). *SmartSet Touchscreen Controller Family Technical Reference Manual, Revision B*. Elo TouchSystems, Inc.

[Fiebrink et al., 2009] Fiebrink, R., Morris, D., and Morris, M. R. (2009). Dynamic mapping of physical controls for tabletop groupware. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 471–480, New York, NY, USA. ACM.

[Goldschmidt, 1991] Goldschmidt, G. (1991). The dialectics of sketching. *Creativity Research Journal*, 4:123–143.

[Löwgren and Stolteman, 2004] Löwgren, J. and Stolteman, E. (2004). *Thoughtful interaction design*. MIT Press.

[MacLean, 2008] MacLean, K. E. (2008). *Reviews of Human Factors and Ergonomics; Vol4*, chapter 5: Haptic Interaction Design for Everyday Interfaces, pages 149–194. Human Factors and Ergonomics Society.

[Norman, 2002] Norman, D. (2002). *The Design of Everyday Things (Paperback)*. Basic Books.

[Sardonick and Brau, 2006] Sardonick, F. and Brau, H. (2006). *Methoden der Usability Evaluation*. Hans Huber Verlag.

[Schön, 1987] Schön, D. (1987). *Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions*. Jossey-Bass Publishers.

[Sears and Shneiderman, 1991] Sears, A. and Shneiderman, B. (1991). High precision touchscreens: Design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies*, 34:593–613.

[Shneiderman, 1983] Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57 –69.

[USB-IF, 2001a] USB-IF (2001a). *Device Class Definition for HID 1.11*. USB Implementer's Forum.

[USB-IF, 2001b] USB-IF (2001b). *Introduction to USB On-The-Go*. USB Implementer's Forum.

Appendix A

Software reference

A.1 Main unit

Source files

- mouse.c
  This file is the primary source file of the Zeus main unit. Its USB functionality is based on the mouse device application of the LUFA USB library. In the *CreateMouseReport* function all input data (physical controls, touchscreen, external mouse) are combined into an appropriate mouse report according to the chosen settings. Beside that, the file contains a SPI slave routine in order to receive the data of Mouse unit.

- Descriptors.c
  This file contains the USB Device Descriptors of a HID USB mouse and belongs the LUFA library. The descriptors are parsed by the host to determine the specific characteristics and functionality of the device. As a scroll wheel is not part of the standard “boot mouse protocol”, the HID descriptor is modified in order to transmit the mouse wheel data.

- encoder.c
  A quadrature encoder reader/driver written by Pascal Stang in order to interface quadrature encoders with the Atmel AVR-series microcontrollers. The library was adapted to the used AT90USB1287 processor via the encoderconf.h file.

Header files

- zeus_defs.h
  Global defines and macros for the Main unit of the Zeus control
- avrlibdefs.h
  Global defines and macros for AVR applications written by Pascal Stang
- Descriptors.h
  Header file for Descriptors.c
- encoder.h
  Header file for encoder.c
- encoderconf.h
  Configuration file for the quadrature encoder driver. As the encoder library uses external interrupts the defines have to be adapted according to the hardware interrupt lines of the used processor.
- EXTmouse_button.h
  Specific button driver for detecting the different mouse button states of the connected mouse on Port B (Pin 4,5,6).
- Mouse.h
  Header file for mouse.c
- zeus_Buttons.h
  Specific button driver for the Zeus Control. The driver is designed to interface buttons wired in an external pull-down circuit on Port C.
- zeus_Inputs.h
  Specific driver for detecting external signals on Port F (touch state signal of touchscreen, touch sensor signals of encoder and sliders, pedal input).
- zeus_LEDs.h
  Specific LED driver for the Zeus Control for LEDs connected to Port A.

A.2 Mouse unit

Source files

- MouseHostWithParser.c
  This file is the primary source file of the Zeus mouse host unit. Its USB host functionality is based on the mouse host application of the LUFA USB library using a parser to decompose the HID report descriptor report of a connected USB mouse. This approach is necessary as the scroll wheel is not part of the basic, fixed-length report of the standard “mouse boot protocol” supporting just X/Y movement and three buttons. Therefore the LUFA parser was used to identify the features of the connected mouse and how they are represented in the report. This allowed to access the mouse wheel data in order to transmit them to the Main unit.
  The file contains also the SPI master routine to send the extracted values of the movement in X and X direction as well as the mouse wheel input to the Zeus main unit.

- HIDReport.c
  This file processes the HID report descriptor from the attached mouse via the HID parser routines of the LUFA library in order to readout the mouse data.

- ConfigDescriptor.c
  This LUFA file reads in the configuration descriptor and setups the host pipes in order to communicate with the attached USB device correctly .

Header files

- ButtonStatus.h
  A specific output driver for the mouse host unit. The driver sets the pins 4 to 6 on Port B according to the button status of the connected mouse (left, middle, right).

- MouseHostWithParser.h
  Header file of MouseHostWithParser.c

- HIDReport.h
  Header file of HIDReport.c

- ConfigDescriptor.h
  Header file of ConfigDescriptor.c

A.3 Touch unit

Source files

- Mouse.c
  This file is the main source file of the touch unit. It uses the mouse device application of the LUFA library to implement an USB HID hardware touchscreen driver. The main modifications are made in the device descriptor Descriptor.c. As a result, the touch unit is recognized as an "absolute-coordinate" HID device by the operating system. The file contains also an interrupt-driven USART routine to receive the touchscreen data sent via RS-232.

- Descriptor.c
  The file contains the modified mouse report descriptor in order to implement the USB touchscreen functionality. The main changes in contrast to a mouse report descriptor are:

  – X and Y are reported as 16bit values instead of 8-bit (mouse)

  – X and Y are reported as absolute values instead of relative ones

  – Beside the logical min/max values, also the physical min/max values have to be reported

Header files

- touch_status.h
  This file is a specific port driver for the touch unit of the Zeus control. The driver sets the pins 2 and 3 on Port F according to the touch status. Pin 2 indicates the onset of a touch event while pin 3 indicates its duration.

- Descriptors.h
  Header file of Descriptor.c
- Mouse.h
  Header file of Mouse.c

Appendix B

Pre-calibration of the touch controller

The internal controller of the touchscreen has been configured in order to report the finger's X and Y position as 10 bit values to the touch unit. This configuration has been made manually using the SMARTSET program - a small utility program by the ELO company able to communicate with the touch controller via the touchscreen's RS-232 interface.
Within the *Scaling* submenu the *X Low* and *X High* values as well as the *Y Low* and *Y High* values have been adapted in such a way that the reported screen coordinates range from 0 to 1024. This can be checked using the *Touch testing* submenu showing the data transmitted by the controller. In the following the scaling has to be enabled in the *Touch Mode* submenu. Actually, this pre-calibration should be a one-time task as the scaling has been saved to the nonvolatile RAM of the touch screen controller via the *Load/Save Setup*. Just if – for whatever reasons – this saved pre-calibration data become incorrect or deleted, a new configuration of the touch controller is necessary.
Even if it would be possible to directly communicate with the controller via ASCII commands, using the SMARTSET program for the described calibration task is much easier. For a re-calibration the touchscreen's RS-232 output – normally connected to the Zeus Control's touch unit – has just to be connected to a computer running SMARTSET. Unfortunately, SMARTSET application is a DOS-based program, that is – on my inquiry – not currently planned to be updated. The SMARTSET program as well as the SmartSet Technical Reference Manual

can be found on the Zeus Control’s internal USB flash drive.