Large Display Interaction Using Mobile Devices

Vom Fachbereich Informatik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Jens Bauer

Datum der wissenschaftlichen Aussprache: 24.7.2015
Dekan: Prof. Dr. Klaus Schneider

Prüfungskommission/Berichterstatter:
apl. Prof. Dr. Achim Ebert
Prof. Dr. Hans Hagen
Prof. Dr. Bernd Hamann
Prof. Dr. Klaus Schneider (Vorsitz)

D 386

Acknowledgements

The completion of this thesis would not have been possible without the support from my supervisors, Prof. Dr. Hans Hagen, Prof. Dr. Bernd Hamann and apl. Prof. Dr. Achim Ebert. Their constant guidance helped me throughout the work for this thesis.

Furthermore I want to thank all my colleagues in the workgroups I worked with, in both TU Kaiserslautern and UC Davis, for providing a great workplace, to give motivation and new ideas. I hope I was as helpful to you as you were to me. A special mention goes to Ragaad Al Tarawneh, who was working with me on the ViERforES project for over two years. And another special mention goes to Mady Gruys and Roger Daneker for their kind support.

I want to thank my collaboration partners, in the ViERforES project from the Software Engineering Research Group: Processes and Measurement, the Software Engineering Group: Dependability and the Robotics Group in the TU Kaiserslautern, and the nice people of the Volkswagen company I have the opportunity of working with.

And without the help of my family and friends, this thesis would have been a lot harder to complete. Thank you to my parents for their never-ending support, my children Ari and Matti for cheering me up and being the best children in the world. I want to thank Anja Walter for the support on the idea and the realization of my doctoral studies, Marika Dienes and Daniel Walter for the nice weekend workgroups, Manuel Seibel for always having a word and an open ear.

Abstract

Large displays become more and more popular, due to dropping prices. Their size and high resolution leverages collaboration and they are capable of displaying even large datasets in one view. This becomes even more interesting as the number of big data applications increases. The increased screen size and other properties of large displays pose new challenges to the Human-Computer-Interaction with these screens. This includes issues such as limited scalability to the number of users, diversity of input devices in general, leading to increased learning efforts for users, and more.

Using smart phones and tablets as interaction devices for large displays can solve many of these issues. Since they are almost ubiquitous today, users can bring their own device. This approach scales well with the number of users. These mobile devices are easy and intuitive to use and allow for new interaction metaphors, as they feature a wide array of input and output capabilities, such as touch screens, cameras, accelerometers, microphones, speakers, Near-Field Communication, WiFi, etc.

This thesis will present a concept to solve the issues posed by large displays. We will show proofs-of-concept, with specialized approaches showing the viability of the concept. A generalized, eyes-free technique using smart phones or tablets to interact with any kind of large display, regardless of hardware or software then overcomes the limitations of the specialized approaches. This is implemented in a large display application that is designed to run under a multitude of environments, including both 2D and 3D display setups. A special visualization method is used to combine 2D and 3D data in a single visualization.

Additionally the thesis will present several approaches to solve common issues with large display interaction, such as target sizes on large display getting too small, expensive tracking hardware, and eyes-free interaction through virtual buttons. These methods provide alternatives and context for the main contribution.

Keywords: Large Displays, Human-Computer-Interaction, Mobile Devices

Zusammenfassung

Große Bildschirme werden wegen fallender Preise immer beliebter. Ihre Größe und hohe Auflösung helfen bei Kollaboration und sind in der Lage sogar sehr große Datensätze auf einem Blick anzuzeigen. Durch die wachsende Anzahl von Big Data Applikationen wird dies zusätzlich interessant. Allerdings stellen die Größe und andere Eigenschaften großer Bildschirme neue Anforderungen an die Interaktion mit diesen Geräten. Das sind, unter anderem, Probleme, wie Skalierbarkeit mit der Anzahl der Benutzer, der Vielfalt der Geräte, was für den Benutzer einen erhöhten Lernaufwand bedeutet, und andere.

Durch Benutzung von Smartphones und Tablets als Eingabegeräte für große Bildschirme, können viele dieser Probleme gelöst werden. Da sie allgemein verfügbar sind, können Benutzer ihre eigenen Eingabegeräte mitbringen. Dieser Ansatz skaliert sehr gut mit der Anzahl der Benutzer. Solche Mobilgeräte sind einfach und intuitiv zu bedienen und erlauben den Einsatz neuer Interaktionsmetaphern, da sie eine große Bandbreite an Ein- und Ausgabemöglichkeiten besitzen. Z.B. Touchscreens, Kameras, Lagesensoren, Mikrofone, Lautsprecher, Near-Field Communication, WLAN, usw.

Diese Arbeit wird Konzepte zur Lösung der Probleme, die von großen Bildschirmen gestellt werden, präsentieren. Wir werden die Gültigkeit des Konzepts zeigen, dann eine universell einsetzbare, eyes-free Technik zeigen, mit deren Hilfe Interaktion mit jeder Art großer Bildschirme, ohne den Einsatz spezieller Hard- oder Software möglich ist, und damit die Einschränkungen der speziellen Ansätze aufheben. Diese Technik wird in einer Anwendung benutzt, die auf großen Bildschirmen aller Art lauffähig ist, in 2D und 3D. Durch spezielle Visualisierungstechniken wird es möglich 2D und 3D Daten in einer gemeinsamen Visualisierung darzustellen.

Zusätzlich wird in dieser Arbeit mehrere Ansätze präsentiert, mit denen allgemeine Probleme der Interaktion mit großen Bildschirmen, wie z.B. zu kleine Elemente auf dem Bildschirm, oder teure Tracking-Hardware, gelöst werden. Diese Methoden zeigen Alternativen und Kontext für die Hauptbeitrag dieser Arbeit.

Schlagwörter: Große Bildschirme, Mensch-Maschine-Interaktion, Mobile Geräte

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large displays are a common sight nowadays, due to the price drops for hardware in the last years, the increased capabilities of displays in general and large displays especially. Immersive systems featuring large displays became more common and stereoscopic displays are available on the consumer market. Especially in collaborative environments they are very useful as their content is clearly visible to all users. In engineering domains, such as the automotive area, they are integral part of the design process.

The screen estate provided by large displays can be used in various ways. Visualizations can provide several views of the same data at once. This can be done in different levels of detail to provide Overview and Detail, where details of a subset of the data is seen in one view and a less detailed overview of the full dataset in another one. Focus-and-Context views can be used where details are directly presented in a part of the view (e.g., fish-eye-techniques). On large screens it is no problem to have a large details area and simultaneously a lot of non-detailed context screen estate. Instead of changing detail levels, different aspects of the data can be shown in different, linked views to allow easy comparison. Even if only presenting data in a single view without making special use of the large display, the amount of data that can be seen at a time is larger than on a normal screen. As the user is usually able to freely move around in front of the display, a simple but very intuitive interaction method is already provided.

Large displays come in different technological setups. Probably the simplest large display is a projector as it can scale its picture to almost any size, only

limited by its luminance and available space. Unfortunately the resolution does not scale, limiting its usability. Large TV screens of up to 98 inches diagonal size and more (at the time of writing) are marketed as large displays in the mainstream consumer market. With resolutions of 4K (e.g., 4096 x 2160 pixels) they can provide a lot of data.

A combination of multiple of these devices in a matrix creates a tiled display wall. It can provide an even larger display area with a very high resolution. Theoretically, there is no limit to what size these display can scale, but practically setups larger than 4x4 are rare, due to the increasing maintenance effort needed.

By using 3D monitors or two projectors together with an overlaying mechanism (such as polarization, or fast switching), it is possible to provide a stereoscopic view and thus enter the domain of Virtual Reality. The 3D impression from stereoscopic displays can give a better understanding of three-dimensional data. Changing the physical layout of such a system to form a small room with display walls, a so-called CAVE is created. It is a fully immersive environment, where users stand right inside the presented scene while viewing and interacting with it.

1.1 Issues and Research Questions

Despite all the benefits of large displays, they also have their issues. More than 10 years ago Swaminathan and Sato [98] already came to the conclusion, that large displays need different interaction techniques than traditional workstations.

Due to the diversity of large display systems, there is no standard when it comes to input devices or metaphors. Depending on planned use cases, target audiences and also available budget different devices are used. Examples include traditional desktop input devices, such as computer mice, trackpads, and keyboards. Specialized devices comprise tracked flysticks (6-degrees-of-freedom (DOF) pointing devices), gesture detection, VR gloves, etc. This requires users of multiple systems to familiarize with multiple input devices [26].

Another important research problem is the scalability to the number of

users. Since large displays are often used in collaborative environments, granting interaction possibilities to each user (simultaneously) is important. Traditional devices, such as keyboard/mouse or specialized devices, such as flysticks, are no practical solution to this problem. Specialized devices tend to cost a lot of money and often come with their own restrictions (e.g., maximum number of flysticks a system can track). Keyboard and mouse combinations need a solid base to be used, and thus are limited by the amount of space in front of the display. In environments with focus on the user being able to physically move around, such as CAVEs or wall-sized displays, the solid base effectively denies the user of this ability. Additionally, support for multiple keyboard, mice, etc. is limited on major operating systems.

Scalability issues of large displays include task/space management problems, when organizing available space on the screen becomes troublesome. With increasing pixel-count, distances on the screen also increase. This means the user has to decide to either take longer to move a pointer from one side of the screen to the other, or to increase pointer speed. However increasing pointer speed makes targets (such as icons) on the screen harder to hit, according to Fitts' Law [49]. This is called distal-access-problem [91]. Furthermore, the pointer on the screen might get smaller (relatively to total screen size) as well, leading problems detecting its position [85, 91].

There is also an issue for visualizations in general when to be presented on different display setups, especially on stereoscopic *and* non-stereoscopic displays. Depending on the nature of the data's representation, which is usually 2D *or* 3D, one device type is usually better suited. 2D representations can be shown on stereoscopic 3D displays by just ignoring the third dimension. However, this does not make use of the capabilities of these displays. 3D visualizations presented on standard 2D displays on the other hand lose all depth cues, severely impacting the usability. When dealing with combined 2D/3D representations, it might not even be possible to select a better target display.

In this thesis, we propose to solve these issues using mobile devices, especially smart phones and tablets. The sales figures for these devices increased dramatically in the last few years. They are becoming ubiquitous in our society. It can be assumed, almost every user will have at least one of these devices. Thus, the problem of limited amount of input devices can be solved

by a Bring-Your-Own-Device (BYOD) policy. Apps on the handheld devices can be employed to interact with a large display, through WiFi, Bluetooth, NFC or other means. Since the devices have several different input and output capabilities, such as cameras, accelerometers, compasses, touchscreens, microphones, speakers and more, they can be a more versatile device, than most other, standardized, input devices in use today.

As an added bonus, mobile devices can be used as mobile data storage, means of identification (and authentication token) and provide a personalized interaction experience for each user. Even between different applications, smart devices can provide an uniform way of interaction. This is not limited to applications on different setups of large display.

From these issues the following research questions are derived:

- Are smart phones and tablets valid choices as input devices for large display interaction?
- How can smart phones and tablets be used in a generalized manner, to be applicable as input devices, for usage with different applications and use-cases?
- How can common issues with large display usage be avoided, either through interaction or application design?
- Can applications use both stereoscopic and monoscopic displays to their full capabilities?

1.2 Goals of This Thesis

The main goal of this thesis is to validate the usage of smart phones and tablets as input devices for large displays and find ways of interaction depending on the application and usage scenario in question. More detailed this should be achieved in the following ways:

- Find and validate use-cases as proof-of-concept
- Devise a generalized approach that does not rely on a special usage environment, hardware or software, and that does scale well with the number of users
- Develop an example application, making use of that approach, being independent of special hardware and software and that runs under many

different usage environments. That also means it needs to be scalable, based on display size.

There are also additional goals, that are not the main focus of work, but help solving issues similar to the main goals, provide alternatives, and/or help to provide context for the main goals. These minor goals include:

- Finding solutions to common issues of large displays, such as the distal-access-problem, that occurs on 2D GUIs when the target size decreases in relation to the distance a mouse cursor has to travel to reach it as the total display size increases.
- Evaluating/Rating of existing techniques for large display interaction, especially in regard to using mobile devices

1.3 Contents and Contribution

In chapter 2, examples of large display setups are given. Application areas, issues and software for large displays in general are stated. For context, research already done in this field is shown and rated. Contents of that chapter was partly published in [21]. It contains an assessment of techniques and approaches proposed by other researchers for employment in Virtual Reality settings. This is also the chapter's main scientific contribution.

An expanded evaluation of specialized test scenarios where a smart phone is used for interaction with a large display already given from our previous work is presented in chapter 3. In two test scenarios we developed novel eyes-free 3D interaction techniques. One employs the touchscreen and accelerometers for 3-degrees-of-freedom (DOF) positioning of objects. The other method uses a joystick metaphor to enable camera movement along a 2D plane in a 3D scene.

By displaying 2D content as seen on the large display on the smart phone, we allow independent examination and editing by multiple users. In contrast to previous work, we do not only present new interaction techniques, but also present an evaluation where the performance these techniques in terms of speed and user acceptance are compared to the performance of the traditional input devices keyboard and computer mouse. The evaluation will show, that most users like the interaction through smart phones and learn quickly how to use

it [20, 24].

The second part of that chapter contains a method specialized for CAVEs, where the user can use the touchscreen of a tablet in order to interact with the VR world presented to them. This allows direct touch interaction without expensive VR gloves, while the usually employed flystick does only allow indirect interaction. On top of that we can even use well-known multitouch gestures for interaction.

The main scientific contributions of this chapter include:

1. A proof-of-concept for the viability of using smart phones and tablets as interaction devices for large display systems
2. Novel specialized interaction methods for 3D object manipulation and navigation, for tagging on 2D objects and for manipulating 2D objects
3. A Specialized interaction method allowing low-cost multi-touch interaction in immersive environments

Chapter 4 contains solutions to common issues in large display interaction. A way of using only the built-in functionality of today's smart phones and tablets to track the position and orientation of the phone is presented. While GPS and similar technologies can track positions on the large scale, the micro-scale position estimation of centimeters needed for this use-case cannot be provided even when GPS data is extended by WPS or similar approaches. Our sensor-fusion approach mixes computer vision and and accelerometer/compass data to avoid the need for expensive tracking equipment and allows usage of tracked interaction methods in low-cost enviroments. Implementation of this work is also described in the master's thesis of Michael Ebel [46].

A second solution addresses pointer handling on large screens. To avoid the distal-access-problem, screen distortion approaches are ususally suggested, such as icons stretching towards the pointer [91]. This has the obvious downside of distorting the view on the screen. Using multi-touch on a tablet programmed to work as a trackpad, users can scale the speed of the pointer as they move it. This avoids the distal-access-problem without causing distortions, as the pointer can be sped up while crossing larger distances and sped down when exact positioning is needed. Through an additional gesture the cursor can be highlighted on the screen, in case the user loses track of it. Nicolas Engel wrote about the implementation in his master's thesis [47].

The main contributions of this chapter are:

1. A solution to the distal-access-problem using scalable pointer speed
2. A way for the user to find the cursor on a large display, should they lose track of it
3. Microscale tracking of smart phones and tablets fusioning built-in sensor data
4. A low-cost tracking solution for large displays

A general method of interaction is then presented in chapter 5. The concept of Marking Menus [69] is used on smart phones and tablets to provide a uniform means of interaction with a variaty of different applications and large display setups. By expanding the original idea of Marking Menus to the capabilities of modern smart devices, it is possible to include not only menu options, but even full 3D interaction into this metaphor. Expert users can employ this method eyes-free, while novice users will have all menu structure visible on the screen of the smart device, eventually becoming experts themselves.

This method does not need special hardware, neither at the large display nor at the mobile device. It can be employed with any touchscreen, and might be applied to wearable devices as well. It can be employed to any application, even on non-large display systems. Therefore it solves the problem of diversity of input devices in the domain of large display interaction. This chapter contains results published in [25] and [26].

The main contributions of this chapter are:

1. A generalized eyes-free interaction method usable in any usage environment with any consumer level smart phone or tablet. It is also possible to employ this approach for any application. The method scales well with the number of users. This solves the problem of diversity of input devices across the different large display setups.
2. An expansion of the Marking Menu approach from [69] for multitouch-capable devices with support for clutching, menu shortcuts and in general a flatter menu structure

Chapter 6 contains an in-depth description of a practical application of this method. The result is a Virtual Reality application running in a multitude of settings. It leverages the collaboration of safety- and hardware-engineers when creating and improving embedded systems, by including and linking both

domain-specific views of the embedded system in one application, instead of each party of engineers having their own view, data, and/or application.

The two linked views feature a novel stereoscopic highlighting technique applied to a node-link diagram. Stereoscopic highlighting uses the depth cues provided by 3D displays to encode ordinal properties of the nodes displayed. This allows to use established 2D layout techniques for the graphs and avoid the occlusion problem that occurs when using 3D graph layout techniques.

To further improve the visual depth cue provided by this technique, a reflection at the bottom of the graph is provided. This application can be controlled by an improved version of the Marking Menu interaction shown in the chapter before. This information and more was published in [7–10].

The main contributions of that chapter are:

1. An approach to combine 2D and 3D content in a single visualization system, scalable to virtually any display size. The application avoids space-management problems, the distal-access-problem and other common issues hindering scalability
2. A visualization method for 2D node-link-diagrams on stereoscopic displays that makes use of the visible depth to display additional information instead of using it for layout purposes
3. An extension to the above visualization method, allowing for better perception of depth, even on non-stereoscopic displays.

Evaluations of these ideas are presented in the correspondent chapter 7. Besides showing the validity of the stereoscopic highlighting and reflection layer approaches (as published in [8, 9]), as well as of the whole system designed. A small case study shows that 3D models should have metadata about their preferred interaction method. This goes further than the meta-data provided by established formats, such as VRML.

In another evaluation we show the influences of various device sizes, button sizes, device orientations and user properties on the interaction with so-called virtual buttons. Virtual buttons are created by applying a regular grid of arbitrary size to the touch area of a device. Since the regular grid can be estimated by users, this creates another eyes-free interaction approach. While this is of interest in current HCI research, the influences of device factors had not been evaluated before. The user group contained small children, to gain

a broader user base for the evaluation. This is interesting as mobile devices are used by more and more children in general, yet sources for the difference of their usage behaviour compared to adults is scarce. This will be published in [22].

The main contributions of chapter 7 are (besides the evaluation for the visualization methods stated above):

1. A case-study reasoning for annotation 3D models for improved interaction
2. An evaluation of a intuitive eyes-free interaction technique, showing the influence of device and user parameters on its usability
3. Evaluating the special properties of children when using smart phones and tablets

We will conclude in the final chapter, give an overview of possible improvements of the presented results, their applicability to other areas and finally summarize the important points of this dissertation.

Chapter 2

Related Work

Before focusing on interaction techniques developed for large displays in section 2.2, we will present an overview of existing technology and architectures for large displays. To give a broad overview to the reader, we will further show typical application areas and common issues with large displays.

2.1 Large Display Setups

A great overview about large displays, their setup, applications, and issues was given in a survey paper by Ni et. al. [85]. Therefore this paper will be the main source of information in this section.

2.1.1 Hardware Configuration

As mentioned in the introduction, there are several types of large displays. Ni et. al. [85] identified six categories of large display technology setups.

CAVE and Derivatives

CAVE is a recursive acronym for CAVE Automatic Virtual Environment. It is a fully immersive environment, built of multiple screens arranged in a room-like fashion. CAVEs consist at least of three walls, i.e. displays, front, left and right. Additionally some installations include screens at the back-side, at the ceiling and/or the floor. Immersion is achieved through synchronized stereoscopic projections onto the screens, typically using shutter glasses. Tracking

is employed for interaction purposes and to provide a perspective-correct view to the user. It is not possible to achieve perspecive-correct views for multiple users. Interaction is usually performed through specialized 6DOF devices, such as tracked flysticks or gloves.

Multi-Monitor Desktop

Called "multimon" in some literature, these setups with a few monitors connected to a normal PC allow more screen estate for consumer level systems. Operating systems support for multiple monitors is available for several years now, and technology for combining several GPUs, such as SLI^{TM}is widely available. Also many if not most modern graphics chips have more than one output port, that can be active all at one time. Combined with the price drops for PC hardware in general and displays especially, the popularity increase for multi-monitor desktops is easily explained. As all screens are driven by a single computer, normal applications can be used on those systems. A downside is the limited scalability, as the number of usable monitors is limited by the number of GPUs in a system and the number of ports available on them.

Tiled LCD Panels

Arranging multiple LCD Panels in a 2D array, possibly using multiple computers driving the panels, solves the scalability issue. The size of these tiled displays is only limited by increasing communication and synchronization overhead when using multple computers. But a more pressing limitation is usually the cost involved and the physical space available for such a setup. Usually the displays are arranged to form a wall, a table or a curved shape. Due to their shape these setups can be used as a single large display with a very high resolution. While the bezels of the individual panels hinder sight, they can also be used to segment the screen into smaller areas.

Projector Arrays

The bezel problem can be avoided entirely by using projector arrays instead of LCD panels. However projectors come with their own drawbacks. Depending on the actual technology used, drawbacks include high maintenance cost, low

brightness or limited geometry control. Projectors also need more space, since there needs to be a distance between the projector and the screen. However projections can be scaled in physical size by simply changing the projection distance. However this requires a recalibration of the projector array, since overlapping areas change.

Stereoscopic Displays

The basic technique for rendering stereoscopic views is to provide the same picture or pixel set from two slightly different viewpoints, Those pixels need to be presented to the users' left and right eye respectively. Usually this is done by having the user(s) wear special glasses, either polarized (passive stereo) or shutter glasses (active stereo). There are also autostereoscopic displays not needing special glasses at all. Generally stereoscopic displays can be set up as described in the categories above. In all cases an important performance issue is the double amount of pixel in the stereoscopic display. Additionally user tracking is needed to provide a perspective-correct view of the stereoscopic data. Tiled Display Walls with stereoscopic capabilities are ususally called Powerwalls, regardless of available tracking.

Volumetric Displays

Volumetric Displays do not create the illusion of three dimensions to the users, but instead create a real 3D image by stacking voxels (3D version of pixels) in all three dimensions. This can be done by creating an array of transparent LCD displays. Only one LCD panel may be active at a time to avoid occlusion issues and each panel needs to be active exactly once during one display frame. To show content at a rate of 60 Hz on an array of 20 displays for example, a total refresh rate of 1200 Hz is needed. This is together with the high voxel count the major limitation in scalability for this technology.

Another way is to use a fast rotating surface on which the image is projected (usually through lasers). By correctly timing the projection with the current orientation of the surface a 3D image can be created.

2.1.2 Rendering and Streaming

For the second regarded point, rendering and streaming, Ni et. al. [85] propose three categories of interest:

Architectures

Molnar et al. [77] propose a taxonomy for parallel rendering algorithms base on when primitives are sorted in the transition from object to screen space.

Sort-first systems partition the display space into disjoint areas. Each area is assigned to one rendering node, which is then responsible for rendering all primitives in that area.

Algorithms falling into the *Sort-middle* category assign each primitive to exactly one rendering node to transform the primitive into screen space. The result is then reassigned to one or more nodes, depending on the screen position.

When each primitive is assigned to exaclty one node and that node renders the primitive completely, the algorithm is classified as *Sort-last*. In this case, it is necessary to combine the pixel-output of all rendering nodes to a complete output.

Data Distribution

Data distribution is classified into two models by Chen et. al. [34, 35].

Client-Server systems consist of a single client on which user interaction and logic is computed. The client communicates with the rendering servers to provide the output. Rendering server can either store local copies of the rendering data, thus only needing data updates every frame. Less complicated systems will send the whole rendering data to the server for each frame.

Master-Slave systems have the same application running on all nodes. The master node is responsible to handle synchronization and state consistency on all nodes.

Software and Application Areas

Common application areas for large diplays are, according to Ni et. al. [85]:

- **Command and Control:**
  Large displays are often used in command and control centers, especially in military, aerospace and telecommunication.
- **Vehicle Design:**
  Especially stereoscopic large displays are interesting for vehicle design. Having the possibility of seeing a planned vehicle in (near) real-life size is very valuable. To achieve similar results expensive prototypes were necessary.
- **Geospatial Imagery and Video:**
  Geospatial data makes good use of the screen estate and high resolution provided by large displays in general. Immersive Environments can present the data as if the user was directly at the size of interest.
- **Scientific Visualization:**
  Visualizations can make use of the space provided by large display in different ways. It is possible to display the same data set multiple times in different linked views, e.g. for focus-and-context applications. Alternatively huge datasets can presented at once, without needing interaction. This can be very useful for multi-user applications.
- **Collaboration and Tele-immersion:**
  Another multi-user application with large displays are the so-called virtual whiteboards. They can be used like a regular whiteboard by several users either at the same time or at different times, as they would use a regular whiteboard. It has the added benefit of change tracking and the availability of multimedia content. Also large display visualizations can be enhanced by allowing remote users to participate. Video streams or even 3D streams can be included in the visualization creating virtual collaboration environments.
- **Education and Training:**
  Large displays also serve as a presentation space for education and training. This is a special case of collaboration, where only one user (i.e., the instructor) will interact with the display in order to provide content to several other, passive, users.
- **Immersive Applications:**
  Immersive Systems can be used for realistic visualizations, especially with

3D data, or for training applications.

- **Public Information Displays:**
  Billboards are getting replaced by electronic versions. While more expensive, it is easier to replace the contents, and they are even able to show animations or cycle through several advertisements. It is possible to realize interactive billboards, where users interested in an advertisement can interact with the billboard, e.g., though a smart phone.

2.1.3 Common Issues

Six common issues with large displays have been found by Robertson et. al. [91]:

1. **Losing the cursor:**
   As screen size increases, users accelerate mouse movement to compensate and it becomes harder to keep track of the cursor.
2. **Bezel problems:**
   Bezels introduce visual distortion when windows cross them and interaction distortion when the cursor crosses them.
3. **Distal information access problems:**
   As screen size increases, accessing icons, windows, and the start menu across large distances is increasingly difficult and time consuming.
4. **Window management problems:**
   Large displays lead to notification and window-creation problems because windows and dialog boxes pop up in unexpected places. Window management is made more complex on multimon displays, because many users try to avoid having windows that cross bezels.
5. **Task management problems:**
   As screen size increases, so too does the number of open windows. As a result, users engage in more complex multitasking behaviors and require better task management mechanisms.
6. **Configuration problems:**
   The user interface for configuring multimon displays is overly complex and difficult to use. When users remove a monitor from the display configuration, they can lose windows as well. Also, different monitors might have different pixel densities; currently support is poor for dealing

with such heterogeneity.

Problems 1–5 have also been identified before by Ni et. al. [85].

2.2 Survey of Existing Techniques

Interaction with large displays is not a new field of research. There are very useful examples of previous work done to use mobile devices, especially smart phones, to interact with large displays. This section contains an overview of the most important approaches and also categorizes them. We also present a rating for these methods in VR environments. The methods shown below are not targeted at VR, since research in Virtual Reality interaction is generally aiming more at specialized input devices, but still methods not tailored towards it, might be usable. In order to classify the results, we will first present taxonomies of input devices and metaphors in section 2.2.1. In the later sections notable research results in this regard will be presented: techniques that base on trackpad-like interaction (section 2.2.2), methods using the camera of the mobile device (section 2.2.3), menu-based approaches (section 2.2.4), gestures (section 2.2.5) and application-tailored methods (section 2.2.6) before concluding in the last section.

2.2.1 Taxonomy and Classification

There are some possibilities to classify the research results presented in the later sections. Quality dimensions of different input device approaches include the actual capability to effectively execute tasks in applications, the volume of control or number of different input vectors (e.g., degrees-of-freedom) and other small factors. In this section we introduce the taxonomy of tasks by Foley, Buxton's input device taxonomy and a few additional criteria. With Foley's taxonomy we can show the capabilities of input devices (and techniques) to execute different tasks. Buxton's taxonomy can be used to measure the amount of control vectors. A few special properties of input methods in the area of virtual reality interaction are not covered by these two taxonomies, which is why we added four additional factors.

Foley's taxonomy of tasks

An old, but still useful taxonomy is from Foley et. al. [50] (as for example confirmed by [16] and [17]). It identifies six different tasks, that cover all uses of input devices. Complex interactions can be decomposed into these six basic tasks.

1. **Position:**
   Set the absolute or relative position of an object in 2D or 3D space.

2. **Orient:**
   Set the absolute or relative orientation of an object in 2D or 3D space.

3. **Select:**
   Select an item out of a list of several items.

4. **Ink:**
   Define a path consisting of one or more positions and orientations. The name Ink refers to the visual line represented by a path.

5. **Quantify:**
   Select a number from a continuous or discrete set.

6. **Text:**
   Enter arbitrary sequences of characters.

Buxton's Taxonomy of Input Devices

Another taxonomy of devices by Buxton [33] might seem appropriate at first. It classifies input devices by their physical properties, like the control agent (e.g., the hand), what is sensed by the input device and in how many dimensions (i.e., this is related to the number of Degrees-of-Freedom (DOF) provided by the device). This is also affected by the number of buttons or similar triggers and modifiers on a device.

This approach does not work well to classify different metaphors and techniques on similar input devices. But it still can be used in some circumstances, when a technique is deviating from the default multi-touch-screen interaction schema. Additionally, as smart phones and tablets are not only input devices,

but also feature output facilities (i.e., the screen, speakers, vibration, etc.), this taxonomy can be extended to the kind of output used (if any) by a certain method.

Other Traits

The taxonomies explained above do not cover all important characteristics. More interesting traits are:

- **Eyes-free interaction:**
  This is a characteristic not very important for traditional input devices. Most of them are eyes-free by design. For example, a simple computer mouse can be operated without any trouble while looking at the screen instead of the mouse. This goes for almost all input devices in common use, with the notable exception of keyboard, at least for inexperienced users. But with the screen on smart phones and tablets and the resulting possibility of displaying content to the user via the input device, the discrimination between eyes-free techniques and those requiring the screen of the hand-held device is important.

- **Tracking:**
  While technically part of Buxton's taxonomy (as the number of dimensions provided), it is especially interesting for collaborative setups to know, if a certain method needs some form of tracking. This is due to the fact, that tracking does not scale really well to the number of users, as most tracking systems have a fixed maximum number of markers to track.

- **Secondary Device/Method:**
  Some methods and techniques might be usable only for a subset of features provided for the main input device. This can either help to provide at least a minimum number of interactivity to users who would otherwise be only spectators, or to have a specialised device, that is only used for a certain interaction and will do this in a better way than a standard input device.

- **Scalability:**
  It is important to measure for each technique, if it is actually applicable to more than one user and if there is a point, where it will perform worse when additional users are employing this technique concurrently. For example, all methods needing a mouse pointer will suffer from this, since it is becoming more and more difficult for the users to distinguish their mouse cursor from the others.

2.2.2 Trackpad-Based Techniques

The most straight-forward method of using a (small) touchscreen is probably to have it emulate a trackpad. This has the advantage of most users already knowing the trackpad metaphor and being able to easily handle it. Of course with this technique none of the advanced capabilities of smart phones and tablets are actually used. Several commercial apps doing this are already available, like *Remote Mouse*[1]. Such apps usually allow to send text to the connected computer. Extending from this, the *ArcPad*[76] can be used as a trackpad, but additionally tapping on any point of the pad will move the mouse cursor to the position on the screen according to the point tapped on the pad. E.g., a tap on the lower left corner will put the mouse cursor to the lower left corner of the screen. Respectively tapping the center of the pad will put the mouse cursor to the center of the screen. This method was created for large display environments, where movement of a mouse cursor might be cumbersome. Unfortunately, users can run into trouble with losing track of the cursor when tapping on the pad. This is especially an issue for users of computers with trackpads, where a tap normally is interpreted as a click instead of a cursor-relocation.

Since these methods are used to control a mouse cursor, they are basically able to complete all 6 of Foley's taxonomy (including text since text can be entered through the keyboard on the mobile device) through a level of indirection. Directly only position, select and text can be supported. Trackpads (including the *ArcPad*) provide 2DOF and one or two buttons. Output facilities of the smart phone or tablet are not used in this approach. It works eyes-free, as

[1] http://www.remotemouse.net/

there is nothing to be displayed at the screen at all and does not need any tracker to be used. But it will not scale well to many users, as multiple mouse cursors will puzzle the users. Depending on the setup and the experience of the users this might work for a small group of users, with coloured mouse cursors or a similar technique to help differentiating the cursors.

For VR environments the use is actually limited due to the fact that it is only 2D-cursor-based. This creates a big problem when interacting in a 3D space, when requiring multiple DOF. Still this might be a viable solution for special setups, where 2D interaction is sufficient (e.g., selection of objects in a 2.5D scene on a powerwall).

2.2.3 Camera-based Techniques

Some interaction methods rely on the camera of the mobile device. While all of the papers in this section use phones as devices, they also apply to tablets with cameras. They rely of markers that are available for the device to scan as a base.

Prior to the smart phone era, Madhavapeddy et al. [75] created a system where users can use a phone with a camera and bluetooth to interact with a world map application (Fig. 2.1). The application displayed a map of the world, augmented with Spot Codes, a circular bar code. The user(s) can take pictures of the application, containing a Spot Code. The phone will then query the application via the bluetooth connection using the Spot Code's content as an id, to get further information about the special spot on the map.

Thelen et. al. [100] took this idea further by customizing the information transferred to the user's phone. This leverages on the idea of having a separate device for each user. The *3D Human Brain Atlas* (Fig. 2.2) [100] is basically a quiz about the human brain, featuring different levels of difficulty. By scanning a barcode prior to the beginning of the quiz, users select their own difficulty level. The phone remembers this throughout the game and the combination of a code scanned on the main screen together with the difficulty is sent to a server, which in turn sends the quiz question. Also the phone is used as a identification mechanism, where certain phones can be registered as instructor phones and will get the answers together with the question.

Point and Shoot by Ballagas et. al. [15] uses the visual marker approach,

FIGURE 2.1: *Selection on the world map with spot codes. Courtesy of Madhavapeddy et al. [75]*

FIGURE 2.2: *A marker being selected on the* 3D Human Brain Atlas. *Courtesy of Thelen et al. [100]*

but displays them only for short periods of time. The phone will send a notification to the main application to display the markers, then the picture is taken including the markers, after which the markers disappear. The position where the phone was pointed to is then calculated based on the markers.

In the same paper another technique, called *Sweep* [15], was introduced. By optical flow calculations done directly on the phone, it is possible to detect the phones movement and use it similar to an optical mouse, even in mid-air. Unfortunately, due to technical limitations, this method incurred a high latency (about 200ms) making its usage cumbersome. To our knowledge, there is no current study if the technical advances till today could lift this restriction, but it is very likely.

Rhos [93] proposed to use markers on physical objects to create Augmented Reality (AR) games. While this is not directly related, this can still be used with the approaches above, to create additional content and seamlessly add it into the basic content provided by the application to all users.

The following classification will exclude *Sweep*, since it is different from the other methods. It will be classified separately. The methods are basically only used to perform selection tasks (according to the Foley taxonomy). It can be noted that some of the techniques follow up with text input after the selection by using the phone's native keyboard. Combining several selections allows for the other tasks to be completed, too, but this might be cumbersome for the users.

All of these techniques feature input in 2 dimensions (as restricted by the

camera). It is notable, too, that the input device is also used as an output device for most of the methods, making collaboration easier on a larger scale, since interaction is possible without interrupting the workflow of other users.

None of the methods are actually eyes-free, as it is necessary to point with the device at some position on the screen, which is of course only possible by looking at it. This might interrupt the user's workflow, but assuming this is used to present further information on the screen of the mobile device, this is not a problem. Tracking systems are also not needed for any of those methods, all of them are actually designed to do their own kind of tracking their position. Depending on the application, external tracking might be used to improve the accuracy of the chosen method. Transferring content to the smart phone or tablet also makes it a valuable secondary device. A program might provide the standard means of interaction (depending on the program and setup) and additionally can allow for mobile devices to be used for further information, or for additional users to get information. Regarding scalability, all the methods work well with many users, actual restrictions are posed by available screen space where the users can interact. *Point and Shoot* might also have some scalability issues because of the bar codes flashing up on the screen, whenever a user interacts. This might distract other users looking at the screen. The problem increases with more people using the system at the same time.

Depending on the actual setup of a VR environment, these method might work very well (e.g., a powerwall) or might not be a good choice at all (e.g., fully immersive environment).

Sweep can only be used for position tasks, adding a button functionality will also allow for selection and, through composition, all other tasks as defined by Foley. It tracks input in 2D, but it could be augmentable to actually support 3D. It is eyes-free, as it does not use any of the output facilities of smart phones and tablets and is used similar to a mouse. It is also not dependant on external tracking, since doing its own tracking is the core of *Sweep*. The technique in itself is highly scalable. The real problem is similar to trackpads: the number of mouse cursors that need to be displayed on the screen. The cursor is also the problem why this might not be a good choice of input method for VR applications. Similar to the trackpad solutions already described, this is only useful if a 2D pointer can be effectively used with the application.

2.2.4 Menus

Another approach of using smart phones and tablets as input devices is an external menu structure on the mobile device. This can be as simple as presenting all required functionality in the device's native UI and transferring all user input to the actual application. But this most likely causes a break in user interface design between main application and mobile device and is additionally not eyes-free in most cases, interrupting the workflow when used. For this reason several advanced menu versions have been proposed, most of them featuring an eyes-free interaction mode.

Many of those menus base on the idea of Marking Menus [69]. Marking Menus are basically a structure of radial menus. The user can use them eyes-free after learning the menu structure, by just remembering the path to draw (with finger or pen) to a certain menu item. When applied to mobile devices, the device can present the menu to the user who can then either progress to move to the desired menu item blindly, or look at the structure to find the menu item and thus help to remember the path next time. The strokes of the path can be drawn continuously or stroke-after-stroke, depending on the actual implementation.

As current smart phones and tablets allow for multi-touch interaction, standard Marking Menu interaction can be improved by utilizing this. Kin et. al. [63] used multi-touch to enable faster input of the strokes, using two fingers at the same time (Fig. 2.3). They use the stroke-after-stroke version of Marking Menus, allowing to either draw the strokes simultaneously one with each finger, or one after another, alternating between fingers, a bit slower, but offering double the number of menu items, by changing the menu depending on the starting finger.

Flower Menus by Bailly et.al. [13] improve the number of items in a Marking Menu structure by not only allowing straight strokes, but instead take the curvature of a stroke into account. Using the stroke-after-stroke variant of Marking Menus, they have 12 items on each level. Each basic direction (i.e., Up, Down, Left, Right) can be combined with either a straight stroke, or a curved stroke to either the left or right side to get up to 12 items per level. Of course multiple levels of menu structure can be chained one after another.

Basing on the idea of novice and expert mode interaction as used by Marking

FIGURE 2.3: *Two-handed Marking Menus employ multi-touch to draw parts of the Marking Menu strokes using alternating fingers. Courtesy of Kin and Hartmann [63]*

Menus, Francone et.al. [52][51] proposed the *Wavelet Menu* (Fig. 2.5). It is derived from the Wave Menu by Bailly et.al. [14], but better suited to the small screen estate on mobile devices. The first time the user touches the screen, the root level of the menu is revealed around the touch point. Then when the user starts moving the finger towards one menu item, the submenu is appearing from below the current menu, with the current menu expanding outwards. When the user releases the finger at the menu, the next level of menu (or actual functionality) gets selected. If the touch is released prior to this, the last menu stays on top instead. Like with Marking Menus, the user can remember the sequence of strokes needed to access a certain item and can then do the strokes without actually looking at the device.

Gebhardt et. al. [54] use a HTML-based menu displayed on a mobile device to interact with VR spaces. It is targeted for use with configuration tasks of the system, but can also be used for other tasks. They created a custom set of widgets to be presented on the mobile device. An example of this can be seen in 2.6. The strong point of this method is the device-independence, since the

FIGURE 2.4: *The design of* Flower Menus. *Each basic stroke direction (Up, Down, Left, Right) has 3 variations (straight, curved left/right). Courtesy of Bailly and Lecolinet [13]*

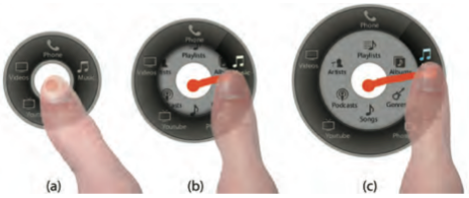


FIGURE 2.5: Wavelet Menus *are a sequence of radial menus, that are conceptually hidden under each other and revealed as the user strokes from the center towards an menu item. Courtesy of Francone et al. [51, 52]*

only requirement for the mobile device used is HTML5-support. Also there are basically no requirements for the server-VR-system; it does not even need to be a VR system.

Menu selection only offers support for selection tasks. By including the Control Menu mechanic, quantify tasks can also be accomplished, and by composition everything else, even text (if paired by a technique like QuikWrite [88]. The input dimension of menus is defined by the number of items it contains, which can be a large number. With the Control Menu mechanic and common multi-touch gestures, like rotation and pinch-to-zoom, four dimensions of input can be achieved, enough to allow for 3D interaction, important for VR applications.

All of the presented menus, with the exception of the HTML-based menu, can be used eyes-free after a training period. They do not need external tracking, but it could be used to provide a means of positioning and/or rotational input to the presented method. This will, of course, negatively affect the scalability of that method. Still any of the menu methods can be used as a secondary input method, especially to avoid menu structures in immersive environments, where navigating menus can be very cumbersome. Without tracking, menu interaction scales very well to the number of users, due to the minor communication overhead per device, the fact that users can practically interact from

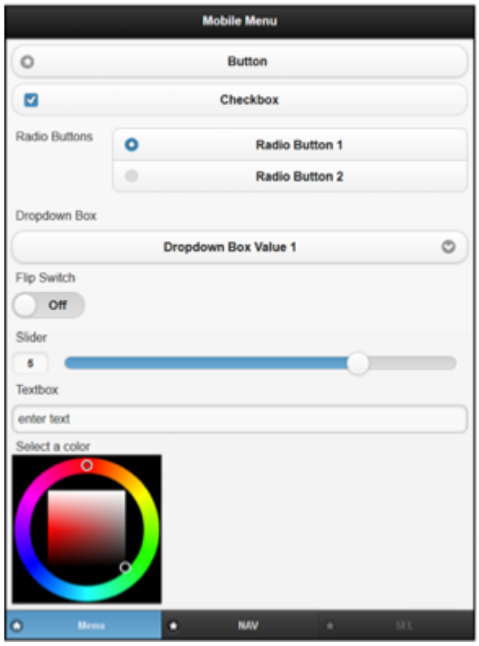


FIGURE 2.6: *A* HMTL5 Menu *composed of several standard and custom widgets. Courtesy of Gebhardt et al. [54]*

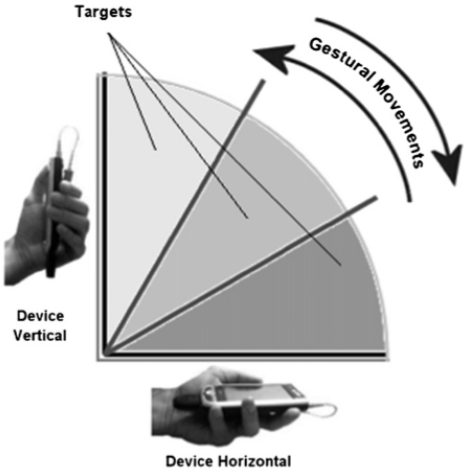


FIGURE 2.7: *The principle of* Motion Marking Menus *is to divide the possible angle range into different parts, each corresponding to a menu item. By alternating the movement or adding small stops in between, a menu structure can be traversed. Courtesy of Oakley and Park [86].*

any position they can see the application screen/area and individual mobile devices do not interfere with each other during interaction.

For these reasons, all presented menu approaches are very well suited to be used in a VR environment, with the actual implementation being dependent on the parameters of the VR application.

2.2.5 Gestures

Using the built-in sensors of smart phones and tablets, especially accelerometers, gestures with the device itself can be recognized. Since these gestures do not use the touchscreen or only for activation of the gesture sensing, it is easy to implement the same gesture recognition for other devices as well, as long as they feature the necessary sensors.

Device gestures can also be added to other input methods, such as the menus presented in the previous section. One has to keep in mind, that the gestures are more difficult to execute with larger and heavier devices, so large

tablets are not a good input device to use the following methods with.

Motion Marking Menus (Fig. 2.7) as presented by Oakley and Park [86], base on the concept of Marking Menus, as shown in the last section. But instead of using swipes for the selection of menu items, selection is done via the angle it is held in. They use a button or a touch on the screen as an activation mechanic. The angle the device can be held is divided into a number of menu items. Two menu items, for example, each get a 45 ° angle, meaning that holding the device horizontally or up to a 45 ° deviation from the horizontal plane will invoke the first menu item, any other posture will activate the second item. By tilting the device up and down from the current posture while still holding the activation button, a menu sequence can be invoked, similar to touch menus.

Building from this general idea, *Jerk-Tilts* (Fig. 2.8) is a method by Baglioni et. al. [12] for selection by gesture. Instead of having the user remembering a special posture or sequence of postures, they use a single tilt-and-back gesture in different directions to invoke functionality. For example, holding the device and executing a quick tilt to the right and then back to the original posture will invoke a selection of a certain kind. The same can be done in all four basic direction and combining two directions allows for a total of eight different possibilities. A big advantage of this method is, that it works without any activation button, allowing to use this together with other input methods using the touchscreen.

Dachselt and Buchholz [39] suggest to use a throw gesture alongside tilt gestures (forming *Throw and Tilt* (Fig. 2.9). Tilt can be used for selections, similar to the methods presented above in this section or to move a mouse cursor, while the throw gesture can be used to transfer data to the main application. Care has to be taken when the throw gesture is executed to have a firm grip on the input device, but it is intuitive to throw content towards the application.

All the techniques above provide means to accomplish selections tasks. Using tilt, *Throw and Tilt* can additionally provide means for position and/or rotation tasks. This can again be composed into all other possible tasks. The number of input dimensions is different for all methods. *Motion Marking Menus* have a dimensionality dependent on the number of menu items.

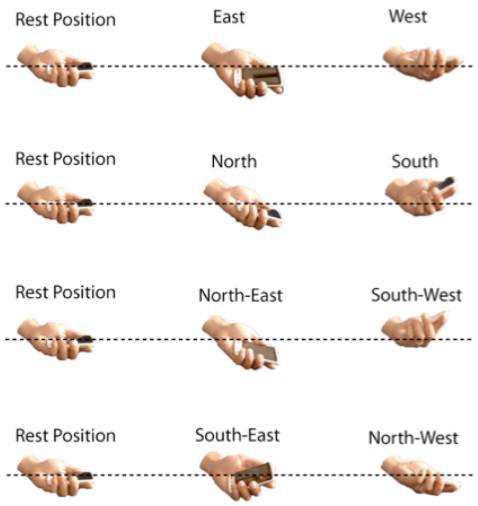


FIGURE 2.8: Jerk-Tilt *are a tilt-and-back kind of gesture, that can be done in eight different directions, allowing for eight different functions to be invoked. Courtesy of Baglioni et al. [12]*

FIGURE 2.9: *The characteristic throw gesture from* Tilt and Throw. *Courtesy of Dachselt and Buchholz [39]*

Jerk-Tilts have a fixed dimensionality of eight, the number of possible gestures. With only three dimensions (two tilt dimensions and throw) *Tilt and Throw* has the least number of dimensions. All methods work eyes-free and need no external tracking and are also scalable to the number of users. Due to the greater motion of the throw gesture, *Tilt and Throw* might scale a little bit less than the other approaches, but this can be somewhat mitigated by providing an alternative to the throw gesture.

All these methods can well be used as a secondary input method to provide a subset of the application's main functionality to additional users. The methods can also be used with tracked input devices, that might already be present in a given VR setup, such as flysticks. This allows for a common input metaphor over multiple input devices.

2.2.6 Application-Tailored Techniques

This section contains several techniques, that are specifically tailored towards a certain application. While this limits their general applicability, it is still possible to use them in similar usage scenarios. Because these techniques do

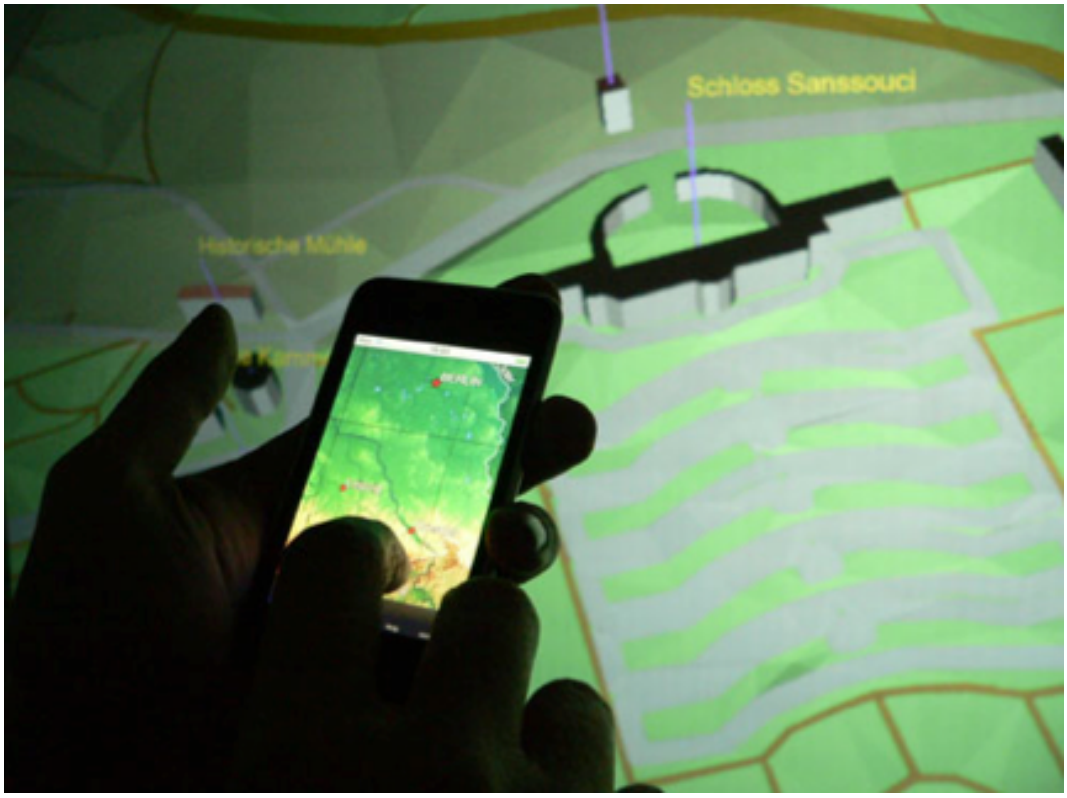


FIGURE 2.10: ModControl *can be used to present an overview of the content to a mobile device and allows independent interaction. Courtesy of Deller and Ebert [40]*

not have much in common, their classification will be written individually instead of grouping them together as in the previous sections.

The *Mod Control* (Fig. 2.10) system by Deller and Ebert [40] was designed for interaction with a map application. The system itself is modular and not directly tailored towards the application. But the reference implementation features several interaction modules to improve control of the map application through a smart phone. Among them is a combined touch/accelerometer interface for 3D navigation. The user can modify the current view as flying, controlling the forward and sideward speed using the touchpad while the devices rotation control yaw and heading of the view. This is a common problem, especially in VR environments. This module can be classified to accomplish position and orientation tasks, in 5 DOF, it can be used eyes-free and does not need external tracking, it is usable as a secondary device to other input devices which do have less DOF. Also it scales well to the number of users.

Another interesting module is the image module, featuring a small version

of the map shown by the main application. This allows users to explore the map independently from other users and send their current view back to the application if needed. It fulfils a position and select task, in two dimensions. Though it cannot be operated eyes-free, it scales very well since every user can interact totally independent of the others. For the same reason it also makes for a good secondary device for additional users. The applicability in the VR domain is dependent on the VR application. For terrain simulations, for example, it will work very well, while it is not useful for architectural applications.

Seewonauth et.al. [95] propose two techniques, *Touch & Connect* and *Touch & Select* to initiate data transfer between a phone and a computer. Since today's smart phones and tablets have quite large memory capacity, they can be used as a mobile data storage. To avoid complicated file management, one of those two method can be used to easily copy a data set. *Touch & Connect* bases on NFC. The users select a file on the source device and simply touch a NFC tag on the target computer to copy the file. Note that the same approach can be used to get the current data set from the display or to initiate a connection (as input device) to the currently running application, avoiding the need to enter IP-Adresses or Host names.

Touch & Select uses several NFC tags to track the position of the mobile device. The user again selects a file on the smart phone or tablets and touches a point directly on the screen. This initiates the file transfer. The same principle can be used to interact with the screen directly for other effects.

Both methods allow to accomplish selection tasks, with *Touch & Select* having additional two dimensions of input. Since the user has to point with the device, it cannot be used eyes-free, but especially if used for data transfer, both methods are valid secondary input means. It does not scale as well as other methods to the number of users, since it is also dependent on the number of NFC tags provided and also it requires the users to stand near the tagged spots, limiting the concurrent use. External Tracking is only needed by the means of providing the NFC tags.

For VR applications, *Touch & Connect* is probably more useful than *Touch & Select*, since the latter selects a 2D screen position, which is only situational useful. But to initiate connections between a mobile device and a stationary

computer system, *Touch & Connect* is very convenient. Please also note, that a similar behaviour is achievable by providing a visual tag, that can be scanned by a smart phone or tablet camera if NFC is unavailable.

Semantic Snarfing describes a technique by Myers et.al. [82] using a laser pointer to mark a spot on a large display. The semantic area (e.g., a dialog) this spot belongs to is then transferred to a mobile device's screen. Due to technical progress since the time the paper was published, the method can be used today without a laser pointer by utilizing the camera of a smart phone or tablet. This makes *Semantic Snarfing* a method similar to the camera methods described in their own section. What makes *Semantic Snarfing* special from the camera techniques is the semantics that needs to be provided in order to make the system work. The method has in general the same properties as the camera methods, as it allows selection (and then more tasks on the mobile device's screen), scales generally well to the amount of users, but does not support eyes-free interaction. Tracking is not required. Its usability in VR environments is dependent on the application and setup, quite similar to the camera methods. But as a secondary device, *Semantic Snarfing* is applicable to even more VR environments, as the primary device can be used to select the "snarfing point" for the mobile device.

2.2.7 Conclusion

Several interaction techniques are possible with smart phones and tablets. For applications where collaboration is expected, supporting interaction via mobile devices will add benefit to users and their collaboration. Even if not the full set of functionality can be accessed by smart phone or tablet interaction, it will still help users to be active users of an application instead of being only passive spectators.

However, it is important to choose the right form of interaction for the application in question in order to reach a high usability of the resulting system. The overview given by this paper should help in considering which methods are feasible for certain settings. Table 2.1 contains an overview of the properties reviewed in the previous chapter for easier comparison.Moreover, it can form a starting point for implementation of an own interaction metaphor. If possible, a metaphor for the mobile device should not deviate from the metaphor of

TABLE 2.1: *Overview of all presented methods. (★: methods using output capabilities.; c: achievable through composition; parenthesis depict simple improvements to the method.)*

Method	Position	Orient	Select	Ink	Quantify	Text	Dimensions	Eyes-free	Tracking	Secondary	Scalability
Trackpad - Methods											
ArcPad	✓	(c)	(✓)	(c)	(c)	(✓)	2(4)	✓	✗	✗	-
Camera - Methods											
World Map			✓			✓	2★	✗	✗	✓	o
3D Human Brain Atlas			✓			✓	2★	✗	✗	✓	o
Point and Shoot			✓			✓	2★	✗	✗	✓	o
Sweep	✓		(✓)				2 (3)	✓	✗		-
AR			✓			✓	2★	✗	✗	✓	o
Menu - Methods											
Marking Menus			✓				any	✓	✗	✓	+
Multi-touch Marking Menus			✓				any	✓	✗	✓	+
Flower Menus			✓				any	✓	✗	✓	+
Wavelet Menus			✓				any	✓	✗	✓	+
HTML Menus	(c)	(c)	✓	(c)	✓	✓	any	✓	✗	✓	+
Gesture - Methods											
Motion Marking Menus			✓				any	✓	✓	✓	+
Jerk Tilts			✓				any	✓	✗	✓	+
Throw and Tilt	✓	✓	✓				any	✓	✗	✓	+
Application-Tailored											
Mod Control - Navigation	✓	✓					5	✓	✗	✗	+
Mod Control - Image	✓		✓				2	✗	✗	✗	+
Touch & Connect			✓				-	✗	✓	✓	o
Touch & Select			✓				2	✗	✓	✓	o
Semantic Snarfing	(✓)	(✓)	(✓)	✓		✓	2★	✗	✗	✓	+

the main device to avoid the necessity of learning to use two devices. In some application areas it is possible to avoid the usage of other devices completely, and using smart phones or tablets as the only input device.

Chapter 3

Mobile Interaction: Specialized Approaches - Proof of Concept

As a proof-of-concept work, first results were achieved creating specialised approaches for interaction. Generally, specialised approaches can be developed quicker, as there is only a single use-case to take into account. For the same reason, it is easier and faster to evaluate the performance of these methods. We developed the following techniques to show the general viability of large display interaction through smart phones and tablets.

3.1 Evaluation of Special Use Cases

The actual programming and first evaluation was not done in the scope of this dissertation, but as diploma thesis. Therefore this chapter will not focus on the implementation, but on the results of the extended evaluation.

3.1.1 Motivation

A user study was conducted with four different case studies on a 3x3 tiled high-resolution display. 17 test candidates of various ages and different levels of user experience were asked to complete all test scenarios using a smart phone (HTC Touch Diamond 2) and also using the traditional keyboard or mouse or a combination of both.

Description of Test Cases

A short summary of the test scenarios follows. For a more detailed description, see [23].

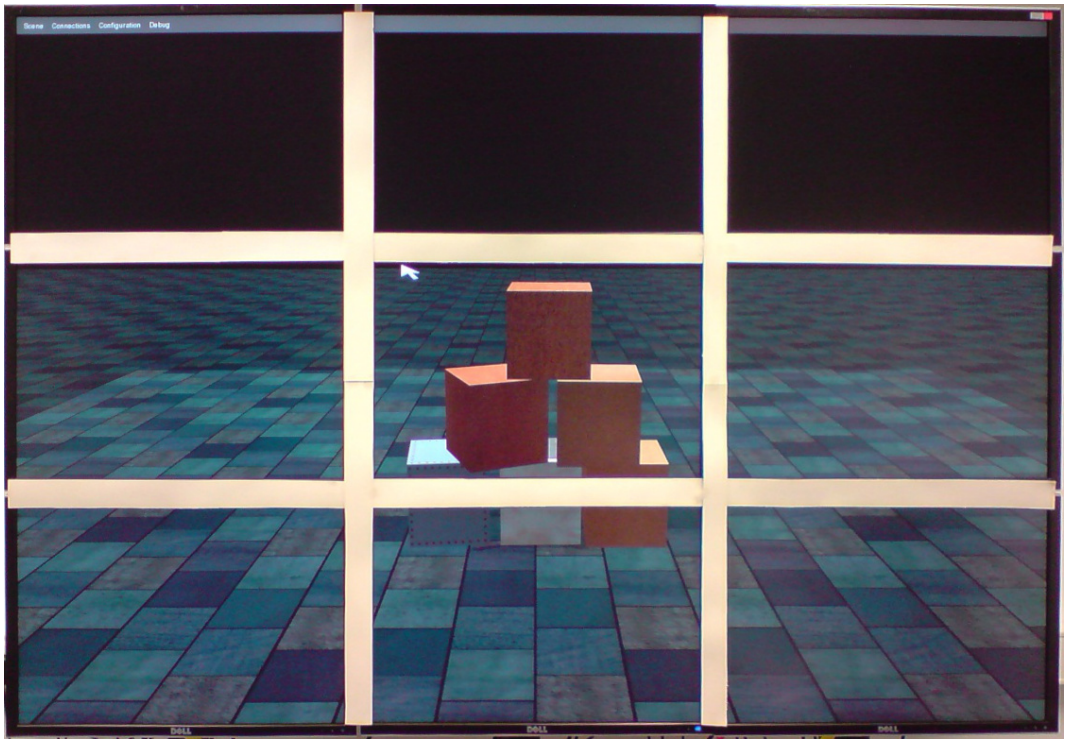


FIGURE 3.1: *Stacking Cubes, the user has to stack up three cubes in a physics-enabled environment, moving one cube at a time*

Stacking Cubes In this three-dimensional test scenario users were asked to stack three cubes one on another in a simple physics driven environment. This involves 3 degrees-of-freedom (DOF) movement in space, without regard to rotation. This could be accomplished by either using a keyboard (using two keys per DOF), a mouse (normal position tracking + mouse wheel) or a smart phone (using the touchscreen to perform movement in two dimensions and tilting the phone itself to perform movement in another two dimensions, resulting in 2 ways to control x-direction) as input device. It should be noted, that the phone is the only device providing continuous movement in all dimensions. This scenario uses the subtasks of *Position* and *Select*. Figure 3.1 shows how the scene looks like in the study.

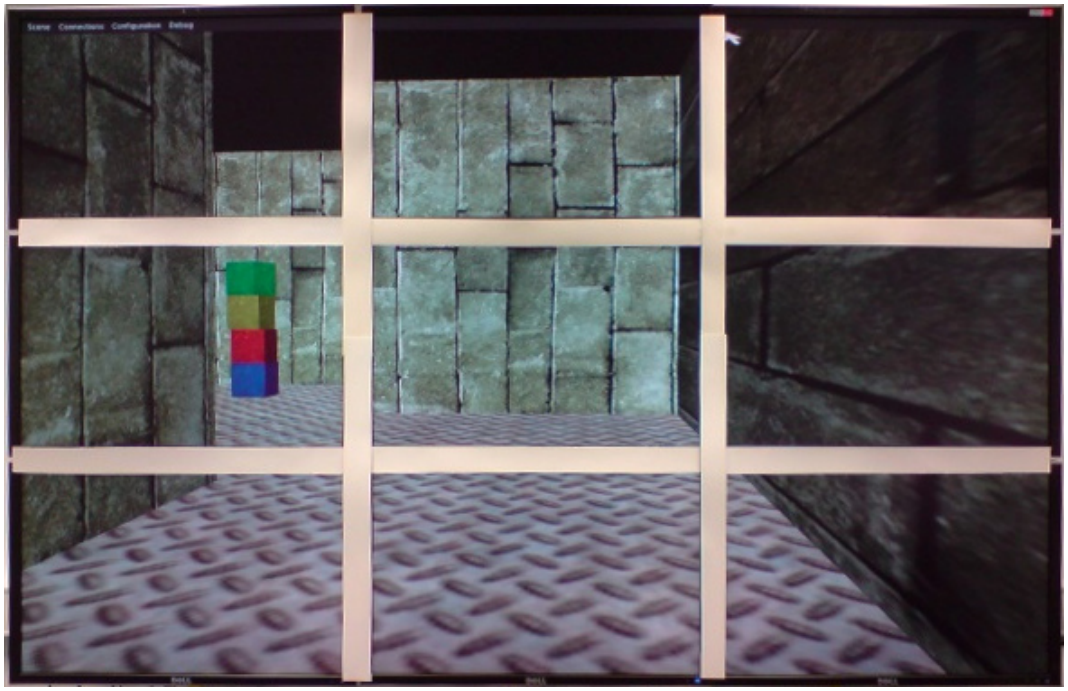


FIGURE 3.2: *Maze, colored cubes can be found at one spot in the maze. Users need to navigate through the maze to find them*

Maze The second scenario features the first-person view of a simple maze (as can be seen in Figure 3.2). The test candidates have to navigate through the maze (with the help of a map) to a certain spot where they can pick up a colored cube by just touching it. Afterwards, the goal is to backtrack the way and to drop the cube into a bin outside the maze. The most commonly used input method for this case is probably the combination of keyboard and mouse, known to a wide audience of first-person computer games. Furthermore, control is possible using only keyboard (arrow-keys) or only the mouse (mouse position orients the view and holding the mouse button accelerates). Using the smart phone as it was a joystick (tilting the phone in the desired direction) was the last input method implemented. This scenario consists of the subtasks of *Position* and *Orient*.

City Map Annotation On a local city map (Figure 3.3), the candidates are able to place flag markers. These markers contain some more information: A descriptive text and a picture. The flags can be placed by selecting the flag in the upper left corner in the map view and dragging it to the desired position.

FIGURE 3.3: *City Map Annotation, the small flags from the upper left corner can be dragged-and-dropped onto the map*

Then a small popup-window will appear (the size can be seen in Figure 3 in the lower middle screen), where the user can enter the description and select a picture to be displayed. The smart phone displays a smaller version of the same map, that can be panned and zoomed individually without affecting the main view on the large display (Figure 3.4). Unfortunately, the smart phone did not support multi-touch interaction, so zooming had to be done by using a menu. With the same menu a selection mode can be activated. Then the next tap on the map will place a flag marker and open a new view on the phone where the user can enter description text and select or even take a photo. As none of these actions will directly affect the main view (besides adding the marker on the map at some point), multiple users can perform this task at once without interfering with each other). In this scenario *Position*, *Select* and *Text* was used.

Jigsaw-Puzzle One of three simple 5x5 tile jigsaw-puzzle had to be solved in this test case. The puzzle was shown on the tiled wall (Figure 3.6) and simultaneously a live copy of it on the smart phone (Figure 3.5). On the phone

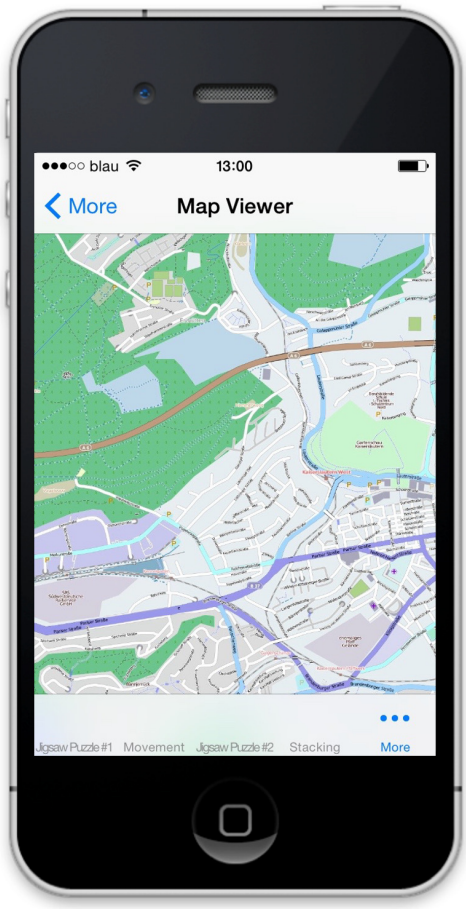


FIGURE 3.4: *Screenshot of the City Map on the Phone*

FIGURE 3.5: *Screenshot of the Jigsaw-Puzzle on the Phone*

the tiles can be dragged around with a simple touch and drag. If necessary the user also can zoom in or out of the puzzle. Additionally, each test candidate had to solve another puzzle (for a total of three) with the keyboard and the mouse. *Postion* and *Select* were the subtasks needed for the jigsaw-puzzle.

Evaluation Setup

The evaluation was done with the already mentioned 17 participants of various ages and levels of computer experience. They were asked to carry out the tasks described above in random order using all available input methods, again in random order. In each case the candidate had a short opportunity to get used to the input method, to a point where he or she was able to perform the task. Unfortunately not enough time was available for the candidates get a higher

FIGURE 3.6: *Jigsaw-Puzzle, the tiles have to be brought in the correct positions by dragging*

level of proficiency. This probably had a negative effect on the score of the smart phone especially, as no user has used a phone for large display interaction before, but each one had at least a bit of training using a computer mouse and a keyboard. For a quantitative analysis the time to complete each task with each input method was measured. To get comparable times of all candidates, the times were normalized by dividing each time by the accumulated time of all input methods of the observed task. This yields times on a scale from 0 to 1. Another quantitative measure was a grade given by every participant for each input method per test case. Possible grades range from 1 (best) to 6 (worst). To get some qualitative results, each candidate was also asked for his/her comments on the smart phone control and also for improvement proposals. The rest of the paper will now describe the results formally and draw conclusions.

3.1.2 Formal Evaluation

The measured times (total and normalized) are on a ratio scale, grades are ordinal. A set of popular descriptive statistics about the normalized times can be found in Table 1. Using a one-way analysis of variance (ANOVA) for each test scenario, the mean times can be shown to be statistically significant different on a confidence level of 5%. The basic requirements of the ANOVA, the mean times being normal distributed and all mean times having the same variance, are assumed to be met. For timed tasks normal distribution can safely be assumed and to be sure about the variances a Levene-Test has been performed for each test scenario. Unfortunately the Levene-Test did not confirm (on a 1% level) that the variances in scene 2 are equal, but as the ANOVA is known to be a very robust test, it was done anyway, but this fact has to be kept in mind. The ANOVA itself showed that the mean-time differences in the input methods are statistically significant on a 5% level (with F-values of 7.109, 21.733, 43.898 and 67.096 for scenes one to four, respectively). For scene 2, the F-value can show significant differences in means for confidence intervals below even 0.1%. This fact and a Welch-Test (also testing for differences in mean values, but also valid for non-equal variances) also showing differences in the mean normalized times leads us to the conclusion, that the ANOVA yield correct results even for scene 2. To get deeper insight into which mean times actually differ, a Tukey-HSD test was conducted. This test shows the pair-wise (in-)difference between the normalized mean times. The results are shown in tables 2-4.

TABLE 3.1: *Descriptive Statistics of Normalized Times. 1st Q and 3rd Q stand for 1st and 3rd quartile, CV is the coefficient of variation and IQR is the interquartile range*

		Min	1st Q	Median	3rd Q	Max	Mean	Std Dev	CV	Range	IQR	Curtosis	Skewnes
Scene 1	Keyboard	0.104	0.172	0.261	0.307	0.570	0.265	0.125	0.471	0.467	0.135	0.346	0.797
	Mouse	0.079	0.190	0.316	0.383	0.576	0.301	0.141	0.469	0.498	0.192	-0.840	0.327
	Phone	0.230	0.348	0.401	0.512	0.740	0.435	0.136	0.314	0.510	0.164	-0.170	0.548
Scene 2	Keyboard	0.145	0.169	0.208	0.228	0.264	0.204	0.035	0.174	0.120	0.059	-1.087	-0.038
	KB + M	0.119	0.177	0.196	0.216	0.328	0.204	0.051	0.248	0.209	0.039	1.591	0.951
	Mouse	0.157	0.226	0.245	0.275	0.372	0.250	0.048	0.194	0.215	0.049	1.531	0.478
	Phone	0.211	0.281	0.335	0.412	0.483	0.343	0.076	0.221	0.271	0.131	-1.112	0.124
Scn 3	Keyboard	0.235	0.295	0.385	0.476	0.543	0.380	0.102	0.269	0.308	0.181	-1.423	-0.062
	Phone	0.457	0.524	0.615	0.705	0.765	0.620	0.102	0.165	0.308	0.181	-1.423	0.062
Scene 4	Keyboard	0.563	0.650	0.697	0.761	0.829	0.701	0.073	0.104	0.267	0.111	-0.654	-0.178
	Mouse	0.171	0.239	0.303	0.350	0.437	0.299	0.073	0.245	0.267	0.111	-0.654	0.178
	Phone	0.263	0.405	0.476	0.558	0.771	0.483	0.128	0.265	0.508	0.153	0.295	0.253

TABLE 3.2: *Descriptive Statistics of Grades. CV is the coefficient of variation*

		Mode	Median	Mean	Std Dev	CV
Scene 1	Keyboard	2	2	2.1	0.96	0.45
	Mouse	3	3	2.5	1.19	0.48
	Phone	4	3	3.3	1.07	0.33
Scene 2	Keyboard	2	2	2.35	0.836	0.36
	KB + M	2	2	2.12	0.963	0.45
	Mouse	3	3	2.88	1.1315	0.3
	Phone	3	3	3.19	0.9656	0.3
Sc. 3	Keyboard	1	1	1.4	0.48	0.35
	Phone	2	2	2.3	0.89	0.39
Scene 4	Keyboard	4	4	4	1.6202	0.41
	Mouse	1	1.5	1.69	0.8455	0.5
	Phone	2	2	2.5	1	0.4

TABLE 3.3: *Tukey-HSD Results for scenario 1*

	KB	CM	SP
KB			✓
CM			✓
SP	✓	✓	

TABLE 3.4: *Tukey-HSD Results for scenario 2*

	KB	KM	CM	SP
KB				✓
KM				✓
CM				✓
SP	✓	✓	✓	

TABLE 3.5: *Tukey-HSD Results for scenario 4*

	KB	CM	SP
KB		✓	✓
CM	✓		✓
SP	✓	✓	

KB = Keyboard, CM = Computer Mouse, KM = Keyboard + Mouse SP = Smart Phone; A checkmark denotes a significant differece in mean times.

The most important fact from this results is, that the mean times of the smart phone users always differ significantly from all others. Knowing this, we can safely interpret the normalized times to get an overview of the test results. Since the grades are on an ordinal scale, no ANOVA was conducted for them.

A (Pearson) correlation test shows significant (again on a 5% level) correlation in the different times taken to solve each scenario using the smart phone. An exception for this is scene 3, the *Map Annotation*. A possible explanation for this may be the fact, that in this test case the smart phone's built-in menu had to be used a lot. This was difficult for most users, as nobody had any experience with a Windows 6 smart phone. The correlation for the other test cases show, that there is some kind of taste or distaste for the phone control. This fact is hardly surprising, but still notable. It also hints, that the usage of the native menu of the phone is a very unintuitive way of providing interaction possibilities. This was also mentioned by a few of the test candidates.

The ARC-Pad cursor control method [76] was also implemented. This control turns the phone into a trackpad, where a tap causes the mouse cursor to jump to the position on the screen relative to the position the tap happened on the phone, e.g., A one finger tap in the center of the screen lets the mouse cursor jump to the center of the screen. A swipe on the touchscreen moves the mouse cursor normally. The goal of this control is to have fast cursor positioning together with the accuracy of a touchpad control. While the idea sounded very feasible, early tests showed very bad results. Therefore, a formal evaluation of this interaction pattern was not conducted. The main cause of the ARC-Pads issues was the inaccuracy when selecting a cursor position by tap. The resulting corrections of the position took too much time to be comfortable for the users.

The test candidates were also asked for an informal description of their experience using the new input mechanisms. Many of them stated that they had little to no experience with the control of the 3D scenarios. They also pointed out, that the control was a big lagging (Probably caused by the slow CPU on the smart phone). Virtually all users liked the 2D scenarios, where direct interaction was enabled. This was a very intuitive way of solving the tasks at hand. The issues identified here were almost all about the absence of multi-touch and the need to use the clunky system menu to activate selection

mode in the Map Annotation Scenario.

3.1.3 Conclusion

The smart phone didn't get the best grades or the best times. What still makes it a viable input option is the fact, that it solves the problem of scaling the interaction against the number of users. Using the improvement suggestions made by the test candidates will further improve the interaction metaphors used in this first study. Together with newer and more capable hardware the smart phone seems to get on par with the other input methods. Unfortunately, there is no formal study at this time to show this, as this is still work in progress. Informal evaluation including the comments made by the test candidates shows, that all candidates liked the interaction metaphors and were also able to understand them. The main complaint was about insufficient hardware capabilities, especially sketchy accelerometers and missing multi-touch capabilities. This is something to include in further approaches. Informal studies made using the most recent Apple iPhone show much better results. Unfortunately at the time of writing no formal evaluation is was available to be included here. The better accelerometers, combined with filtering of the acceleration sample data provided by the sensors yield very stable results and greatly improve user experience. If this translates into faster solutions of the given tasks is to show in a formal study similar to the one presented here. For the 3D scenarios, *Stacking Cubes* and *Maze* users were mainly burdened with the special hand posture needed to activate the accelerometer on the HTC Touch Diamond 2. Using the touchscreen to do the activation causes better results. In *Map Annotation* and *Jigsaw-Puzzle* using the touchscreen of the smart phone for direct touch interaction was preferred by most users. What made the tasks in both cases a bit more problematic, was the inclusion of the system menu, as already stated in the last subsection. For those 2D tasks the multi-touch capabilities of modern smart phones will greatly improve performance of these tasks. Usage of multiple finger to move multiple jigsaw-puzzle tiles and using pinching gestures for map navigation allows for a more intuitive interface and greater user satisfaction.

When grouping the scenarios using the Foley-Taxonomy, *Select* was a task that can be done with the smart phone most easily. The smart phone provides

direct touch interaction for selection tasks, while the mouse only provides indirect methods. Using the keyboard either means finding the correct key for the regarded object or instead cycling through all available objects until the object to be selected appears. *Orient* was not performed so well with the smart phone. The main cause may be the use of the accelerometers for this. Maybe better accelerometers, more experience with this kind of interaction or a new metaphor will solve this, but further research is needed. For *Position* the results seem to be mixed at first. But when looking at the different techniques used, one can see, that position with the accelerometer did not perform well, for the reasons already stated. *Position* with the touchscreen worked very well and got a high level of user satisfaction.

Future improvements are, as already described, to use better hardware. We already tested the setup with iPads and newer iPhones, but no formal tests were done. As a general note, it is very advisable to use a communication protocol with a low memory footprint with smart phones. This can help to reduce lag in the connection between large display and phone. While this increases development time, lag decreases the user experience by a large amount. When using the accelerometers, it is also recommended to allow user configurable settings for home positions, dead zones and sensitivity. As seen in the Map Annotation test scenario, menus should be avoided if possible, as tends to break the intuitively of the interface. If really needed menus should only contain the most necessary items and be large enough to be selected with ease.

From the view of an interaction designer, the results show that touchscreen input (and output) can be very well used for large display interaction. Even older phones have a touchscreen good enough for this task. Employing multitouch gestures enhances the user experience, but is not needed for basic functionalities. Of course it is necessary to keep an eye on the smaller screen real estate on the phone, but especially for 2D tasks this is the preferred way to go. Using the accelerometer for 3D interaction is a good approach per se, but in reality requires some practice on the user's side. This is therefore only feasible if the same interaction pattern can be reused many times.

3.2 Transparent Touch Surface

Another approach, not specialized towards an application but a certain hardware configuration, is to use a tablet as a magic lens. Since (6DOF) tracking of the tablet is mandatory for this, only environments providing tracking facilities can employ this technique. In most cases this limits application of the Transparent Touch Surface to VR, especially CAVEs.

3.2.1 Related Work

The Magic Lens[29, 97] was developed as a filtering tool. It originated as a movable frame in computer programs. Inside the frame, the underlying view is altered some way, e.g., zoom (hence the term Lense), Markup, etc. It became a standard tool in information visualization and therefore found its way into mobile interaction as well.

The camera-based methods, presented in section 2.2.3 in chapter 2 are basically variations of the Magic Lens approach. As described there, using mobile devices as Magic Lenses allows to display additional information on the device itself, even when the device is moved away from the actual spot of interest. This allows for a better multi-user-scalability. In those approaches the Magic Lens is merely a starting point for interaction directly done on the mobile device.

3.2.2 Idea

In contrast to the traditional Magic Lens approaches, we do not aim on providing additional information. Instead the Lens is used to allow touch interaction. Flysticks are the main way of interaction in most CAVEs. They provide 5 to 6 degrees of freedom (depending on actual implementation) and several buttons to interact. The metaphor is a 3D-point-and-click interface, basically the idea of a computer mouse translated into the world of Virtual Reality. Usually touch interaction is more natural to the user. Unfortunately, it is still not easy to track the individual fingers of users without the use of expensive or encumbering hardware (such as a VR glove).

Our low-cost solution employs tablets (or even smart phones) as a kind of Magic Lens on which users can interact through touch directly with the part of

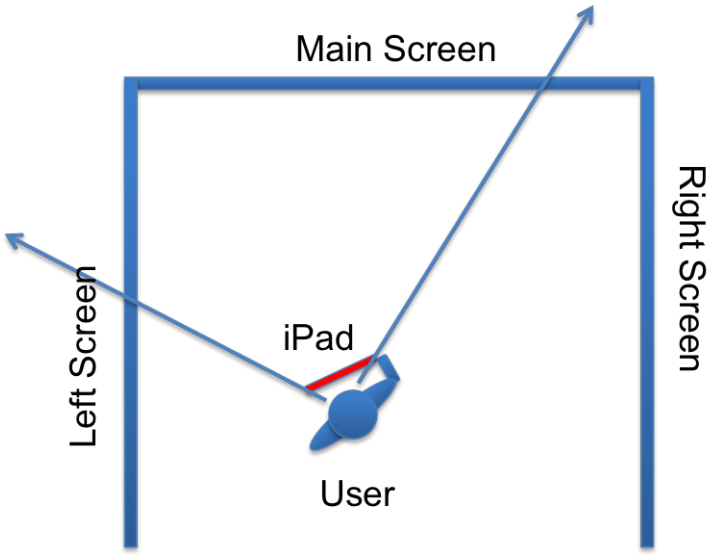


FIGURE 3.7: *Field of View of the Transparent Touch Surface*

the scene they see. We use the normal tracking capabilities of a CAVE to trace the movement of the tablet all time. By knowing the position and orientation of the device, we can therefore render the scene from that point of view and transfer that to the device. When done with the correct aspect parameters, the rendered image on the mobile device will look exactly the same as the scene would look for the user without having a tablet held in front. These correct rendering parameters can be calculated from the position of the device and the position of the viewer (which is also usually tracked by CAVE-Systems, see 3.7) in the same way as it is calculated for the other screens (i.e., the CAVE's walls).

As the user touches the screen of the tablet, the touch points are converted to rays. This is done by converting the touch point and its direction (i.e., the device normal) from device space to scene space. These rays can then be used for interaction as done with a flystick. The multitouch capabilites of the tablet can either be used to enable gestures for interaction, or to touch the virtual object with several fingers at once, basically granting the user an input device for each of their fingers.

3.2.3 Implementation

The approach was implemented for the Vrui[65] framework using the built-in support for streaming. For any given VR setup, an extra screen is added. This screen is then bound to the tracked position (and orientation) of the tablet. That makes Vrui update the screen position automatically whenever the tracking-system updates. The rendering output of that screen is then sent to the tablet via a WiFi connection as a series of JPEG images, thus creating a MJPEG video stream. This stream is presented on the tablet via a small app, that simply receives the frames and displays them as they are sent. Any touch input received by the app is directly sent back to the main application. This keeps the calculations done on the tablet to a minimum, in order to keep latency low. Touch updates received by the main application are mapped to one input device per finger, so that each finger controls its own device.

We used an iPad-2 as tablet for the prototype setup and run in a 4-sided CAVE (3 walls + floor) with active stereo displays.

3.2.4 Limitations

The Transparent Touch Surface is an intuitive to use input method, allowing to interact with multiple fingers. Unfortunately, employing (normal) tablets in immersive environments causes a break in the immersion. Having a 2D image in the 3D immersive world is clearly visible to the user. Even if the tablet is capable of providing 3D images, immersion is still broken when the user touches the surface of the tablet.

Also the unnatural posture the device needs to be held while interacting puts a lot of strain on the arm(s) of the user.

Nevertheless, this approach is very easy to use and can allow difficile interaction at a low cost.

3.2.5 Future Work

At the moment, this method is only usable on iOS devices, mainly due to the programming done for a special app in Objective-C. We are currently working on an implementation using web technologies to avoid the need of an app on the user's device. Instead every device supporting an HTML5-browser

(which is virtually every device in use today) can be used. Another possibility of improvement is to support better video compression than MJPEG streams. This is entirely dependent on the hardware of the mobile device and the CAVE, since better compression usually needs more computing power.

Chapter 4

Mobile Interaction: Solving Common Issues

The ideas presented in this chapter solve common issues in interaction with large displays, after the preceding chapter only contained specialized techniques. The first approach uses the sensors in modern smart phones and tablets to enable low-cost self-tracking on those devices. Common usability issues with large displays, such as the distal-access-problem, are the topic of the second part of this chapter.

4.1 Self-Localization

Knowing the physical location and orientation of a mobile device can be advantageous for many interaction methods. As shown in the previous chapter, device orientation and position can be used as (part of) an interaction method. Other ideas, as for example presented in chapter 2, also make use of the device position and orientation.

Especially in VR environments it is common to have some form of tracking technology, to follow the movements of user(s) and/or input device(s). Typical CAVEs, for example, track the position of the viewer in order to calculate the projection of the presented scene and also track a flystick, that is used as primary means of interaction. Depending on technology and precision these devices can get very expensive and are usually only used when necessary. That is the reason for why these trackers are rare.

4.1.1 Idea

Smart phones and tablets contain accelerometers, compasses, and usually two cameras. Compasses and Accelerometers can be used to calculate the orientation of the device in three-dimensions. Using a low-pass filter on the output data of these sensors, jittering and acceleration from actual device movement can be filtered out quite effectively.

Unfortunately the reverse is not true. While a high-pass filter will give signals about the device acceleration, those signals are not accurate enough to gain insight about device position, since position is calculated by integrating acceleration over time twice. This double integration of the measuring error increases the error to a point where the result is unusable after a mere second.

The camera available on modern smart phones and tablets can provide pictures of their surroundings. By using the stream of pictures given by the camera, relative movement can be calculated. Especially in bad lighting conditions and faster movement speeds, consumer level smart device cameras tend to produce blurry pictures. This can render the output useless in the course of the movement. Since that makes relative positioning impossible, creating a map of image features (such as SIFT [73], SURF [27], etc) that can be rediscovered on later image frames. This is called Simultaneous Localization And Mapping (SLAM) [101] in robotics. This method is unfortunately not usable on current consumer-level smart phones and tablets, due to their low processing power.

Detection of rotation is also inaccurate using the camera. Since the camera only produces a two-dimensional projection of the real world, translation in the image-z-direction can not be tracked. This hinders exact pose estimation.

Combining the data we can extract from the camera together with the accelerometer and compass, we can improve the method. Using visual markers with known positions, the camera can be used for localization as long as it can see a marker. Accelerometer and Compass can be used for pose estimation at any point. When the device is moved at a higher speed, the only way to localize is to use the integrated device acceleration. This is (as mentioned above) inaccurate. But position can be localized accurately again as soon as the camera can provide a stable image of a marker, which should be the case when the movement of the device stops.

This approach allows us to track position and orientation of a mobile device with an accuracy of centimeters only with the help of visual markers. An obvious application is to use a commercial-grade smart phone as a replacement for a flystick in a Large-Display environment, allowing for intuitive pointing gestures.

4.1.2 Related Work

Using GPS might seem as a natural approach for localization. Unfortunately, GPS is designed to work outdoors on larger scales. Even with recent improvement to GPS receivers and their signal processing, deviations are still in the range of 10 cm or more. An additional limitation is the need of receiving data from at least 4 satellites. Indoors this is often not available and therefore too much of a limitation.

Other technology related to tracking include Near-Field Communication (NFC, e.g., RF-ID tags), the Apple iBeacon, or the WiFi-positioning system (WPS). Those system are based on wireless network technologies and basically try to find the position by finding nearby network senders and comparing that to a database of known locations of senders. If applicable the signal strength can be used to refine the result. WPS is designed for the large-scale application, comparable to GPS, and thus not usable for precise self-localization. NFC and iBeacon is targeted at the communication itself and can therefore only be employed to locate markers.

Similar approaches to marker finding were described earlier in chapter 2, in the subsection 2.2.3 Camera-Based Methods. In essence, all these methods work by having the user take a photo of one or more markers and base interaction on that marker. This is also a low-precision version of tracking.

Also, most Augmented-Reality (AR) applications also rely on visual markers to track the position of the device used. This is usually done with high precision of millimetres. The tradeoff is, however, the low speed allowed before the camera picture becomes blurred to a point where the marker is undetectable.

4.1.3 Implementation

An iOS app was written using OpenCV, an open-source framework for computer vision. The app continuously queries the camera of the device and tracks for markers in the image. Since a limitation of iOS is to only have one camera available at a time, there is an option of changing the cameras after a set amount of time. This way both cameras can be employed. We use 10 markers with BCH codes. For simple tracking applications, putting 4 markers on the floor and 2 on both walls is sufficient to have at least one marker visible to the camera most of the time. More markers are needed if users are expected to do more complicated gestures with the phone, than pointing at areas on the screen.

Output from the marker detection and pose estimation is fused with the results from the device's accelerometers and compass as explained above. The resulting position and orientation is sent to a host application via UDP at a minimum rate of 10 updates per second. For testing purposes, it is possible to erase and register markers on the fly.

4.1.4 Result

We can conclude, that the method works fine with limitations. Fast movement of the device, especially in bad lighting conditions make position data very inaccurate for a short amount of time. It needs to be tested, if interaction should be halted during this time. Maybe accuracy is still high enough to allow to track gestures "drawn" by the device in the air.

For pointing at a large display, in the way flysticks are used, the method is viable. Better sensor fusion algorithms (e.g., using a Kalman-Filter) can be used to further improve accuracy. Given it is a low-cost solution, even this first prototype allows for enjoyable interaction.

4.2 High Resolution Pointer Control

As stated in chapter 2, there are some common issues when it comes to the usability of large displays in general as identified by Robertson et. al. [91]. The method presented in this section aims to tackle the following two issues: *Losing*

the pointer and *Distal information access problems*. Especially when the resolution of large displays become high, pointers and cursors tend to become small in comparison to the whole screen estate. Additionally moving the pointer over screen, from one corner to the other takes more time. Speeding up the pointer solves that issue, but introduces the new problem of overshooting the target or having issues in precise pointer positioning. Common solution to this problem are based on screen distortion, such as the drag-and-pop technique suggested by Baudisch et al. [19]. When the user initiates a drag-and-drop operation, possible drop targets bend towards the dragged icon. While this reduces the pointer movement needed, it distorts the current view. Also the new position of the target icon has to be recognized after the dragging begins, introducing a slowdown to the operation. Our solution does not rely on changing the view, but instead allows to change the speed of the mouse pointer as needed.

4.2.1 Target Aquisition Theory

This tradeoff between speed vs. accuracy can be expressed through Fitts' Law [49]. The original paper states equations 4.1 and 4.2:

$$I_D = -\log_2\left(\frac{W_S}{2A}\right) = \log_2\left(\frac{2A}{W_S}\right) \tag{4.1}$$

$$I_P = -\frac{1}{t}\log_2\left(\frac{W_S}{2A}\right) = \frac{I_D}{t} \tag{4.2}$$

with:

I_D Binary Index of Difficulty

A Average Amplitude in Movements (Distance to the center of the target)

W_S Tolerance Range (Width of the target area)

I_P Index of Performance

t Average Time for Movement

To find the depency between I_D and t, they can be inserted into the linear regression model

$$\hat{y} = a + b \cdot x \tag{4.3}$$

to get the well known formula:

$$t = a + b \cdot \log_2 \left(\frac{2A}{W_S} \right) \tag{4.4}$$

The new variables a and b are the linear regression parameters, that are dependent on the input device used.

It has to be noted, that Fitts' Law only regards one dimension. A generalization is the Steering Law [3]:

$$MT = a + b \int_C \frac{ds}{W(s)} \tag{4.5}$$

with:

C being a path

W the width of the path at point s

a and b again are model parameters

When only dealing with a straight path of length L and constant width W, this can be simplified to

$$MT = a + b \cdot \frac{L}{W} \tag{4.6}$$

which is very similar to Fitts' law (equation 4.4).

4.2.2 Scalable Pointer Speed

Both Fitts' Law and Steering Law indicate the tradeoff between speed and accuracy when it comes to pointer movement. In a more practical example, increasing the pointer speed is basically multiplying the pointer movement with a constant per update. Therefore the number of reachable pixels degrades by the same amount, reducing the target size (in reachable pixels).

Our approach uses a tablet as a trackpad. When touching the tablet with a single finger, the pointer moves at default speed over the screen, as with a traditional trackpad. But when two fingers are used, the pointer speed can be controlled using a pinch gesture, even while moving the pointer. This dynamic speed change allows to gain speed when needed to travel greater distances and to drastically slow down when exact positioning is required. The pinch gesture is commonly associated with scaling in touch application. Since slowing the pointer or speeding it up is similar to scaling the movement speed, it is an

intuitive gesture. Additionally it can be done during the pointer control itself and therefore blends into a single action.

In order to click at the mouse pointer's position, a simple (one-finger) tap on the tablet is sufficient. This is the default for most trackpads. The same goes for a double-tap and keeping the finger on the touch surface on the second tap. It simulates a mouse click with the button still pressed while the mouse moves. Since these standard ways of clicking do not integrate well with the scalable pointer movement (using two fingers), we added another way of clicking. When already interacting using two fingers, the user can tap with a third finger to click. If a longer click is needed (e.g., for click-and-drag) the mouse button is considered down as long as the third finger is on the touch surface. Also, a tap using two fingers used as a normal click or could be used as a right-click, should the application require it.

4.2.3 Pointer Zoom

With an interaction method integrating into the above, we want to help the user to find the mouse pointer if they lose track of it. Since we already established two finger swipes as mouse pointer movement gesture, using another two-finger gesture is appropriate. Therefore a double-tap using two fingers will highlight the current mouse position, by increasing the mouse pointer itself by a large amount. The pointer will go back to normal size when other interaction is resumed.

4.2.4 Implementation

The concept was implemented on a 3x3 Tiled-Wall system with a maximal total resolution of 5760x3240 pixels. The tablet we used was an iPad 2 (running iOS 7). Both systems were connected through WiFi.

On the iPad an Objective-C app was written, that connects to the Tiled Wall (using TCP).Normal touches are sent to the main application as they are registered. The touch coordinates are normalized to be as device-independent as possible. Any action registered as a tap is sent as a mouse click. When a double or triple touch is registered, the center point is sent as the touch coordinate and additionally the distance between the touches is measured. When

this distance changes the scaling of pointer movement is adjusted accordingly. This scaling is then applied to any further touch movements sent to the application. This has the advantage of the pointer speed scaling being completely transparent to the main application. For user feedback, the current mouse speed is displayed on the touchscren of the tablet.

The main application running on the tiled wall, on five computers (Windows 7) connected though Gigabit-LAN was programmed in C++. It features a classic WIMP GUI spanning across the whole area of all 9 screens. To aid evaluation, the GUI can be scaled to any desirable size. This scaling does also affect mouse pointer speed, in same same way as if the screens resolution actually changed.

4.2.5 Evaluation

For a preliminary evaluation, the jigsaw puzzle scene already introduced in section 3.1 of chapter 3 was used. Test candidates had to solve one puzzle using a normal computer mouse, one using the tablet working as a normal trackpad and one using the method described above. The order of the method to solve the puzzle was chosen randomly for each of 10 candidates. All puzzle had the same total size of 1024x758 pixels, divided into 25 tiles and had to be solved on a 3000x3000 pixel area. Before solving each puzzle, candidates were given two minutes of training time with the correspondent input method. Each test run, the time between touching the first puzzle piece and completing the puzzle was measured.

The average time for mouse, trackpad and scaling trackpad were 176.2, 499.3 and 635.1 seconds, respectively. This shows that the scaling method outperforms the normal trackpad (which was the case with every single test candidate). Still the method is slower than using a normal computer mouse. This is probably due to the higher experience of the users with the mouse over using a trackpad. Additionally, more training to get used to the scaling functionality of our method may be needed.

Two participants volunteered to take part in an extension to the study, where they showed large improvements (about 50%) in time taken to solve a puzzle with the scaling method after some additional training. In another short test a higher resolution for the whole GUI (including jigsaw puzzle tiles) was

chosen. This increased the time needed to solve the task by about 20% using the mouse, while the performance of the scaling trackpad stayed constant.

A bigger study, involving more test candidates needs to be performed to get more insight into these results. We need to evaluate the influence of experience with traditional trackpads on the test results per participant. Also it will be interesting to test different resolution to find out, if the described method scales with the resolution.

4.2.6 Results

We found indications that our scalable pointer speed is an actual improvement in terms of usability and performance, at least for users of standard trackpads. Of course these findings have to be replicated in a bigger study, but as our method performed better than the normal trackpad with every single test candidate, there is a high probability that the results will be similar.

For the mouse seemingly being a superior input device, we have to find out if this is tied to personal preferences of the users or a general trait. We could apply the same scalable pointer speed to a mouse, e.g., using the mouse wheel as speed adjuster. For combined touch/mouse devices, such as the Apple Magic Mouse, the same scaling metaphor could be used as used with the trackpad.

The speed of the pointer could also be controlled by an individual touch on the border of the screen in order to avoid changing the default way of interacting with trackpads. It might be interesting if this affects results.

The pointer zoom was not evaluated, yet. For an actual evaluation other techniques could be implemented as well, such as permanently resizing the pointer while is moves, maybe according to the current pointer speed scale. Additionally, instead of helping the user to find their mouse cursor, they could be given the possibility of setting it at a certain position. An example is to make bezel swipes (i.e., swiping from the outside of the touch area to the inside) reset the cursor to the appropriate position at the border of the large display and move from that position according to the further touch input.

4.3 Conclusion

In this chapter we presented methods to solve common problems of large display interaction. Through sensor fusion of camera, accelerometers and compass of consumer-level smart phones and tablets we can estimate the position and orientation of the device accurately. The method is unfortunately limited to using markers at this time, but as CPU power of mobile devices increase, this limitation can be removed by using a map of features detected by methods, such as SIFT [73] or SURF [27]. Another limitation is having adequate lighting for the camera to deliver images of good quality. In most, if not all, cases this can easily be provided in the usage environment.

The scaling mouse pointer solves the distal access problem, where targets are generally further apart and harder to hit as screen size increases. By allowing to change the speed of the pointer as it moves, longer distances can be covered more rapidly without sacrificing pointer accuracy as the speed can be reduced when the pointer is near the target.

Chapter 5

Mobile Interaction: Marking Menus

In the preceding chapter, we have shown approaches to solve issues common to large display interaction in general. Specialized Methods, such as those presented in the chapter before that, can be well designed and tailored towards their specific use case. Unfortunately this also means change to the method is needed if the underlying application or any other parameter changes. As these parameters in general depend on the chosen input device or the (work) environment the application is used, change quite often can be needed. Moreover, it is desirable to reuse existing interaction methods. Not only does this save time for the developer, but also does not require the user to learn new ways to interact (possibly within the same application). In this chapter an extended version of *Marking Menus* [67] as well as their application as remote interaction using mobile devices is presented. They support a great number of different functions, and through extensions can be used to control non-binary functions, too. After showing their general applicability, usage and design examples are given, before drawing a conclusion.

5.1 Motivation

Non-trivial applications typically provide the user with a moderate to large number of functions, which need to be mapped to the set of available input devices. Normally, each function is mapped to one device button or gesture or

to one user interface (UI) button displayed. Problems arise when the number of required functions exceeds the number of input device buttons or gestures, or when the number of displayed UI buttons clutters the display. The touch-capable screens of current mobile devices, on the other hand, provide enough screen estate to offer a large number of buttons and mapped functions – significantly more than traditional input devices. However, the naïve approach of using mobile devices disrupts users' workflows, as they have to shift their attention back and forth between the devices' small screens and the main display environment.

We propose to utilize *Marking Menus* [67], a radial menu structure, as a central element of a novel interaction method employing the multi-touch-screen of mobile devices. The touchscreen is used to show (hierarchical) radial menus as they pop up. This enables eyes-free interaction for experienced users who do not need the visual feedback from the mobile device, and leads to increased efficiency, while at the same time the menu structures are kept visible should they be needed. This selection method is extended by the usage of tracking sensors, accelerometers, multi-touch and/or in-menu slider controls to provide a larger variety of interaction possibilities, which are explained in detail later in this paper.

The advantages of this design over existing interaction methods are:

- Eyes-free interaction makes complex user interaction possible without interrupting the workflow.
- Auditory or haptic feedback is given to support eyes-free interaction even more.
- The general approach allows application of the design to a wide array of new and existing applications.
- The ubiquity of smart phones and tablets and their usability for other tasks make this approach very cost-effective.
- As the system uses its own mobile screen it can replace large parts of the application's graphical user interface (GUI) up to the complete GUI in some cases.

- Support for multiple users and the system scales well to the number of users.

5.2 Related Work

Kurtenbach et al. [67, 68] proposed and evaluated *Marking Menus* as an improved version of radial menus. The main difference between Marking Menus and transitional radial menus is the absence of a completely bounded target area for each menu item in the former. Radial menus simply arrange their items in a circular pattern around a center point in one or more "rings." An item is selected when the selection cursor is within the bounds of the item, and the selection is usually affirmed by a button press on the input device (or any other available confirmation gesture). If the selected menu item has subitems, a new radial menu will pop up at or near the current cursor position. Marking Menus only have a single ring of items, and their respective target areas expand infinitely outwards from the center of the menu in a wedge-like shape. Holding the cursor still in the selection area of an item with sub-items will cause another Marking Menu to pop up showing the sub-items, and a confirmation event (button press, or, in the original example, lifting the pen from the display surface) will select the current item. The main benefit of this menu layout and selection mechanism is eyes-free item selection, which was proposed to address the high latency of pen-based direct interaction displays on then-current workstations. Using Marking Menus, users could either put the pen down on the screen, wait for the menu to appear, and select items and sub-items by drawing a stroke from the center into the (wedge-shaped) selection areas, while experienced users could just draw the chain of strokes used to reach a certain item without even waiting for the menu to pop up. (Kurtenbach called this "Expert Mode.") Since the selection areas are theoretically infinitely large, user accuracy is very high and Marking Menus are easy to use. These assertions are backed up by Fitts' Law [49] and Steering Law [3]. While the high-latency problem Marking Menus were originally designed to address no longer matters, their effectiveness still does, and Marking Menus have been incorporated into many modern applications.

Pook et al [90] proposed *Control Menus*, a general improvement that also

applies to Marking Menus: instead of using menus only for selections, the continued motion of an input device after an item has been selected is used to change a continuous value associated with that item. They use an example of a "zoom" menu item, where any continued input device motion after selecting that item directly alters the current zoom level. This method could also be applied to related continuous values by using both display axes simultaneously.

We present the design and implementation of our prototype. A formal user study is not done yet, but is planned as part of our future work. Studies by Kurtenbach et al. [68] are applicable to this interaction method and show its general usability.

5.3 Design

Smart phones and tablets are not only usable as input devices, they also have output capabilities. When designing a user interface, one has to decide whether, and how, to use those capabilities. A device's screen, typically the primary output channel, can be used in a variety of roles: as a primary screen for a focus-and-context displays [18]; as a full additional screen, as in Air Display[1]; to clone the environment's main display, as in remote access applications such as Remote Desktop Protocol (RDP) or Virtual Network Computing (VNC); or even not at all, as in mouse control applications such as Remote Mouse[2]. One reason not to use a device's screen is that having to shift focus between multiple unrelated screens might interrupt a user's workflow.

Our approach uses touchscreens for interaction feedback via radial menus similar to Marking Menus, described in Section 5.2. Kurtenbach categorizes the usage modes of Marking Menus as Novice Mode (waiting for menus to pop up) and Expert Mode (completing interaction before or while a menu pops up). In the remainder of this paper, we refer to Expert Mode as eyes-free mode, to emphasize the fact that it does not require shifting attention away from an environment's main display.

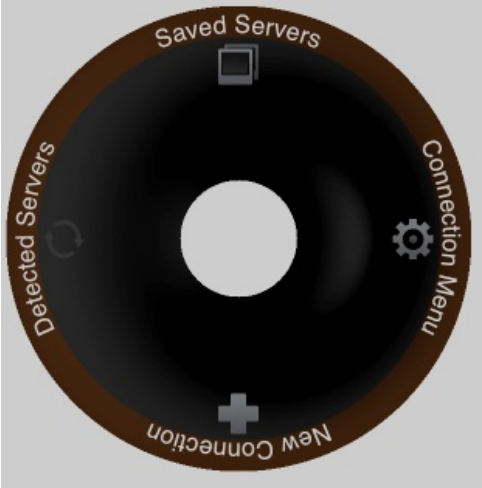


FIGURE 5.1: *The general design of the menu, in this example for four menu items, all with a descriptive name and an icon.*

5.3.1 Menu Design

A radial menu is presented to the user when they touch the screen (see Fig. 5.1). Its design is similar to the original Marking Menu. The menu is centered around the initial finger position, and menu items are selected as the finger enters their selection area. If the selected item has sub-items, a submenu will pop up on selection (see Fig. 5.2). Unlike regular radial menus, the submenus have an additional wedge-shaped *dead zone*, i. e., a zone where no selection is made, along the line from the center of the current submenu through the center of its parent menu. This dead zone improves usability in eyes-free mode: when not looking at the screen while interacting, users could possibly make shorter or longer finger movements they intended to. To account for that, submenus initially move with the user's finger until the movement changes direction. This ensures that the stroke length does not influence item selection.

The wedge-shaped dead zone has the additional benefit of allowing to undo selections, as users can backtrack their strokes through multiple levels of the submenu hierarchy, including canceling menu invocation entirely. This is an extension of the original Marking Menu method. To support eyes-free interaction, the phone can provide auditory or haptic feedback whenever the user tracks back one submenu level.

[1]http://www.avatron.com
[2]http://www.remotemouse.net/

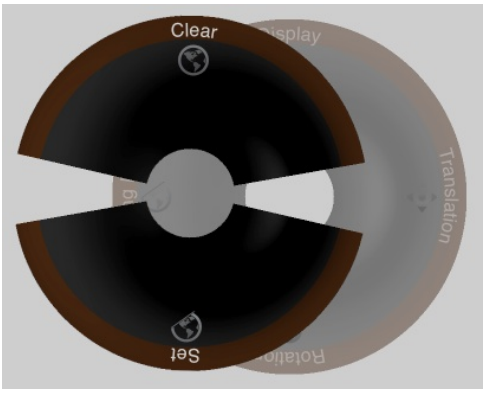


FIGURE 5.2: *The menu with a submenu opened up after the user moved their finger to the left. The submenu is on top of the half-transparent parent menu. Note the dead zone through in sub-menu to improve usability in eyes-free mode and to allow backtracking of selections.*

To avoid unintentional selections caused by touchscreen jitter, another circular dead zone is defined in the center of the menu. This dead zone should be as small as possible, as it defines the minimal length of a stroke to be recognized. The dead zone additionally improves selection accuracy, as the length of a stroke determines the accuracy of measuring that stroke's angle. Due to the rather coarse resolution of current-generation touchscreens, a very short stroke may only be a few pixels long, resulting in inaccuracy when detecting the angle of that stroke. Thus, the size of the central dead zone is a device- and user-dependent configuration parameter.

To reduce display clutter in deep submenu hierarchies, where submenus are drawn on top of their respective parents, only the currently active submenu is drawn fully opaque, while parent submenus are drawn with increasing levels of transparency, i. e., the root menu is most transparent. Drawing the entire menu hierarchy in this way helps users to see their current position in the menu hierarchy and the direction of the most recent stroke, should they lose their place during eyes-free interaction. It also provides valuable feedback for the multi-touch interaction described in Section 5.3.3.

5.3.2 Value Control

Our menu design also supports value selection similarly to Control Menus, as described in Section 5.2. Menu items with associated values are indicated by

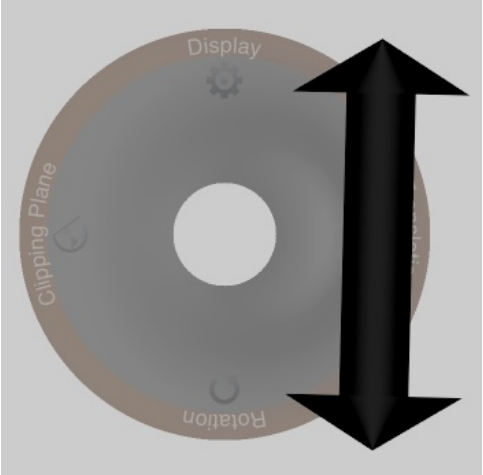


FIGURE 5.3: *An arrow shows up when the selected submenu supports direct value selection, just like a regular slider.*

arrows (see Fig. 5.3). But instead of only allowing simple slider-like control over one or two dimensions, our design allows two modes of operation. In absolute mode it works like a regular slider control, reporting the absolute position of the touch(es) to the application. In relative mode, the actual location of the touch(es) is irrelevant. Only the changes in position are reported to the application instead. Absolute mode is useful for any fixed range of continues values, i. e., whenever a traditional slider control is useful. Relative mode is for situations where the absolute value of a variable is not of interest, but the actual change to it is. A common example for this is the position of an object. The user is interested in moving the object a certain amount of units, instead of setting it to an absolute value. This relieves the application of keeping track of an absolute value (i. e., the position of the slider) that is of little interest.

For many visualization applications 1D or even 2D positioning is not enough. Employing the well-known multi-touch gestures of pinch and rotate, four degrees-of-freedom (DOF) can be controlled at once (x-Direction, y-Direction, Pinch and Rotation). This allows for simultaneous two-DOF-Movement, Zoom and one-DOF-Rotation, for example, to allow the user to drill down into a part of the visualized data without having to apply a new menu selection. Alternatively multiple quantifications can be made at the same time, with each finger that touches the screen functioning as a separate one-DOF slider control, theoretically allowing for a 10-DOF control. The practical limit depends on

actual touchscreen size, the size of the user's fingers and the user's dexterity. Most people should be able to use four sliders simultaneously without serious problems. The control provided might be too coarse or too fine. Therefore the control itself can be scaled using the pinch gesture with a second finger (one-DOF or two-DOF controls) or a third finger (three-DOF and four-DOF). The scaling is visualized by scaling the control on the touchscreen accordingly.

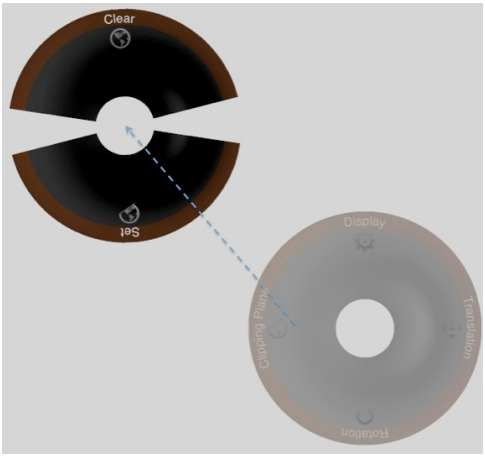


FIGURE 5.4: *Multi-Touch enabled navigation allows the user to detach a submenu and control it with another finger while the original menu still sticks to the old touch.*

5.3.3 Multi-touch Capabilitiy

In our method, multi-touch is not only used for value control, but also for normal menu navigation. As shown in Fig. 5.4, putting down another finger on the touchscreen while a menu item with sub-items is selected causes the submenu for that item to detach from its parent menu, move to the new finger's position, and be controlled by the new finger. The parent menu sticks to the original finger, and move with it, but is otherwise locked as long as a detached submenu is active. If, on the other hand, the currently active menu item is a value selection item, the new finger will invoke multi-DOF value selection as described in Section 5.3.2. If the currently selected item is a regular menu item, then each additional finger will cause a selection event for that menu item, enabling rapid multiple selection of the same menu item by repeated tapping with an additional finger.

Detachable submenus are useful when stroke navigation through a deep

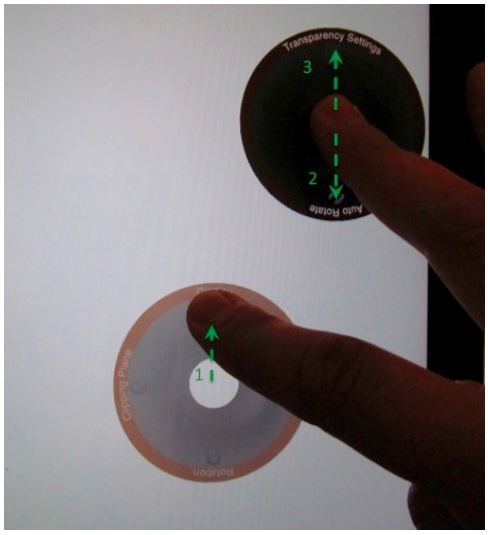


FIGURE 5.5: *Using detached menus as shortcut: The index finger touches the surface and swipes up (1). The middle finger swipes down causing the submenu to detach and toggle the auto-rotate function (2). The middle finger touches the surface another time, swiping up to activate the transparency settings (3).*

menu hierarchy reaches the edge of the touchscreen at some point. In that case, the user can place a second finger on the opposite side of the touchscreen, and continue with menu selection normally. To prevent awkward positions, the initial finger can be released once the second finger starts interacting with the menu hierarchy. If, however, the first finger is kept held down, only the detached part of the menu structure, and not the entire hierarchy, is dismissed after a successful selection. This enables a shortcut for multiple subsequent selections that have the same prefix in the menu hierarchy. For example, if the user wants to execute *left*, *up*, *left* followed by *left*, *up*, *right*, it will be possible to stroke *left*, *up* first, then perform *left* using a second finger, causing the menu at the first two parts of the stroke to be still open after the selection. With either the original or an additional touch the *right* stroke can be performed. A similar example is depicted in Fig. 5.5. Such shortcuts becomes more useful with deeper menu hierarchies. Very experienced "power users" can push the paradigm by using more than two fingers, detaching multiple submenus at the same time. While having a relatively steep learning curve, such expert shortcuts can lead to highly efficient interaction sequences.

After the taxonomy of Foley et al. [50], the methods described up to this

point support selection, quantification, an approximation of positioning and orientation via sequences of quantification, path via sequences of approximated position and orientation, and text when combined with QuikWrite or a comparable approach. True 3D interaction, however, requires at least 6 degrees of freedom, and single- or multi-touch gestures are not an intuitive replacement. While it is possible to chain two individual three-DOF interactions together to achieve six-DOF, this is difficult to handle for the user. Also it is only possible for two-handed interactions, and users should be able to choose one-handed vs. two-handed usage freely for themselves. Another way is to employ the smartphones build-in accelerometers to deliver the current orientation of the phone as additional information for each menu. As orientation provides three DOF (pitch, yaw and roll), seven DOF can be achieved through the combination of multi-touch quantification and orientation. Since 4 of these 7 DOFs concern rotation, a more practical limit is six DOF though. The quantification control described can transmit the current orientation of the device as an additional 3 quantifications. This can be combined into free six-DOF-movement, for example, by letting the position of the user's touches control the x and y position, pinching of the fingers controls the z position (or zoom) and the device rotation can be transferred to the object being moved.

To support clutching (i.e., repeated swipes on the screen to control the same value) and to offer quick access to the last used function, the last invoked action can be part of the menu or can be activated by the use of multiple touches. For example a menu can be designed in a way that a single touch with a swipe to the left and then up lets the user control the position of an object. If the user is not satisfied with the object's position afterwards, a simple touch with two fingers will allow them to refine the positioning without having to swipe left and up again.

5.3.4 Rejected Designs

This section contains a brief overview of design alternatives and an explaination why the actual design was chosen over the alternatives. Selections in Marking Menus can be done in two ways: By drawing continuously connected strokes like in our approach or by drawing separate strokes removing the finger from the touchscreen after each menu level. As each new stroke can start at any

position on the touch surface, there is an advantage of never running into a screen border when drawing the strokes for a selection. As our method has already addressed that issue in another way little is gained by using separate strokes instead of continuos ones. With separate strokes it is impossible to include an undo function in the same intuitive way as described above. Multi-touch interaction also becomes less intuitive that way. If interrupted in the middle of selecting a menu item, a user might lose track of the current state of the menu (i.e., that a submenu is currently active). This cannot happen with the continuous version, as no menu can be active as long as no touch occurs. Curved strokes were not used for a similar reason. Curved strokes allow to use more items per menu level (e.g., different menus for straight line, curved left or right). As shown by Bailly and Lecolinet [13] this increases the complexity of the menu and prolongs the time taken to select something from the menu. Curved Lines tend to be drawn longer than straight strokes, causing the user to run into a screen border more easily.

5.3.5 Hardware Requirements

Our design can be implemented on most of the current consumer-level smart phones and tablets. More specifically, target devices need to support multi-touch, need to have at least accelerometers for six-DOF interactions, and WiFi for communication. Other factors such as screen size or weight influence usability, but are not technically limiting.

5.4 Menu Design and Prototyping

The most important part of this design is the fact that the mobile application receives the information about the menu design from the host application. Therefore only one application on the mobile device is needed to control multiple host applications. There is no need to develop and deploy a new version of the mobile application when the host program changes, potentially saving time and money.

In our architecture the menu layout is controlled via configuration files that are independent of the program implementation, so changes to the menu on the mobile device can be done without the need to recompile the actual program,

allowing for faster prototyping. Output of accelerometers is provided as a simple value manipulation, allowing to make use of these values easily.

Formally the smart device is visible to the host application as two (mathematical) functions:

$$b(N) \rightarrow \{0, 1\} \tag{5.1}$$

$$v(N) \rightarrow [-1, 1] \tag{5.2}$$

Less formally, there are buttons with either the state of 0 (released) or 1 (pressed). These states can be set, reset or toggled by selection of a menu option. This is encoded in function b. v are valuators, that contain normalized values (normalized to the range [-1, 1]). These valuators can be changed using the slider methods described above, or by using the device's sensors (e.g. accelerometers). The host application needs to map these valuators and buttons to its internal functions, a procedure that has to be done only once for each available function. The layout of the menu (and therefore of the mapped functions) can then be changed easily without having to change any internal implementations of the host application. Also the mapping allows to easily transform and/or combine incoming values for the host application should the need arise.

This serves as a kind of model kit for interaction designers to work with and come up with a feasible user interface on the mobile device quickly and easily.

5.5 Implementation

Unlike most other approaches, our proposed interaction method is neither specifically tailored towards nor implemented in, a single application. It is designed as a toolset for application or interaction designers wanting to include smart phone or tablet-based interactions in their applications. As our method is implemented as a third party plug-in for the Vrui toolkit[65, 66], it can easily be included in any Vrui-based VR application.

5.5.1 The Virtual Reality User Interface (Vrui)

The Vrui toolkit is a set of C++ libraries supporting the development of portable VR applications. Vrui-based applications run without change on a wide variety of hardware platforms, including single user workstations, 2D and 3D Powerwalls, low-cost VR environments based on 3D TVs, head-mounted displays, and CAVEs. Vrui's primary aim is to make applications not only work on all platforms, but to make them as useable as platform-native applications. Vrui also includes a number of non-VR-related lower-level libraries to simplify the general application development process, and to foster code reuse and software interoperability. Vrui runs on Unix-like operating systems, primarily Linux, but also including Mac OS X $\geq$ 10.4.

One additional convenient feature of Vrui is its input device abstraction, which treats input devices as having any number of buttons and/or valuators, which have binary on/off statuses or continuous quantification, respectively, which satisfies the requirements listed in Section 2.2.1. Additionally, input devices may have an associated position and/or orientation. The primary reason to choose Vrui to implement our design, however, was that it provides much of the higher-level functionality usually left to individual applications, e. g., 3D navigation and GUI interaction, at the toolkit level. This means that our menu system can replace parts of that shared functionality in a completely application-independent manner.

In more detail, our prototype implementation is a client/server-based plug-in, with the plug-in side acting as a server for one or more mobile devices. The server typically listens on a local wireless network for incoming connections, and also uses zeroconf[3] to broadcast its availability and simplify connecting devices ad-hoc during run-time. Client/server communication is based on TCP, and therefore works with standard WiFi systems and across routers, enabling connections over 3G as well. Data is exchanged using a simple binary protocol. The server sends the menu structure specified by the application to each mobile device upon connection, and each device subsequently sends all relevant user interactions, e. g., selections or quantifications, back to the server. The server translates all events into the appropriate Vrui API calls, and forwards them to the Vrui kernel, and then to the application. The communication protocol is

[3]DNS-SD, see http://www.dns-sd.org/

extensible and can be tailored towards a specific application should the need arise. Extensions are simple to implement on both sides and can make use of the already implemented features.

In high-end VR environments such as CAVEs, mobile devices would typically be outfitted with six-DOF trackers to make them fully functional replacements for traditional six-DOF input devices such as wands, and to directly support position- or ray-based object selection. Desktop or low-cost VR environments, on the other hand, do not typically have tracking systems, and in those configurations, each device is assumed to be at a fixed position. If a device provides orientation measurements based on accelerometers and compasses, standard Vrui methods can be used to turn it into a ray-based device similar to a desktop mouse, or into a plane-bound 3D input device by intersecting its selection ray with the VR environment's screens. Even if the device is moved from its assumed position, these interaction methods still work quite well.

5.5.2 Mobile Device

The prototype device-side application was developed in Objective-C for an Apple iPhone 4S and an iPad 2. Both of them feature the A5 dual-core CPU at 1 GHz, 512 MB RAM, a capacitive multi-touch-screen with a resolution of 960×640 pixels at 326 ppi (iPhone) and 768×1024 pixels at 132 ppi (iPad), three linear accelerometers, and a gyroscope. Both devices support WiFi and bluetooth connectivity, and run the latest version iOS 5 (5.1.1). The prototype application can run on other iOS devices as well, but using older devices might result in inaccurate sensor readings and jittery interaction when using the device orientation.

The device-side application implements the client component of the distributed architecture. On startup, it connects to an application-side server, found either via Service Discovery, or a manually entered host name. Upon connection, the client receives the application's menu structure and builds the menu's visual representation. If the application side requests orientation measurements, the client sends streaming orientation data at the maximal rate of 30 Hz to minimze lag. User interface events such as selection or quantification are reported asynchronously as they happen.

5.5.3 Extensibility

The communication protocol is extensible and can be tailored toward a specific application should the need arise. Extensions are simple to implement on both sides and can make use of the already implemented features.

5.6 Examples

To illustrate the somewhat theoretical possibilities presented in the previous sections, we show two concrete examples of how to interact with VR applications. The first one shows how to emulate a mouse or trackpad connected to the application. The second example uses one of Vrui's packaged sample applications, which was not altered in any way, to show how our method can be applied in an application-independent manner.

5.6.1 Simple Mouse Control

While this method is designed to avoid the need of a mouse cursor on the large display, it can be designed to work with it. For this simple example, we bind a 2D value selection to the one-finger interaction, thus making the remote app behave like a trackpad. For simple 1-button mice, such as used with early Mac-Computers, two finger touches can simply be mapped to a click. Since today's computer mice usually feature two buttons and a mouse wheel, we instead opted to have a radial menu invoked with two fingers. The menu consists of the following options:

- **up:**
  Invoke the function of a normal (left-) click
- **down:**
  Invoke the function of a right mouse button click
- **right:**
  Shows a 1D-slider control to emulate the mouse wheel
- **left:**
  Invokes the same function as a *press* on the mouse wheel

This shows (as a baseline), that the method is capable of emulating all functions, a normal computer mouse can provide.

5.6.2 Earth Model

The next example is a virtual globe application used to visualize global-scale 3D geological or geophysical data such as earthquake hypocenters, tectonic plate boundaries, subduction zones, or mantle convection cells. The globe is rendered as an image-mapped surface layer that can be made transparent to reveal sub-surface features, and can additionally show a latitude/longitude grid. The mantle/outer core and outer core/inner core boundary surfaces can be rendered as transparent spheres. Using standard navigation tools provided by the Vrui toolkit, users can freely navigate around or through the globe at arbitrary scales, using navigation metaphors appropriate for the actual environment type.

Our mobile device plug-in uses the following interactions to control the application:

1. **Position of the Earth Model:**
   The globe can be moved freely in 3D space with one direction doubling as close-up function.

2. **Rotation of the Earth Model:**
   To view all sides of the globe it must be rotated. This is possible with 3-DOF rotation or with an auto-rotate function that works similar to real-world earth rotation.

3. **Transparency:**
   The transparency of the surface, the latitude/longitude-grid, and the inner and outer core boundaries can be controlled independently. This is normally done using a graphical user interface provided by the application.

4. **Clipping Plane:**
   It is possible to set up a clipping plane to cut away part of the globe and data contained therein to reveal occluded features. The user has to set the position and the rotation of the plane.

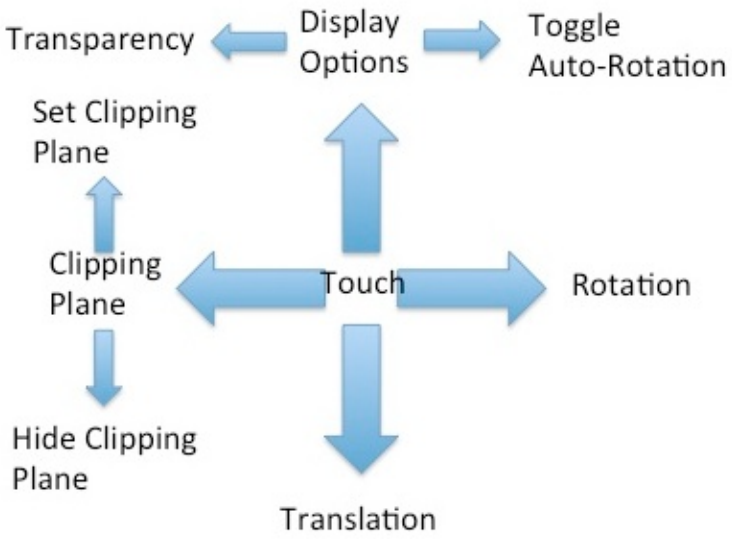


FIGURE 5.6: *The layout of the menu used to control the Earth Model application.*

In most environments, these interactions include pointing at some target, e. g., the globe, a dialog box item, or the clipping plane's icon. This can lead to problems such as occlusion of the target item or difficulties of actually hitting the desired target with a selection ray [83, 87]. Additionally, using pointing-based interactions can lead to fatigue after prolonged use. Therefore, this example does not make use of the mobile device's position at all, which has the additional benefit of making multi-user scenarios easier due to reduced chances of users interfering with each other.

The menu, invoked with a single touch on the mobile device's screen, consists of only four items on the first level of hierarchy to keep it simple. This can be afforded due to the small amount of functionality in this example. With larger sets of application functionality there is a natural trade-off between the depth of the menu hierarchy and the number of items per level. The general layout is shown in Fig. 5.6. Rotation can be selected by swiping to the right, and without lifting the finger swiping up or down starts rotating the globe around the x-axis. Using a second finger allows rotation in 3-DOF. Dragging the fingers in x- or y-direction on the touchscreen causes rotation around the vertical or horizontal axis, respectively (i. e., as if the globe itself was dragged in that direction). Rotating the fingers around their common center point covers rotation around the user's view axis. Similarly, Translation

can be invoked with a downward swipe. Using multi-touch, the position of the globe can be controlled in all three directions. Dragging the fingers in x/y-direction also moves the globe in the appropriate direction and pinching with two fingers controls the z-movement (as z-movement doubles as zoom). Additionally rotating two fingers around their common center also rotates the globe. Swiping upward on the screen causes the *Display Options* submenu to appear and with a swipe to the right the auto-rotation of the globe can be toggled. Swiping Up, then left allows to control the transparency settings of the visualization, with the first finger controlling the latitude/longitude-grid, the second finger the surface, etc. Clipping planes are controlled by a swipe to the left and the either downwards to hide the clipping plane or upwards to show it. When selecting to show the plane, its origin can be moved using the same method as positioning the globe while at the same time the phone's orientation controls the orientation of the clipping plane. With this approach the plane can be set in one continuous action. The last action executed can be recalled by touching the screen with two fingers. This allows clutching, if the screen is too small. If the user swiped down with one finger to control translation, for example, The next touch with two fingers automatically controls translation without the need to swipe down first. This is exploiting the feature of the system remembering the last interaction and assigning it to the menu activated by two simultaneous touches.

The same menu can basically be used for any application displaying 3D models, as medical or geographical data, for example. With little effort it can be expanded in many ways for more complex applications.

5.7 Conclusions

We have presented our eyes-free interaction method using consumer level smart phones and tablets. This method can be used effectively with applications for knowledge discovery and employs multi-touch and the phones sensors to allow full 3D interaction. Being eyes-free users can interact without interrupting their workflow to look at the input device. With its simple consistent design it can replace traditional menus and even whole dialog boxes. Our method is application-independent and can be used in other environments (e.g., Desktop)

as well. Applications do not have to be altered to make use of this approach, but the prototype can be customized to include application-specific behavior.

A possible extension is device-based authentication. Devices can send public keys to the server on connection. Instead of a passphrase, the menu can be used to enter a combination of strokes serving as a password (similar to YAGP [53]). To prevent password sniffing, the communication can be encrypted using Transport Layer Security (TLS). The whole concept is also transferable to other application domains. When a multi-touch trackpad is available (as with most Macintosh computers) the menu can be controlled the same way. Unfortunately, the use of sensors to control rotation is then no longer possible.

The paradigm of our design is usable for a number of other application areas as well, such as desktop computers, tv sets, etc.

This method was also applied to the application described in the next chapter.

Chapter 6

Mobile Interaction And Virtual Reality Visualization: Application In Embedded Systems Development

In this chapter we will present a VR application that leverages collaboration between safety experts and hardware engineers. This application runs under a multitude of usage environments, in both single- and multi-user scenarios. Stereoscopic highlighting extended with a reflection layer is used in this application to facilitate on the possible availability of 3D in the application by encoding properties of the data as depth in a 2D graph. The Marking Menu technique presented in the previous chapter was implemented into this example application. It is part of the goal of this thesis, in order to provide an example of a scalable portable multi-user application.

6.1 Motivation

As part of the goal of this thesis, this application includes the following requirements:

- Independence of special usage environments, especially with respect to the availability of stereoscopic displays.
- Scalability to display sizes

- Multi-user capabilities

In order to develop an application that is more than only another proof-of-concept, we decided to create an application for a real use-case.

As part of a project called ViERforES (Virtuelle und Erweiterte Realität für Sicherheit und Zuverlässigkeit Eingebetteter Systeme, Virtual and Augmented Reality for Safety and Reliability of Embedded Systems), founded by the german federal ministry of education and research (BMBF), an application was to be developed to visualize different aspects of safety in embedded systems.

Embedded systems are computer systems embedded in other products. Examples include everyday items, e.g., dishwashers, car electronics, modern HiFi-systems and are prevalent in other areas, too. In areas, such as aircrafts, power plant components, etc. safety and reliability aspects are very important, as failures can be costly and/or outright dangerous. Safety engineers from the domain of software engineering are developing mechanisms to increase safety and reliability.

Our requirements match to the ones for the safety visualization:

- **Scalability to display sizes:**
  Due to the increasing complexity of products in general and embedded systems especially, the safety models of these systems also become more and more complex. Since large displays allow to present more information at once to users, they can help in understanding large models. On the other hand, users need to work on their own workstations, which only feature one or two monitors.
- **Multi-user capabilities:**
  Especially when dealing with big models, multiple safety engineers work together. Additionally when safety/reliability issues are found, they need to be communicated to the system engineers. Since safety models can deviate from the actual hardware/software model, it is not a trivial task to find out, which real-life systems are affected by these issues. Using a visualization for this will be helpful.
- **Independence of special usage environments, especially with respect to the availability of stereoscopic displays:**
  As stated above, safety engineers and system engineers need to collaborate, in order to improve a planned or existing embedded system. To

help with that collaboration, providing both parties with their own view of the system will help understanding any issues and/or context for all involved. Especially for hardware engineers this means providing a CAD model of the system. Of course stereoscopic displays provide a better view for these 3D models. Since stereoscopic displays are not the standard, we want to support both 2D and 3D displays.

6.2 Safety/Reliability Analyis

Safety and reliability aspects are important when developing embedded systems, since safety and reliability in important in many areas where embedded systems are employed (e.g., aircrafts, power plant components, etc). With the increasing complexity of these systems, the difficulty of detecting and analyzing failures increases as well [59, 60]. Several approaches have been proposed to analyze failures on those systems, such as failure mode and effects analysis (FMEA) [96], or Markov Analysis [64]. The technique implemented in our application is called Fault Tree Analysis (FTA), since it is a top-down approach, capable of identifying multiple sources of failure at once.

As other (functional) computer systems, embedded systems consist of hardware and software components. These components can be interconnected in various complex ways. As systems grow bigger, they tend to be distributed over more components communicating with each others. This separation of functionality into smaller components helps to develop complex system from less complex systems, much like a construction kit [72].

A good explanation of FTA is given in Kaiser et al. [59]:

The concept is to start with a failure event or hazard state and to trace its influences back until the basic influence factors are reached. The resulting influence hierarchy is depicted as an upside-down tree with the failure event (referred to as "top-event") at its root. Mainly two connectives are used to express how influences contribute to a consequential failure:

- the AND connective, indicating that all influence factors must apply simultaneously, and
- the OR connective, indicating that at least one of the influence factors must apply to cause the failure

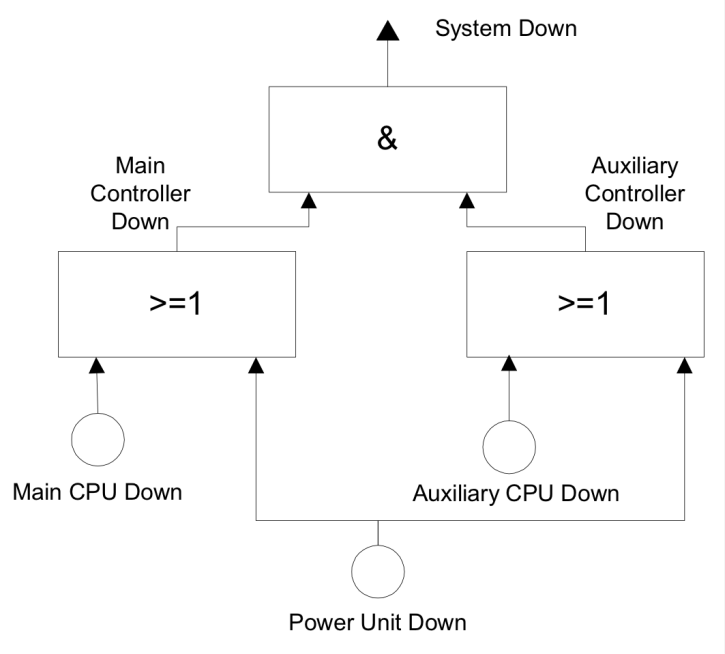


FIGURE 6.1: *Concept of Fault Trees. Basic Events (Round) are connected through gates (rectangular) to the Top-Event (Triangle).*

This leads to graphical representations, as for example Figure 6.1, a Fault Tree (FT). The top-event is connected through two gates to each basic event. A qualitative analysis can show the (minimal) combinations of basic events that cause a system failure (in this case: Main CPU Down + Auxiliary CPU Down, Power Unit Down). Those combinations are called the Minimal Cutsets (MCS). Qualitatively, the probability of the system failure can be calculated if the probabilities of the basic events is known (and they are stochasticaly independent) [59].

As Fault Trees get bigger and more complex, it gets harder to design and understand the system modelled. To tackle this issue, Component Fault Trees (CFT) were developed. They extend the normal model of FTs by adding new elements to the graph: Components. They are containers, basically black boxes with connections going in and out. The behaviour of components is defined by a component fault tree, without a top-event, but instead with ports, that map to the connections that can be made to the component. This allows a divide-and-conquer approach when designing and analyzing safety models. Safety experts can embed fault trees into other fault trees, to partition large fault tree

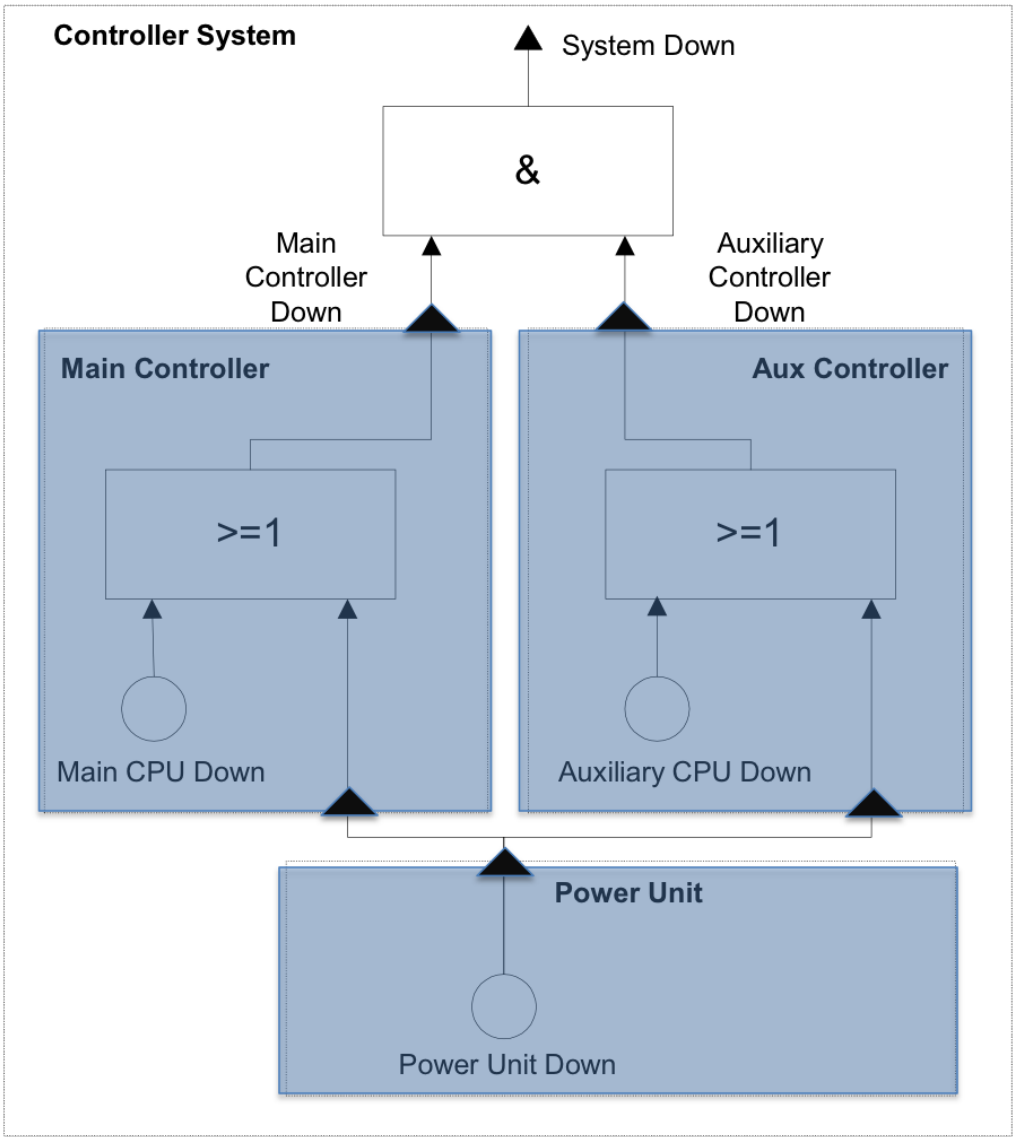


FIGURE 6.2: *Concept of Component Fault Trees. Building on conventional Fault Trees, CFTs add components which are CFTs themselves, that can be included in other components.*

structures into small parts, that can be analyzed individually [59, 60]. Figure 6.2 shows the FT from Figure 6.1 remodelled into a CFT. The representation of a CFT is a Directed Acyclic Graph (DAG).

The main advantage of the CFT over a normal FT is the decomposition and therefore complexity reduction of the system model. However, viewing the whole system is usually time-consuming, as the parts of the system have

to be understood separately. Their interdependence is only visible through their container components. Additionally interdependence might be hidden by the decomposition, if the components in question lie on different levels of components. Our goal is therefore to provide an easy overview of the system as well as detailed information.

6.3 Application Design and Implementation

When modeling embedded systems, components can be either software or hardware. Since FTA lies in the domain of software engineering, most hardware engineers will not be proficient with the technique. Thus, they have problems in easily understanding which component in a CFT corresponds to which hardware component in the real embedded system. We tackle this problem by providing a CAD model of the hardware in question, a view hardware engineers are used to. Linking between both visualizations provides the needed information. An example can be seen in Figure 6.3. These visualizations are

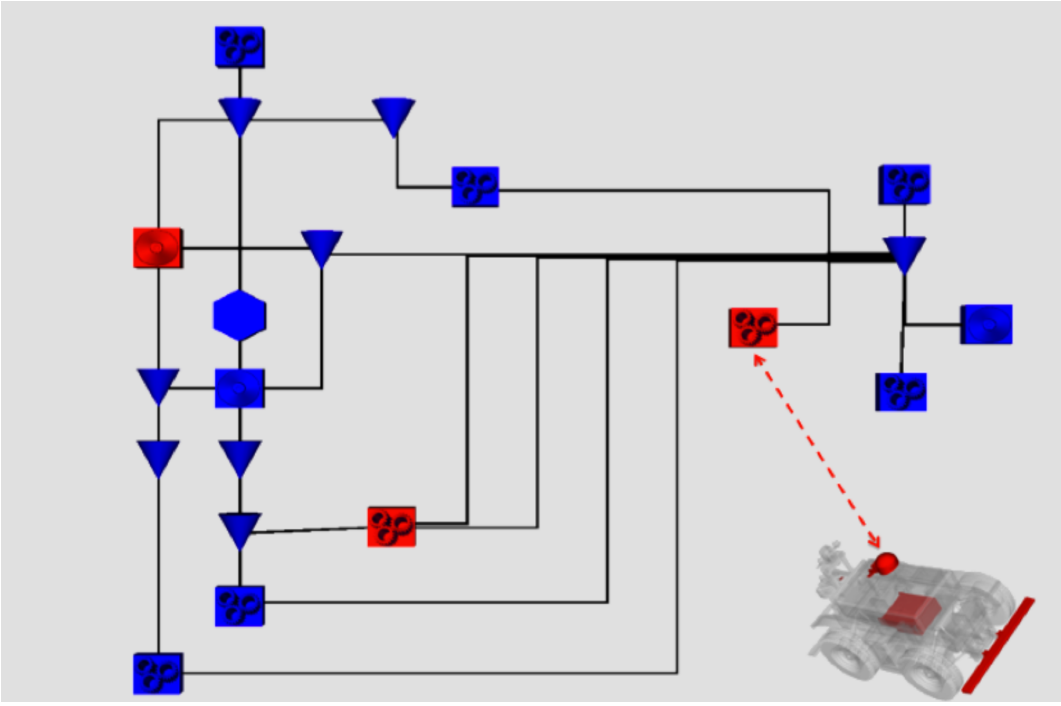


FIGURE 6.3: *The Big-Small view integrates the 3D model and the abstract representation of the safety scenario.*

presented in a VR application. This allows to display the hardware model as

realistically as possible. The safety data was originally planned to be shown as a normal CFT in its traditional representation. Looking at traditional tools to visualize FTs, like ESSaRel [102], UWG3 [59], or Cecilia OCAS [28], the full FT is shown. Colors, text and/or shape is used to convey information to the user. The tools are designed to allow the creation of FTs. The analyzation takes more effort with those tools, since support is limited. In ESSaRel, for example, it is possible to have an automatic qualitative and quantitative analysis. However, the output is a long text file, that is not suited for further evaluation. To find connections between the MCS or the involved components, users have to reenter the data into another program and add any data they want to use for their analysis by hand.

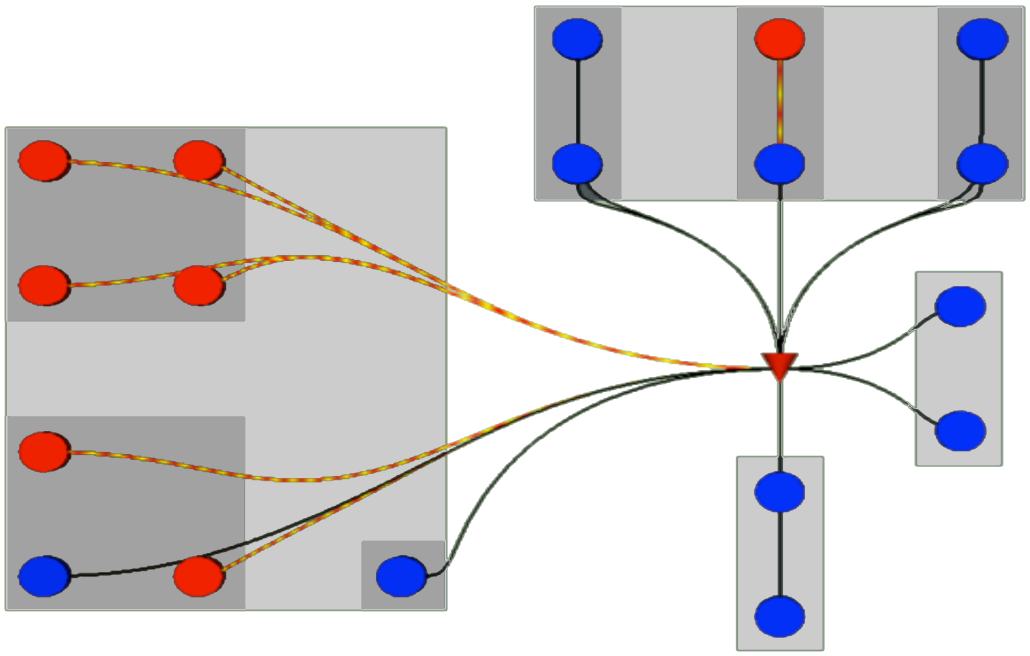


FIGURE 6.4: *A graph with all components expanded. It only shows Basic Events, the top-events and the components as shapes grouping them.*

To start we found the main goals of our users (the engineers) to be:

- Find critical system parts and their contribution to a system failure
- Find the severity of their influence on system failure
- Find interdependencies of system parts leading to failures
- Location if the faulting component (physical or logical)

The focus of our applications is on the (Visual) Analysis. Therefore we chose for a non-standard approach for the visualization. While keeping the node-link-diagram for representing the safety information, we do not simply display single CFTs. Instead we do not show the gates included in the CFT. This heavily reduces visual clutter in the graph. While the information lost is valuable, it is still available through interaction, as explained later in this chapter. Our application presents the CFT containing the top-event on startup. Each component in the CFT (which is itself again a CFT) can be expanded to show the contents of that CFT directly in-place. This allows the safety analyst to easily see the interconnections between the different components, a feature not available in prior software. Figure 6.4 shows the graph maximally expanded.

Any element in the safety graph can be set to be in a failure state and will be marked in red. If any failing element or combination of failing elements cause another element to fail, the corresponding element and its link will also be shown as failing. The links are animated to show the direction of the failure flow. Deviating from the usual constraint, that only basic events can fail, users are allowed to have any component in the graph fail. This speeds up the analysis process, as the analyst does not have to look for certain combinations of basic events to make the desired component fail.

To link the CAD data with the safety data, we also highlight all failing components in the CAD model in red. To get better insight into the interior of the CAD model, an explosion view is also available.

The graph representing the safety data can be viewed in different layouts. Algorithms available include a radial layout algorithm [44], spring layout and orthogonal layout. The latter is the preferred layout, as it utilizes screen space efficiently. It also reduces edge-crossings in the layout [58].

Both visualizations or views can be positioned freely on the screen by the user. Five default placements were provided to speed up the process. Both views can be placed side-by-side at the same size, to have a comparison between both data views. If users want to focus on only one view, that view can be enlarged and centered on the screen, while the other one is scaled down and placed in one of the corners. Alternatively, both views can be shown at screen size, with the currently regarded one placed in front of the other one. We

presented this to safety and hardware engineers, and both groups appreciated the idea.

Another goal of the application is for it to be runnable under several different environments. Large displays, especially Powerwalls, are targeted since they help collaboration between the engineers. For preparation or single user evaluations, normal PCs are also an important target. The workstations often have two screens, that should be utilized. But also single monitor setups are supported, e.g. on laptop computers. The displays can either be 2D or 3D, with 3D of course being preferrable.

The actual implementation was done in C++, using the framework VRUI [65]. This satisfies the requirements for different runtime environments. The visualization application (called EssaVis) takes safety data in the format of the ESSaRel designer. CAD data needs to be provided in the well-known VRML(Virtual Reality Modeling Language) format.

VRML is a scenegraph-based format, that contains a tree structure of nodes with different types [1]. Node types follow a class-like hierarchy, where some node types inherit the behavior of other types and extend it. There are grouping nodes, that all share the behaviour of the standard *Group* node. But the *Transform* node also has a designated transformation that is to be applied to all its children. Important to our application is the *Anchor* node, that allows nodes to be named. There are a set of nodes containing geometry, either in raw vertex form or as parameter sets of basic geometric objects, such as boxes or spheres. Appended to these nodes are other nodes containing all information about the object's appearance. The whole specification for VRML is more powerful, but most other functions are not needed for our application. Vrui comes with a built-in VRML parser that translates the data into an internal scene-graph format used for rendering.

ESSaRel data is encoded in multiple XML-files. To make using the data easier, we added a preprocessing step to the application, where the data is transformed using XLST (Extensible Stylesheet Language Transformations) to a single file that contains all needed data. This single file can be quickly parsed by the main application, reducing general start-up time, since multiple runs using the same data does not need further preprocessing steps.

Connections between the two data sets are made through annotations in

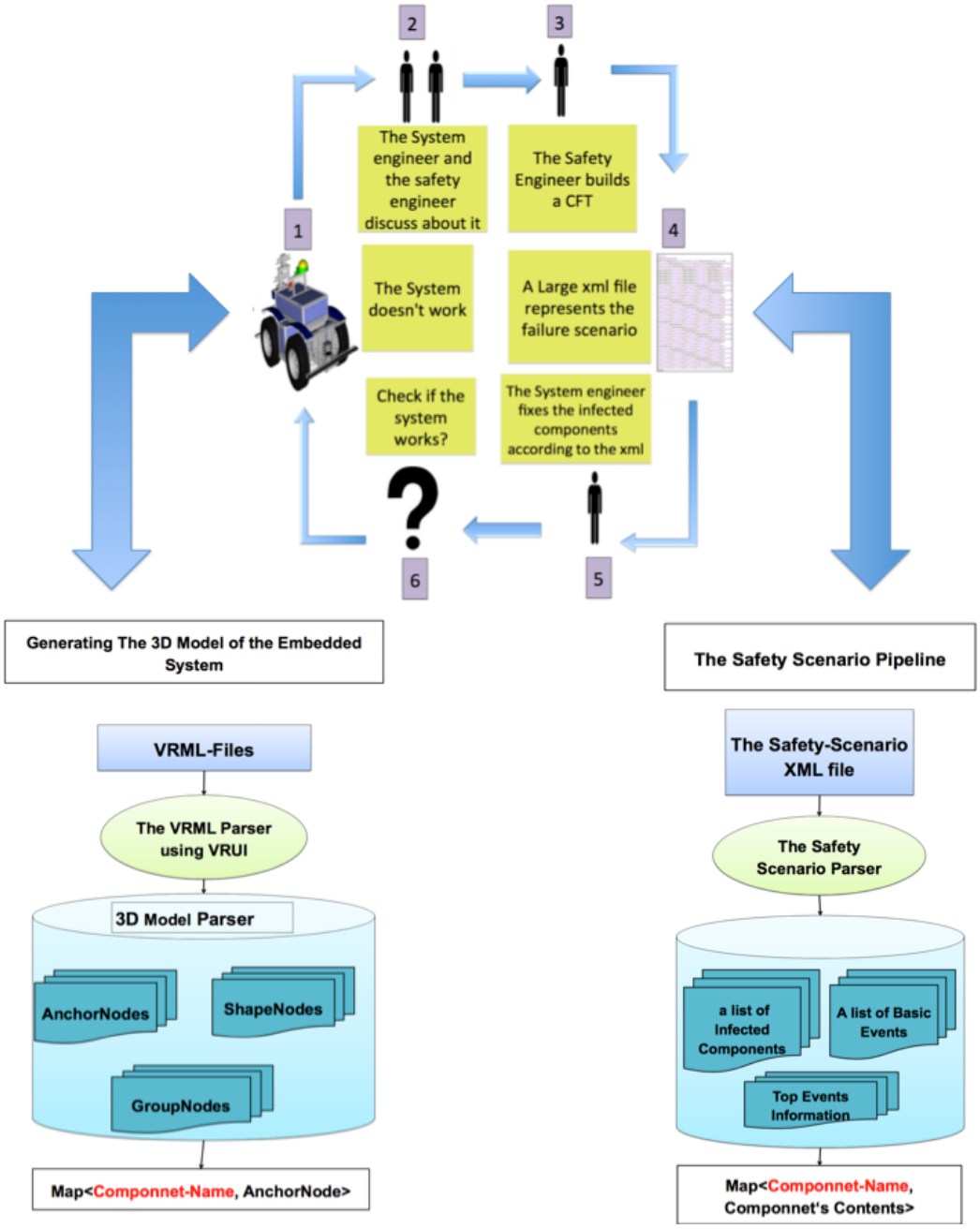


FIGURE 6.5: *(a) Generating the 3D model pipeline. (b) The safety cycle analysis. (c) The safety scenario pipeline, the two red arrows indicate the connections between the safety cycle and the two pipelines.*

the safety data, that refers to Anchor-nodes in the VRML data. Annotations also specify if a safety component is software or hardware.

We ran the application on several normal PCs with single and dual monitor

setup, including 3D monitors. For collaboration meetings we also installed the application on a 3x3 Tiled Wall with 5 computers having one GPU per monitor and a two-projector powerwall with passive stereo driven by a single computer. For testing and presentation purposes we also used a 3-walled CAVE system with 7 computers to run the application. In all cases EssaVis worked fine without drawbacks. To overcome the issue of having different input devices in each case we also used a mobile device for uniform interaction.

6.4 Mobile Devices

As described in this chapter's introduction, this application runs under a wide range of systems and environments, including but not limited to standard PCs, large displays and CAVES. To reduce the complexity to the user, our goal is to minimize the number of different input devices used throughout all usage scenarios. The regarded usage scenarios are

- **Personal Computer:**
  In this most common use case, the hardware or safety expert is working at their own workplace, probably alone. Interaction is done through the traditional mouse-keyboard-combination. Additionally a 3D-mouse can be employed.
- **Laptop Computer:**
  This is special case of the Personal Computer. The difference is that Laptops are usually not stationary. The mouse is often replaced by a built-in trackpad.
- **Tiled Wall:**
  These large displays are usually employed in collaborative settings, where multiple people work together on a problem or are presenting results. Input devices range from traditional keyboard-mouse-combinations to 6DOF-Flightsticks.
- **Powerwall:**
  Large displays capable of presenting stereoscopic contents are called Powerwalls. Therefore this is similar to the Tiled Wall use case, with a special emphasis on using specialized 3D input devices.

- **CAVE:**
  In these immersive environments input devices are often specialized devices, light flightstick or pointing devices. Users typically work alone.

To have a common interaction device, we employed Smart Phones and Tablets with the Marking Menu technique described in the previous chapter. Besides establishing a common interaction metaphor for all use cases, this also enables scalable multi-user interaction.

6.4.1 List View

As the domain experts using the application frequently use lists of the safety components to work with, these lists are included in the main application. Unfortunately these lists are too large and there are too many of them to be displayed on the screen without scrolling or filtering, even on large displays. Since these lists are important, they act as a bottleneck when it comes to multi-user interactions with the application. To circumvent this problem, we decided to deviate from the pure eyes-free approach. As an addition to the Marking Menu control, we added the lists directly into the mobile app (see Fig. 6.6). Working with the lists on the mobile device can lead to the user having to divert their attention between the mobile device and the main screen. Still, this is less disruptive than having to wait for another user to finish their work with a list before having the ability of using it. The lists of safety components have the same contents as the list presented on the main screen. Also their interaction is the basically same, but optimized for touch interaction (i.e., bigger buttons, directly at the item to avoid having to select before clicking). To avoid accidental interaction with the list when interacting eyes-free, the mobile app is divided into the Marking Menu mode and the list mode.

6.4.2 Menu Layout

For the Marking Menu we decided to include the following functionality (ordered in assumed usage frequency):

1. Full 6DOF positioning of the CAD model.
2. 2D-Translation, 1DOF-Rotation and Zoom of FTA graph.
3. Switching to list mode.

FIGURE 6.6: *User interacting with the safety element list.*

4. Resetting the CAD model and FTA views to one of five defaults.
5. Changing the layout algorithm of the FTA graph.
6. Set/Remove and Position a Cutting Plane.

Interaction with the CAD model is available through a swipe *up*. This opens a subselection, where translation of the model is available with a swipe to *right* and rotation to the *left*. Zoom and Z-Rotation is available in both modes through the well-established pinch and rotation multitouch gestures.

A swipe *down* opens another submenu with *left* enabling FTA graph interaction (all of Translation, Rotation and Zoom). *Right* in the same submenu leads to a third menu where the user can control the Clipping Plane.

List mode can be enabled with a simple swipe to the *left*.

Swiping to the *right* opens the layout-submenu. Here a swipe *up* presents all four available graph layout algorithms. *Down* allows to reset the views to

one of five defaults. Four of them are selected by swiping to the upper/lower left/right and the fifth, most often used view default is selected by simply leaving this submenu, making it easier to access.

Using two fingers to interact invokes the last used function. This is especially handy when this is a 3D interaction, as it allows simple clutching.

With this menu layout, all functions can be invoked at a depth of maximum three submenus. Also most menus have only 4 or 2 items, making the stroke very easy to execute.

6.5 Stereoscopic Highlighting and the Reflection Layer Extension

This section describes the technique of stereoscopic highlighting and its extension, the reflection layer. The concepts are presented briefly, since they are implemented in the application described in this chapter. An evaluation of the methods can be found in the next chapter.

The depth cue visible on 3D display is used to encode an ordinal property of nodes in a node-link diagram. It is very useful to enhance 2D visualizations on stereoscopic displays.

6.5.1 3D Graphs

There are many examples in the literature for 3D graph layouts. Examples are the hyperbolic layout [70, 71, 79–81] or the cone tree layout [92]. Using Treemaps [103] in 3D yields the Treecube [99]. Fairchild et. al.'s semNet [48] is yet another example. These methods use lighting and/or animation to help the user to perceive the depth. The main drawback is the occlusion of some layout elements by others.

Also, 3D layouts have no added benefit for the user if they do not have a special 3D display to perceive the actual stereoscopic effect. But if stereoscopy is available for 3D visualizations, there is a benefit [55]. Cockburn and McKenzie found that 3D without stereoscopic effect does not utilize the spacial memory more than 2D [36]. From the lack of inherent spacial relation of many data entities in information visualization Ware and Franck [106] concluded 3D

visualizations not being an appropriate solution. They also state, 3D visualizations need additional interaction techniques, thus getting more complicated. However, they can recommend it, if real-time interaction and real stereoscopic clues are available [106, 107].

In other approaches Peterson et. al. [89] used depth to separate overlapping labels in a 3D world. Deller et. al. [41, 42] used depth variation to filter data sets and to highlight data. Highlighting data through depth is intuitive, since it places more important data nearer to the viewer than less important data.

The third dimension can also be used to encode data aspects [37, 45]. Brandes et. al. [31] use depth to show different versions of a graph to present its evolution through time. Alper et. al. [6] use depth to show the importance of graph elements by rendering them on a single plane closer to the user.

Nakayama and Silverman [84] found out, that it is possible for users to search for an item with a certain depth and other attribute (e.g., color or shape) in parallel, which is not possible for two other attributes.

Ware and Bobrow [105] showed the importance of highlighting for identifying adjacency in node-link diagrams. The propose to not only use color or shape, but also animation and/or multiple clues at once.

6.5.2 Highlighting in Node-Link Diagrams

With the limitations of 3D layouts in mind, and regarding the results of the research about alternative uses of depth perception, we opted for a new approach. Since we want to show the importance (i.e., the safety criticality) of graph elements, highlighting is needed. Extending on the idea of Alper et. al. [6], important nodes of the graph should be presented closer to the user.

A first naive approach would be to just map the z-direction of each graph element to its importance value. This leads to problems, due to the perspective projection used in common 3D applications (such as EssaVis). Objects can appear to be closer or further away, depending on their x/y-position [6]. A compromise is to cluster the nodes into a certain number of sets according to their importance value. Each cluster is then presented on its own z-layer, but still within the original (x/y) layout of the graph.

This allows to visualize importance of graph elements in a intuitive way, while still allowing other visual clues (e.g., color, shape) to be used to convey

further information. This way we can use the shape of graph elements to express their type, as common in other safety data visualizations. Color is used to show failure state and we put a small icon on all components to have the user distinguish between hardware and software components. Using stereoscopic highlighting we can still present criticality of components.

An evaluation of this method and usable number of layers can be found in the next chapter.

6.5.3 The Reflection Layer Extension

Stereoscopic highlighting enhances a 2D visualization when presented on a 3D display. Since it uses the depth perception of users, it is not applicable on standard 2D displays. If the depth is explicitly shown in a visualization, this method is also usable in 2D. Additionally this improves the depth perception even on stereoscopic displays, and for depth-blind users (According to Alper et al. [6] 8–10%).

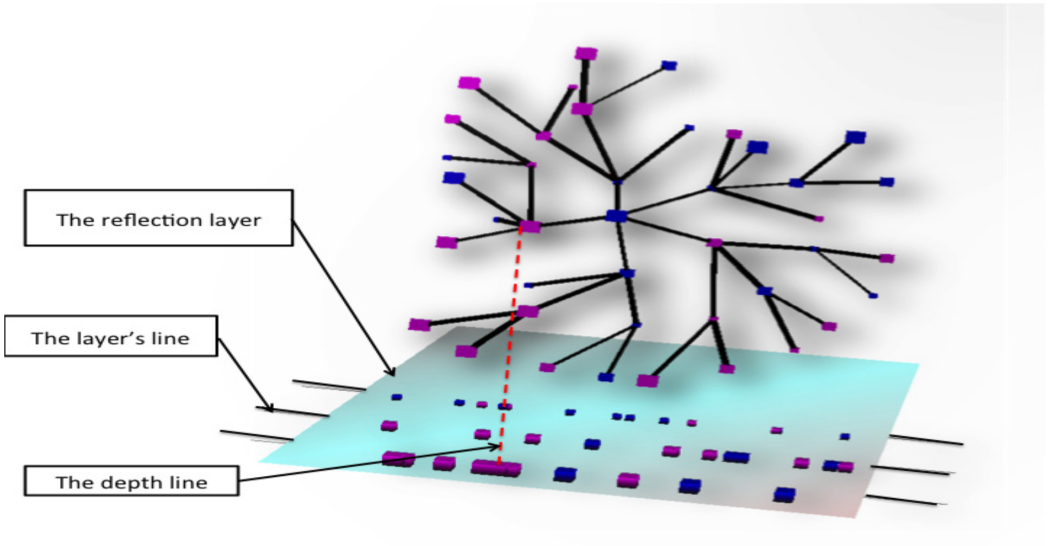


FIGURE 6.7: *The Reflection Layer Extension to the Stereoscopic Highlight Technique.*

Robertson et. al. [92] suggested some approaches of enhancing the depth cues for the users, e.g.:

- increase or decrease the size of the nodes, effectively exaggerate the perspective view
- use lighting to make closer objects appear lighter

- using shadows, basically projecting the z-plane onto the x or y-plane

We use a similar approach. Instead of projecting shadows, we add a reflecting plane under the visualization of the graph (see Figure 6.7). The benefit of using reflection over shadows is to keep the color of the reflected objects. This helps recognizing which reflection originates to which element in the graph. Additionally dotted lines (depth lines) are drawn from each graph element to their reflection counterpart on-demand.

Since we only use a small number of distinct depth layers, we can also mark these layers on the reflection as lines. We call this the layer's line.

The effect of this method was evaluated. The results are presented in the next chapter.

6.6 Conclusion

In this chapter we presented the scalable application developed as part of the thesis' goals. Being more than a proof-of-concept, it is a safety visualization developed to tackle the issues of safety experts when exploring large CFTs. Additionally it leverages the collaboration between the safety experts and hardware engineers when improving the safety and reliability of embedded systems. To have scalable and uniform interaction with the application, the Marking Menu approach as described in chapter 5 was used and extended to allow independent data examination, similar to the results we presented in chapter 3. We employed stereoscopic highlighting with the reflection layer extension to show the importance of safety model elements in a node-link diagram in a intuitive form.

The application can be used with little modification in other collaboration scenarios, where experts of different domains have to look at the same datasets from their individual points of view. The views we used for the safety and hardware angle of view can be replaced by almost any other connected views of data.

Chapter 7

Evaluation

This chapter starts with the evaluation results from the Stereoscopic Highlighting and Reflection Layer Approaches implemented in the safety visualization presented in the previous chapter. Then a case-study about enhancing 3D CAD models with metadata to improve interaction experience is presented.

Virtual Buttons are created by imposing an invisible regular grid onto the touchscreen of an input device. While users cannot see the grid, they can estimate the button positions and press them in an eyes-free manner. In this chapter we present an evaluation of device and user properties on the accuracy and speed of this method. The evaluation includes little children as users to provide a larger magnitude of user properties.

Finally a real-world application that is currently under development for a German car manufacturer is outlined. This shows the popular demand for large display interaction and underlines the viability of this work.

7.1 Stereoscopic Highlighting

Stereoscopic Highlighting, as introduced by Alper et. al. [6] uses the depth perception available with 3D visualization to highlight objects in a 2D graph over others. Expanding this technique, we want to find the limitations.

We implemented a prototype (using C++ and the Vrui framework [65]) to evaluate the accuracy of stereoscopic highlighting. The main goal is to measure the number of different depth layers users are able to tell from another. Additionally the influence of other visual attributes, such as color and shape

to the test outcome is interesting, as well as the user's ability to perceive that extra information conveyed by other attributes.

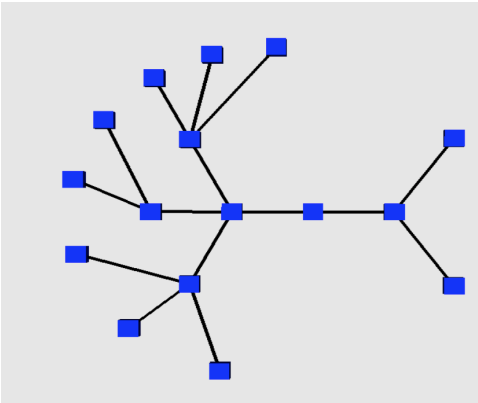


FIGURE 7.1: *2D Sample configuration for uniform configuration*

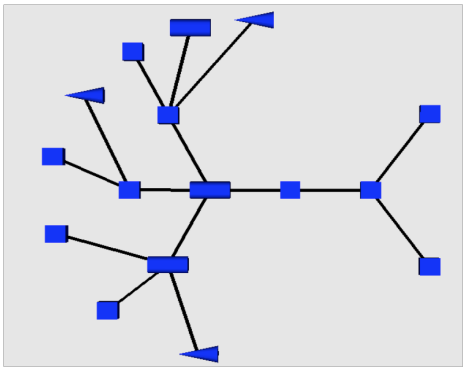


FIGURE 7.2: *2D Sample configuration with different shapes*

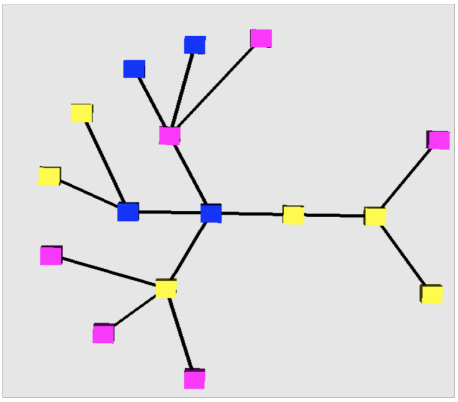


FIGURE 7.3: *2D Sample configuration with same shape and different colors*

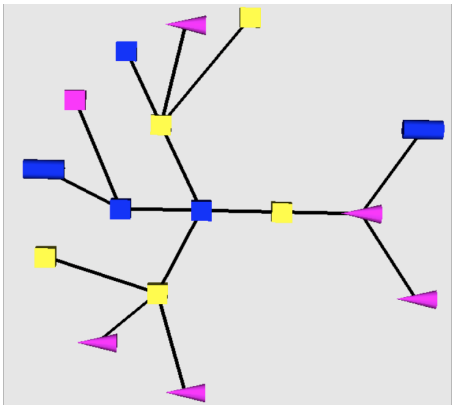


FIGURE 7.4: *2D Sample configuration with different shapes and different colors*

Seven different configurations were presented to each test candidate in random order. This was done once with a traditional 2D node-link-diagram (as for example in Figure 7.1). Then the same configurations (in different order) were presented using depth as highlighting clue. Despite being a 3D visualization, users had a fixed viewpoint and were not able to rotate the view. Later we added color and/or shape to encode additional data (see Figures 7.2, 7.3, 7.4).

7.1.1 Experiment Setup

The Evaluation was done on a 60cm 16:9 Zalman stereoscopic 3D display using polarized glasses. We used a resolution of 1920x1080 pixels at a refresh rate of 56 – 75 Hz. The display had a low response time of 5 ms. Test candidates were seated right in front of the display. The spacial parameters were adjusted for each user individually to guarantee maximal stereoscopic perception.

Our hypothesis for the study are [8]:

- **H1:**
  for all configurations in which the nodes representing components that have the same colors and shapes, three-layer configurations will be the most readable configurations in compared to the other experiment configurations.
- **H2:**
  we expect that combining more than one visual cue with the depth cue will not affect on the users' abilities in achieving the required task from the final layout.
- **H3:**
  users get familiar with the variations in depth values by the experiment time (which is 30 min in our experiment). Therefore, the accuracy of the depth detection will be increased by the passage of time.

Due to the background of this technique (as presented in chapter 6), we decided to have three distinct groups of test candidates:

- **Safety Experts:**
  Users with a background in safety and reliability engineering
- **Visualization Experts:**
  Users knowledgable in visualization techniques. They are expected to give more in-depth feedback on the approach
- **Non-Experts:**
  The catch-all group of people with neither safety nor visualization background.

In total we had 20 participants (6 female, 14 male), with 7 being safety experts, 7 visualization experts and 6 non-experts. The age range was 23 to 60 years

with a mean of 32 years.

Basis for the evaluation was for all users to count the number of depth layers they perceive and the number of graph elements on each layer.

First all users were given configuration 1, 2, 3, 5 and 7 in random order. In this run the node-link-diagram consisted only of interconnected blue cubes. This was done to simply measure the accuracy of users to correctly perceive the depth of the cubes.

In the next run, shapes and colors of the graph elements were different. Only configurations 3 and 7 were used (in fixed order). Here test candidates were not only asked for the depth information, but also to count the number of different shapes and/or colors. The actual goal of the second run was to see if depth perception improves over time.

The final run with three configurations (4, 6 and 8) again colors and shapes of the nodes were different. The order of the configurations were randomized. Testers had to count all different visual attributes. Here we aimed to find out if the additional visual clues affect depth perception or vice versa.

In all cases, skipping a task was possible, if users were confused with the visualization. The configurations had 2–5 depth layers.

The maximum time given for each participant was 30 minutes. At the end of the experiment, participants were given a closed-ended and an open-ended questionnaire to get their feedback. The open-ended questions consists of 12 questions to be answered on a likert scale, with the additional option of skipping a question.

7.1.2 Results and Discussion

Two layers were correctly detected by all participants. For three, four and five layers, the results can be seen in Figures 7.5 and 7.6. Three layers still have a high detecion-rate, but for more layers accuracy drops to slightly above and below 80% respectively. In total the accuracy in the first run was 86%.

Average time also drastically increased when more than three layers of depth were presented. Total average in run 1 was 48.57 seconds.

All but one user liked configurations with three layers of depth most, since they could still easily perceive the depth differences, but had more informations than in the configurations with only two layers.

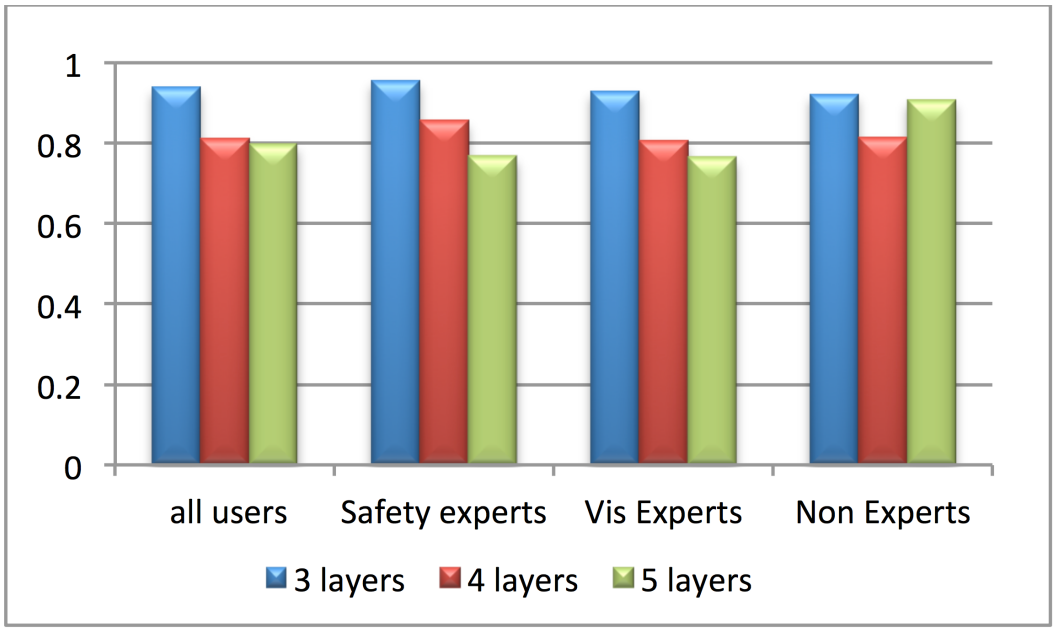


FIGURE 7.5: *Average accuracy for all configurations of run 1.*

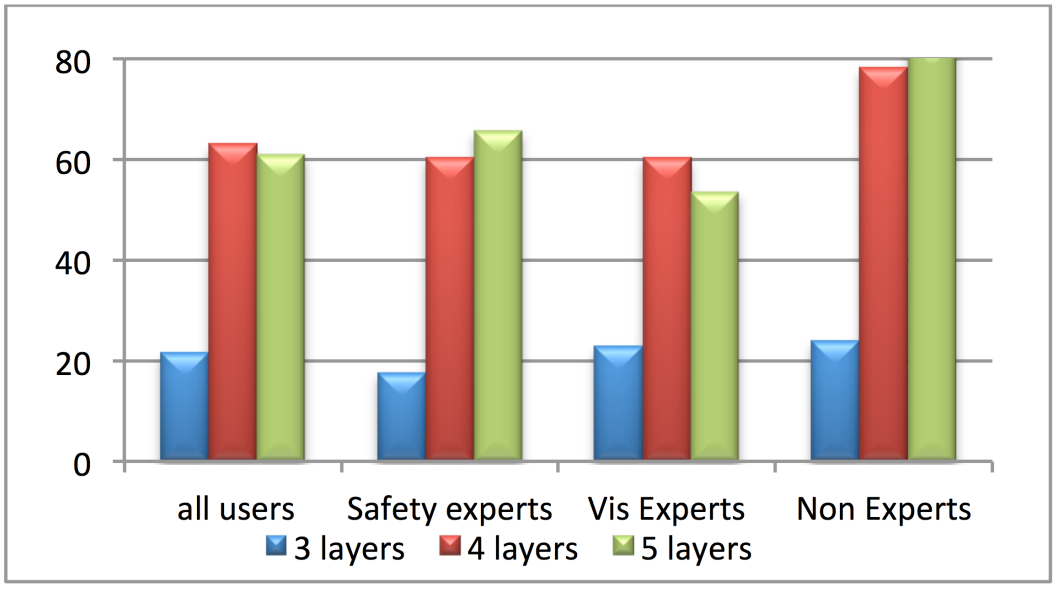


FIGURE 7.6: *Average time in seconds for all configurations of run 1.*

Results from the second run show that accuracy and speed improves over time. Both tested configuration have the same number of layers. Still the second configuration got better results consistently (see Figures 7.7 and 7.8).

In the final run we had one configuration (4) to measure the effect of adding color, one configuration (6) to measure the effect of adding different shapes

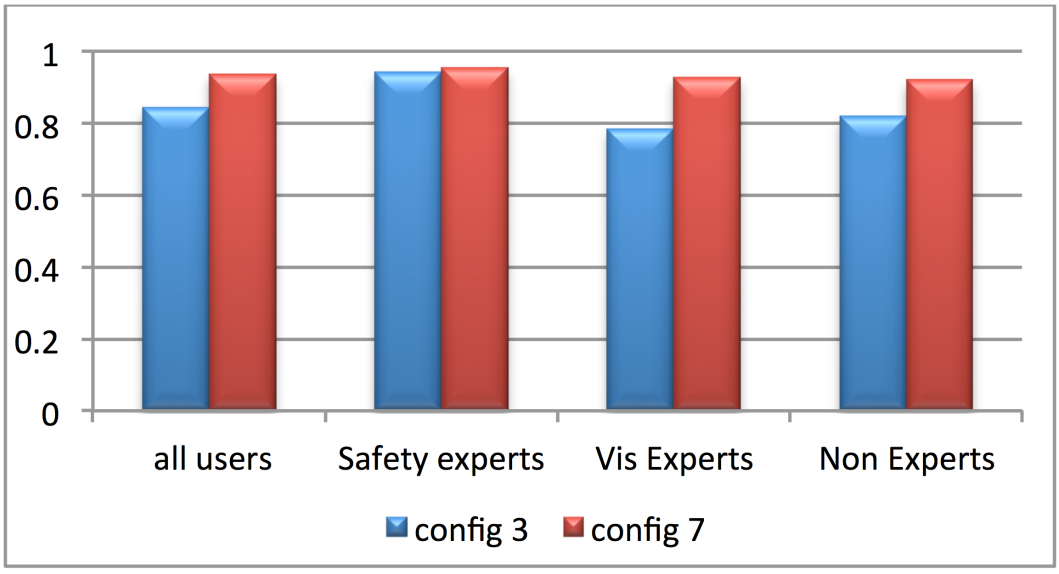


FIGURE 7.7: *Average accuracy for configurations 3 and 7 in run 2.*

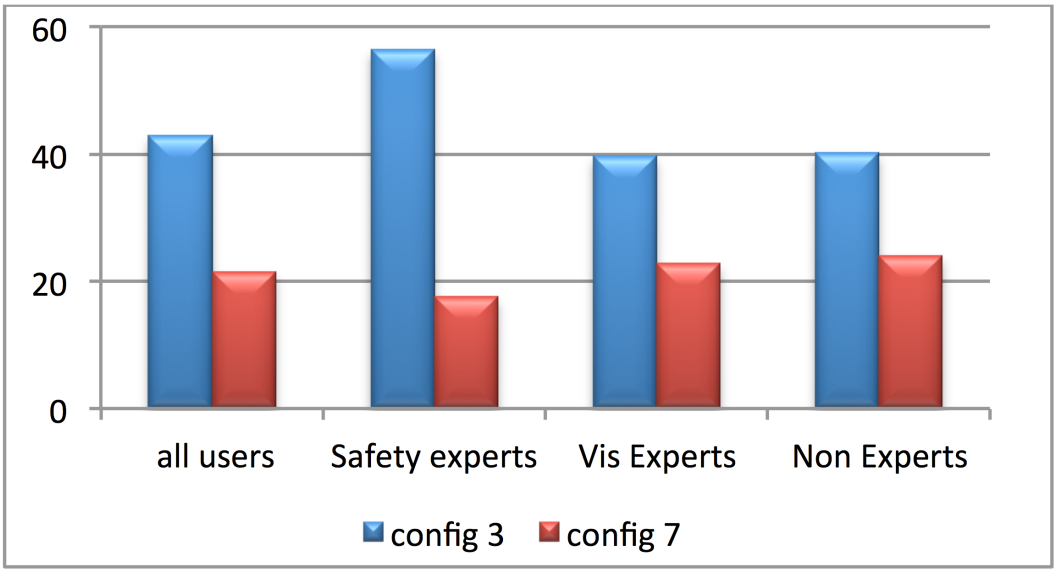


FIGURE 7.8: *Average time in seconds for configurations 3 and 7 in run 2.*

and a third (8) to measure the effect of adding both to the visualization. Figures 7.9 and 7.10 show the outcome. The effect of these additional visual clues, independent of their combination, does not impact accuracy of depth perception in a significant way.

Our results show, that users need some time to get used to the stereoscopic highlighting, maybe simply to get used to the stereoscopic display. The small standard deviation for accuracy of only 0.014 show that the performance of

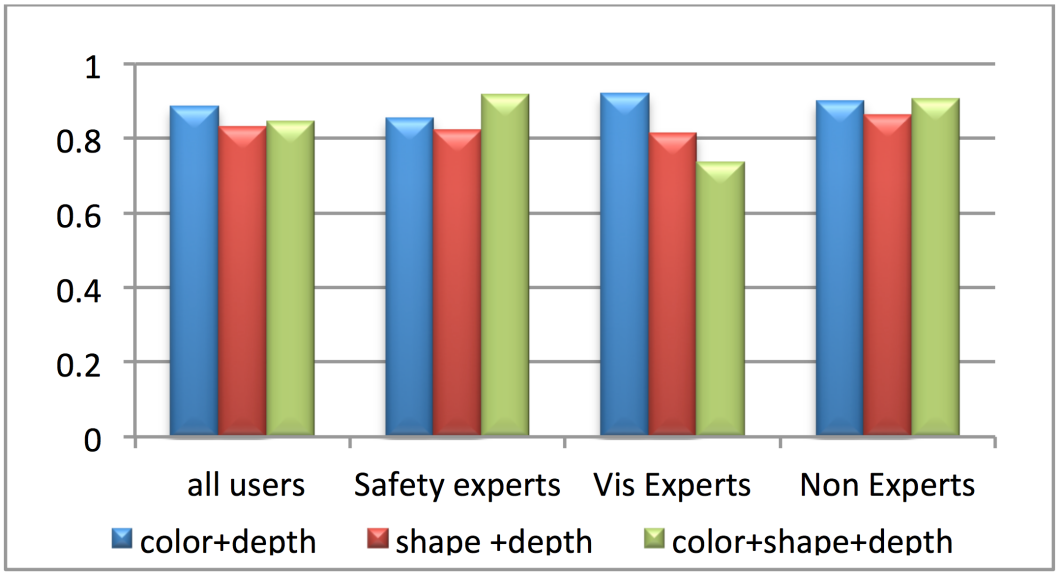


FIGURE 7.9: *Average accuracy for configurations 4 (color+depth), 6(shape+depth), and 8(color+shape+depth) in run 3 with combined visual cues.*

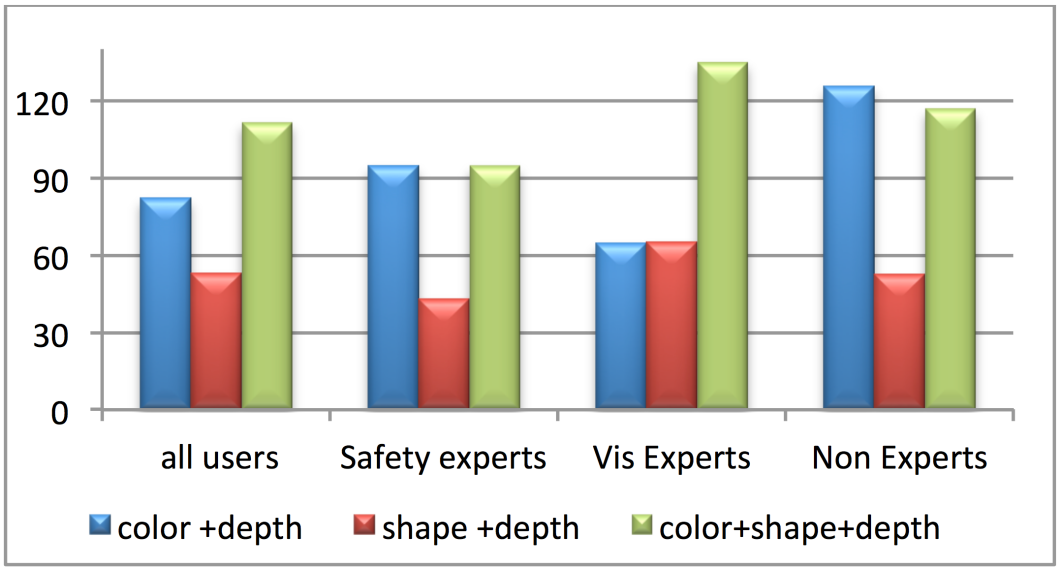


FIGURE 7.10: *Average time in seconds for configurations 4 (color+depth), 6(shape+depth), and 8(color+shape+depth) in run 3 with combined visual cues.*

all participants were not that different (amongst all user classes). A standard deviation of 14.3 seconds for all users show, that the speed of perception does indeed differ amongst users.

Overall we can conclude, that using three depth-layers is superior to any other configuration. Using fewer layers only transports less information, while

using four or more layers comes with a huge penalty in terms of accuracy and perception speed. Additionally using other visual clues does not significantly affect the depth perception, while generally decreasing the perception speed as the scene gets more complicated.

7.2 Reflection Layer Extension

In this section we will present the evaluation done for the extension of the stereoscopic highlighting in the form of a reflection layer. The whole approach is presented in preceding chapter.

7.2.1 Study Design

For the study we use two comparable settings. The first one will use the stereoscopic highlighting technique as presented above, the second setting additionally uses the reflection layer.

The goals in this study are similar to the one of the study about the stereoscopic highlighting. We want to measure the users ability of perceiving the depth with and without the help of the reflection layer and learning effects.

The hypothesis for the test are [9]:

- **H1:**
  The accuracy value is irrelevant to the data size but it requires longer time for users to read a configuration with big data size.
- **H2:**
  The accuracy of detecting the variation in depth values will be significantly increased by adding an extra layer, the reflection layer, in the bottom that reflects the current configuration in the bottom of the 3D world.

H1 directly comes from the results of the previous study, where this was true. H2 is designed after the idea of having the reflection layer improving the depth perception for all users.

The testing environment was similar to the one used in the previous study, again using the 60cm 16:9 Zalman stereoscopic 3D display using polarized glasses. We used a resolution of 1920x1080 pixels at a refresh rate of 56 – 75 Hz. The display had a low response time of 5 ms. Test candidates were

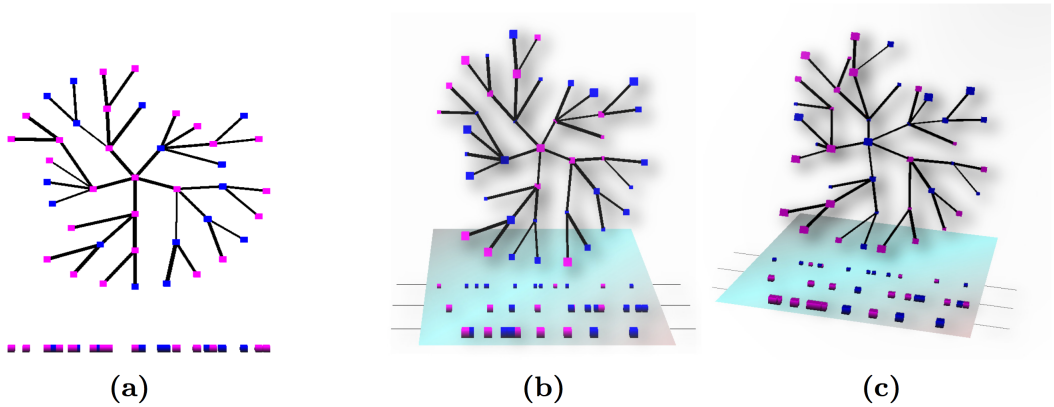


FIGURE 7.11: *An experiment's sample for the three layers configuration. In (a) we show the initial setting, in (b) we show the front view of the graph when the depth values being changed, and in (c) we show a side-view of the same configuration to visualize the 3D effect.[9]*

seated right in front of the display. The spacial parameters were adjusted for each user individually to guarantee maximal stereoscopic perception. We use a modified version of the same software used thin the last study.

This time we only had two different user groups, one being the visualization experts, and the non-experts. They evaluated the proposed solution in a controlled environment through a task-based evaluation tests [43]. After the test they were again given an open-ended (13 questions on a likert-scale) and a closed-ended questionnaire. The user groups were each 4 people in size. Of the 8 participants, 2 were female, 6 were male. The age range was 24 – 34 year, the mean age was 28.5. Half of each group already participated in the previous study about stereoscopic highlighting.

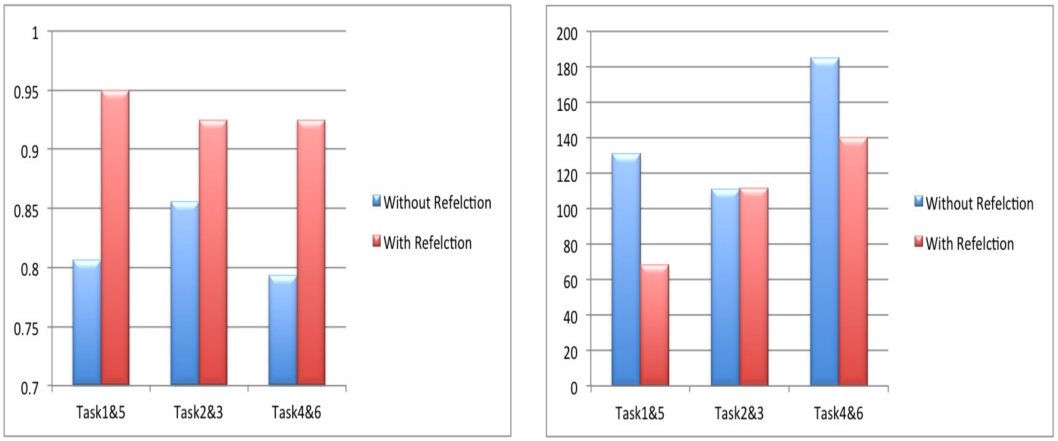
7.2.2 Graphs Used in the Experiment

We used node-link diagrams again, similar to the ones used in the stereoscopic highlighting study. Figure 7.11a shows the initial configuration without any depth. Figures 7.11b and 7.11c show the same graph with depth and reflection layer enabled.

For each available configuration the reflection layer was enabled in one case and disabled in another. The sequence of configurations was the same for all users, but difficulty of the configurations varied. The complete setup can be

TABLE 7.1: *The experiment tasks' configurations.*

Task ID	No. of Layers	Using Reflection
Task 1	3 layers	✗
Task 2	4 layers	✓
Task 3	4 layers	✗
Task 4	5 layers	✗
Task 5	3 layers	✓
Task 6	5 layers	✓



(a) **(b)**

FIGURE 7.12: *(a) Average accuracy of all tasks for all user groups. (b) The average time of all tasks for all users groups.*

found in Figure 7.1. Participants had the options of skipping a task. The maximum time allowed for the whole test was 45 minutes for each candidate.

7.2.3 Results

The results of the evaluation can be seen in Figure 7.12. We can clearly see how the reflection layer positively impacts the accuracy of depth perception. In all cases accuracy is more than 90%, while it is 85% and below without the reflection layer. Standard deviation as 0.018 for the reflection layer and 0.007 for the cases without reflection. This shows results quite similar across all users. Also the graph contained 41 nodes as compared to the 16 nodes in

the previous study. Since the accuracy of the non-reflection-layer cases does not differ significantly from the results of that previous study, we conclude, that in graph size does not impact the depth perception.

Looking at the average times, we can see that the reflection layer does not negatively impact the perception speed. In two of the three comparisons, using the reflection layer actually makes users solve the task quicker. The standard deviation sank from 16.46 to 4.38 when using the reflection layer. In the questionnaire, users stated that using the reflection layer helps to understand the depths of the graph elements, especially when they are unsure about it. All participants preferred to have the reflection layer, even when 2 of them did not state, that it helped them.

Overall the test supports our hypothesis. Using the reflection layer increases not only the accuracy and possibly the speed of depth perception, but also user acceptance and joy of use. Additionally we conclude that the size of a graph does not have a direct impact on the depth perception of users, at least not in reasonable sizes.

7.3 Case Study - 3D Model Interaction

Most datasets in use today are very large, too large to simply display them. Besides sophisticated visualization techniques the user must be able to interact with the visualization. Many of these interaction techniques are widely used and understood, for example as shown in the work of Yi et al. [109]. Guidelines for interactive information visualization from Brath [32] were written almost 15 years ago. While the latter contains only a summary of guidelines, newer work includes a far higher abstraction level. This of course allows to apply the strategies and methods to a broader application field. Unfortunately there is little work done on how to design an interaction on a more concrete level. This is the main focus for the following paper. Featuring a small case study, problems with employing well-known interactions in different cases are shown and conclusions on how to avoid those problems in future work are drawn.

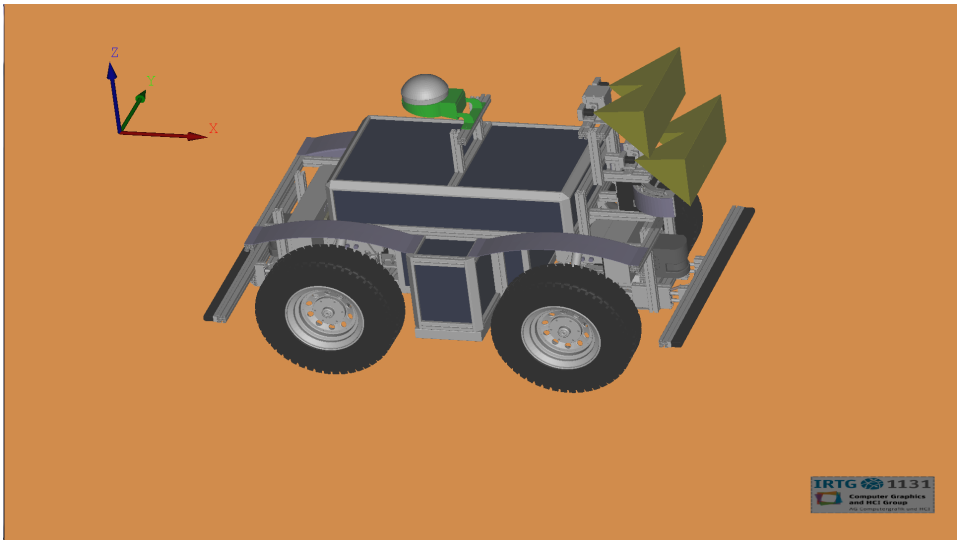


FIGURE 7.13: *Model View showing the whole robot*

7.3.1 CAD Modelviewer

For safety and security analysis of embedded systems, a special visualization environment was created for the VierForES-Project (Figure 1). The visualization shows critical components and combinations of components resulting from a Fault Tree Analysis (FTA) as provided by an external tool [60]. All combinations of components leading to a specific system failure are depicted in the system along with their probabilities. Details about this system can be found in [4], [5] and [62]. Most important for this paper is the *model view*, a 3D CAD model viewer, that displays the system under examination (Figure 1, rightmost view). It can highlight critical parts of the model by showing other parts translucent (as seen in Fig. 1).

Original Model

The mobile robot *RAVON* (Robust Autonomous Vehicle for Off-road Navigation) was the first real system our tool was applied to. For that reason, interactions with the *model view* were tailored towards *RAVON*. General interactions with CAD models are well-known, with rotation, translation, and zoom being the most important ones. These basic actions are needed to allow the user to view any part of the displayed model from any angle. They can be initiated by using a computer mouse, the keyboard, or a 3D mouse. The latter

was chosen for its high popularity with CAD designers, while the traditional mouse and keyboard were obvious choices since about every personal computer or laptop is delivered with those or a variant of it.

Mapping those interactions to the capabilities of the individual input devices is an important task, especially when it comes to 3D interaction with a normal computer mouse, which only supports two degrees-of-freedom (DOF) natively as opposed to the full 6 DOF needed. While numerous attempts have been made to have computer mice with more DOF, like the two-ball mouse[74], none of them really made it into the mainstream market. For standard mice the common solution to this problem is to map x-y-translation to one mouse button and rotation around the x and y axes to another one. Zooming or z-translation is provided by the mouse wheel common on todays computer mice. This actually only results in 5 DOF, but this is sufficient to fulfill the requirement of enabling the user to view any part of the 3D model from any angle as rotation around a third axis can be substituted by rotation about the other two axes. This solution was also applied to the model view. It is feasible to assume rotation to be more important than translation as users will want to see the position of components of interest, especially in their context. Additionally the model initially fits on the screen, so no translation is necessary at all, as long as no zooming is performed before. Therefore the primary (left) mouse button was chosen for rotation. A secondary click (i.e., with the right mouse button on most systems) triggers x-y-translation. Additional mouse or keyboard buttons to gain more DOF were not used to keep the interaction pattern simple.

Zooming as the only action remaining is mapped to the mouse wheel, which is very common in most computer applications nowadays. It is not implemented as scaling of the model coordinates, but translating the viewport along the z-axis. This has little effect besides the user having the ability of moving "through" the model (which is prohibited by the *model view*), but this is an important difference regarding the second model introduced in the next section of this paper and uses the same approach as the 3D mouse.

The 3D mouse already has its own well-defined set of interaction mappings, used in any 3D application supporting the mouse, so it was an obvious choice to implement this in *model view* as well. Pushing the hat of the mouse in

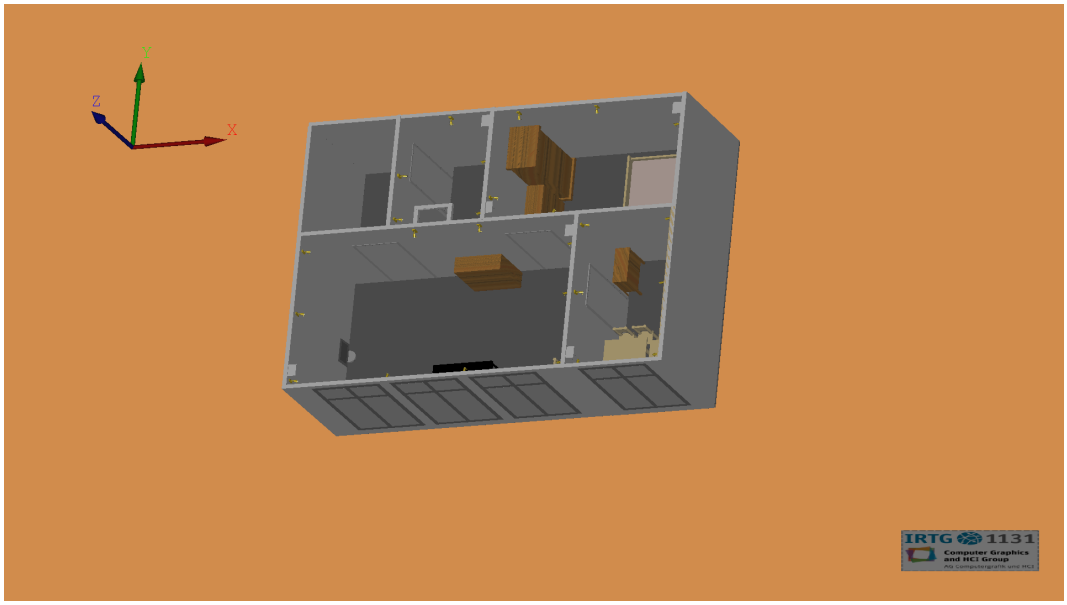


FIGURE 7.14: *Model View showing the AAL-Lab*

any direction (including up/down) translates the model in the appropriate direction, tilting the hat in a direction rotates, and spinning rotates around the third axis not covered by tilting.

The keyboard provides more than enough keys to have an own key for each movement and rotation direction. To keep things simple, translation in x-y-directions is mapped to the arrow keys, zooming (or z-translation) to two other keys, while rotation around an axis is achieved via one key for each rotation axis (e.g., x for the x-axis) and by using a modifier key the direction can be reversed. This keeps keyboard interaction simple. Additional functionality of the system can be activated via other keys, including an auto-rotation animation that allows the user to see the model from all sides without having to push a button all the time.

New Model

The second real system to be analyzed using our system was an ambient assisted living laboratory (AAL, for more information about AAL see [38]). The laboratory was visualized in the *model view* highlighting failing sensors. A number of common problems had to be solved in order to import the CAD model into the visualization system, like model-scaling, but those issues are

already well-known in computer graphics.

Interacting with the AAL model highlighted new issues. The lab is shown from a bird's eye view (Figure 3). When using the computer mouse, the primary mouse button triggers rotation around x and y axes. This causes the model to be seen from the lower side with the floor obstructing the view into the laboratory itself. Even when only rotating a small amount, the walls tend to hinder the view on the rooms. The secondary mouse action is more usable. Translation does not obstruct the line of sight and makes it possible to center the view on the room of interest. It becomes even more important when the user is also employing zoom, for example when zooming in on a certain room. Since zooming in the *model view* is actually z-translation, the rotation becomes a viable interaction here. If zooming was implemented as actually enlarging the scene's components the camera position would still be in the original position. Rotation would then still use this original position as an anchor, resulting in the model rotating out of the user's view. After moving the viewport into the laboratory, rotation offers the possibility of seeing the room as if standing in it from a 1st person perspective this way. Unfortunately, moving through the lab is only supported by using the normal translation functionality. Obstacles, like walls, blocking the user's way is unsupported. This is a good approach for the robot, where an inside view is a viable option, but has obvious shortcomings when exploring the AAL laboratory (This is actually part of the VRML'97-standard, as explained more in detail in the following section).

When controlling the view with the keyboard, a user experiences no big differences between both models, as mentioned above, the rotation functionality is limited, but on the keyboard, it's easy to avoid the rotational keys at all. The auto-rotation is also not applicable for the lab model, because of the rotation issues already mentioned. Rotation of the model around the z-axis would also not solve the issue. The occlusion problem may be solved this way, but still there is no real advantage in rotating the AAL model.

The 3D mouse still has its set of interactions, also valid for the lab model. But especially for the novice user unintentional rotation when tilting the mouse instead of pushing can negatively impact the user experience.

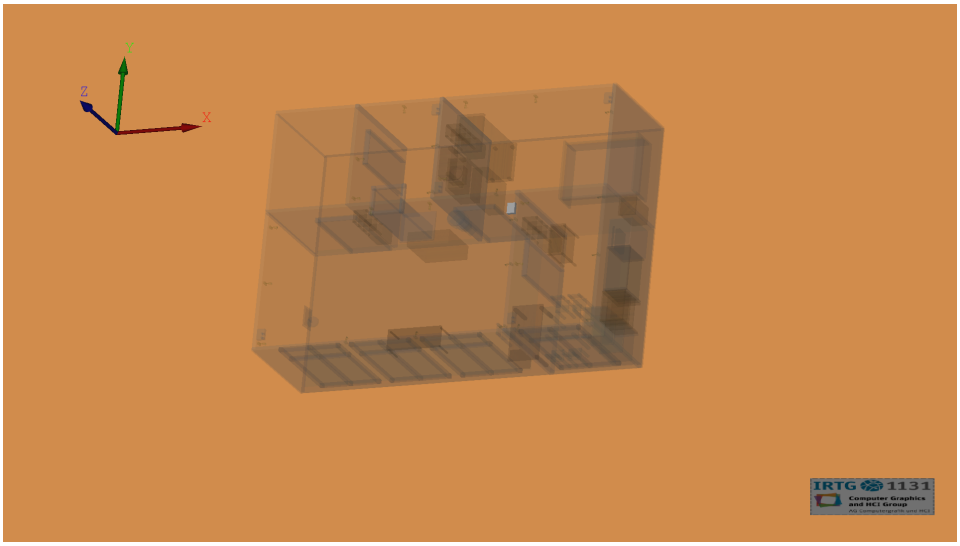


FIGURE 7.15: *Model View showing the AAL-Lab, using transparency to highlight a small sensor*

7.3.2 User Study

The above results were verified in an informal user study. Four users with different levels of computer experience were given the task to examine the models in detail. Afterwards their opinion about the model interaction was asked. The general statement was that the system is usable in the current state, but could be improved, especially the interaction with the laboratory.

7.3.3 Discussion and Conclusion

From the case evaluated, it becomes obvious that even well-known interaction metaphors are not sufficient to provide maximum user satisfaction in all cases. When visualizing data (e.g., 3D models), which may be provided by varying sources (or fields of study), it is feasible to add interaction hints to the data. These interaction hints might be as simple as for example the navigation types specified by the VRML'97 standard[1]. Here four different interaction mechanisms are defined: *WALK*, *FLY*, *EXAMINE*, *NONE*. *NONE* is simply the absence of any interaction, which is not of interest here. The *EXAMINE* mode works generally like the current implementation of the *model view*, with emphasis on rotation and zoom. With the *WALK* metaphor interiors, like the AAL laboratory can be inspected in a very natural way, as proposed in the

previous subchapter. The viewport is moved along the model surface utilizing gravity along a model-specified z-axis and collision detection. The last mode, *FLY* only differs from *WALK* by ignoring gravity.

Interaction hints can also be designed in a more general way. Input can be abstracted from the actual input device as shown by He and Kaufman in [56]. They define categories of input data (position in both 2D and 3D, device rotation and selection as a catch-all category for buttons and alike) in which input from all devices can be classified. Using this model the basic actions of input devices can be directly mapped to properties of CAD models. This also offers possibilities of assigning more than the usual six or seven DOF (helpful for example when the portrayed model contains movable parts that hide other parts). This can also be exploited to add more interaction to a viewer. The downside of this approach is that every model needs its own interaction design. Most designers of CAD models are little to not trained in creating interactions. This imposes a lot of extra work to each new model created, possibly with help from an interaction designer.

In this very example a simple working solution is to differentiate between outer and inner view scenarios. *RAVON* is an outer view scenario, where rotation and to some extend, zooming plays a major role. It is natural for the user to view a robot from multiple angles, as one would do in reality, by just walking around the robot. In contrast for inner view scenarios, a bird's eye view is the best to get an overview of the whole system and the ability to roam through the model makes exploring parts of the model more natural. Having predefined and/or user-defined spots where the viewport can be set to with a simple command will also be helpful for those kinds of use cases.

Two major issues exist with this solution: Users working with both kinds of model will have to adapt to two different usage layouts. The extend to which this has to be done depends on the interaction device. Users of computer mice are facing the biggest change, while a 3D mouse uses almost the same interaction pattern in both cases. The second issue arises when dealing with large models having both interesting interior and exterior, as for example an airplane. Letting the user choose the current interaction mode may help to offset this problem, but introduces new complexity in the interaction. Having both modes active all the time is possible, but need non-standard interaction,

as both modes utilize the mouse in different ways. Having two different (distinguishable, e.g., by color coding) mice could solve this problem (as could be done by other more specialized input devices). But the mouse/keyboard combination was initially chosen for interaction because of its commonness and unfortunately having two computer mice is still not a common sight, despite their low price.

Generally, this case study shows that it is not a simple task to create interaction patterns that produce maximum user satisfaction. Especially, when the visualization is not tailored towards a special use case, as many cases as possible should be taken into account when designing the interaction. While a step-by-step approach (i.e. creating interaction for a first scenario, then adapting for the next one) is possible, people used to the first input methods are (probably unnecessarily) faced with changes. This may even have impact on their working efficiency and the ease of use, at least for some introduction time.

Designing a domain-specific language that defines certain modalities of a model might be a good starting point for intelligent user interfaces to determine the best interaction pattern for not only a single model, but also for different views of the same model. The distinction between inside and outside models might be suitable for this case study, but is certainly to simple for general cases as described above in the airplane example. With a domain-specific language issues like that can be addressed, and interior and exterior areas might be distinguishable for the viewer. Of course more complex properties can be used, for example to provide transparency or highlighting of some special parts of the model under given circumstances. If the domain-specific language is easy to learn, 3D CAD model designers can provide interaction descriptions easily without having to learn about interaction design first.

7.4 Virtual Buttons

In most cases, interaction with mobile phones and tablets is done via the touchscreen of those devices. Users will naturally look at the screen while using it. However, there are some cases, where looking at the screen might not be possible or viable, e.g. while driving or when interacting with another system

via the mobile device. This is called eyes-free interaction. A comprehensive list of such cases is presented by Yi et.al. [108]. Consumer-level devices do not provide haptic feedback on their screens, thus making this style of interaction difficult. One possibility of dealing with this is to divide the screen into a grid of virtual buttons. Other common approaches are described in the Related Work section later. There are studies showing the impact of different grid sizes on the accuracy (i.e., the hit-miss-ration of the user) for this kind of interaction. The paper from Wang et.al. [104] is a prime example of this. More are covered later in the Related Work. No research has yet been done on the impact of different devices and their sizes on the accuracy and the duration of actually interacting, since the device size impacts those values inversely.

In this paper, we describe an evaluation of eyes-free virtual button presses. Unlike previous studies, we look for the impact of not only the number of virtual buttons on the touchscreen, but also the size of the device itself, attributes of the user, etc. To get further insight into this, our participants are not only adults, but also small children. Children are getting their own mobile phones (with touchscreens) in early age, thus it is important to know, if methods working well for adults are actually viable for them.

In the next section we will give a brief overview of related literature. Then we will present the design of our evaluation and the process, followed by the actual evaluation of the results. We finish with the conclusions of the results.

7.4.1 Related Work

Yi et.al. [108] presented a classification of motives for eyes-free interaction and identified several actual reasons. The four main categories are *Environmental*, *Social*, *Device Features* and *Personal*. The number and variety of reasons shows that eyes-free interaction is an important topic, that should not be forgotten about.

Alternatives for eyes-free interaction to the virtual button approach include Bezel Swipe [94] and Touch Gestures [30]. Bezel Swipe requires the user to touch one of the sides of the screen and swipe inwards. The tactile feedback with this technique makes it viable for eyes-free interaction. Touch gestures, that do not require any special position on the screen, such as a simple pinch gesture are also easy to use eyes-free.

Azenkot and Zhai [11] evaluated the effect of using a finger, a thumb or two thumbs on text input on touch capable smart phones.

An evaluation about virtual buttons on smart phones was done by Wang et al. [104]. They found that grid sizes of bigger than 3x3 is not feasible to use for almost any user. Their evaluation was only done with one touchscreen capable phone and only using one handed-interaction.

A huge evaluation done by Henze et al. [57] focuses on the impact of several hardware design factors, such as device size, screen resolution, target sizes and position. They gathered data through a game distributed in an official app store. However their study was by design not abled to measure those effects on eyes-free interaction.

7.4.2 Evaluation

Design

The main goal of the study was to find out how different factors impact on the eyes-free usability of virtual buttons. Those factors are the size of the virtual buttons, as given by the touchscreen size and the layout of the button grid. According to the results of Wang et al. [104], button grids with more than 3 buttons in any direction were excluded, as were trivial layouts with only one or two buttons in total. The remaining grid layouts were : 1x3, 2x3, 3x3, 3x2 and 3x1. To gain insight about the effect of device size, we also used three different devices, an iPad 2 (241.2mm x 185.7mm, 9.7 inch display), iPad Mini (200mm x 134.7mm, 7.9 inch display) and an iPhone 4S (115.2mm x 58.6mm, 3.5 inch display). We chose to use only this device family to minimise hardware influences on the tests. For example the resolution of touch input (not screen pixels) is similar on all devices. This assures the App to behave the same on all devices, too. Since the device orientation might impact user performance as well, all tests were to be done in both landscape and portrait orientation. As shown by Azenkot and Zhai [11], usage of thumb vs. finger can have an impact, too, and according to Karlson et al. [61] this is the primary way of mobile touch interaction. Therefore each user was asked to complete all tests a second time using only one thumb in portrait orientation and two thumbs in landscape. In the other test runs, user had the freedom of choice of how to

interact with the device, the only constraint being, the device has to be held with at least one hand and may not be put down on a table or anywhere else. Because of the physical size of the devices, this was only done for the phone. For additional data, users were asked for their experience with touch devices (self perceived), age and gender.

The evaluation design is straightforward. Every user is presented a grid of virtual buttons, displayed not on the mobile device, but on a separate computer screen, as we focus on eyes-free interaction. The mobile device screen stays completely blank, so if a user accidentally looks at the mobile device, they get no additional insight, and thus no advantage over other participants not looking at the device.

Depending on the complexity of the grid, 20, 30, or 40 touches were needed per grid layout and device orientation (more complex layouts needing more touches). Considering that every participant was asked to perform the test with three different devices and one device having an additional test run under special conditions de-scribed later, every user had to perform a total of 1520 touches. We used three different mobile device sizes, large tablet, small tablet and mobile phone to measure the impact of the device/screen size on the test.

To have a wider age range in the evaluation, we decided to not only have adult participants. To see how age actually affects the results of our study, we created another design, targeted at small children. This will help to understand if virtual buttons is a concept simple enough for even children to understand. We can also get an insight on the effect the grid layout and device size has on their performance, given their smaller hand sizes.

To get results across all possible grid setups, we reduced the number of touches needed per grid layout to ten. Also to help the children understand the task they were supposed to accomplish, their test started with a short example of the test with the grid visible on both the mobile device and the computer screen. These touches were, of course, not counted in the evaluation. The Thumb-Only run on the mobile phone was also not done by the children, since their hands were not big enough to accomplish it. To not negatively influence the study due to the children getting bored by the tasks, they had the option of stopping the test after each device. Thus, less data is available from the children's test, but still enough for an evaluation.

7.4.3 Participants

For the test involving adults, we had 44 participants (30 of them male, 14 female) in the age range of 21 to 62 years. All test candidates were asked to rate their own experience with touch screen interaction (on a 1-5 Likert-scale) and to state if they own a touch device themselves (which the majority of 34 users did). The average experience level was 3.57, unsurprisingly due to the high number of touch device owners in the test. As a side note, the high number of touch device owners shows the ubiquity of these devices and does not affect the viability of the test as we still have 10 participants not owning such a device.

The 29 children (18 male, 11 female) were between 4 and 6 years old. They were not asked to rate their experience, as children of that age are unreliable of their self-assessment. But they were asked, if anybody in their family owns a touch device that they are allowed to use on a regular basis. 18 of the 29 children confirmed that, while 9 had no prior access to touch devices (2 children opted to not to answer that question).

Implementation and Results

For the evaluation we developed a simple mobile application connected to a desk-top application. The desktop application will display the current grid of virtual buttons and highlight the button the user is supposed to be pressing. To avoid confusion, the system guarantees that no button will be highlighted twice in a row, giving reliable feedback to the participants. The system automatically randomizes the test order (or-der of device, grid layout and button to press) to avoid learning effects affecting the results. The grid was displayed on a standard LCD computer display with 21 inches of size, connected to a desktop PC. A dedicated WiFi connected the PC and the mobile device.

Each touch was recorded individually. Accuracy and duration was measured by aggregating the gathered data. Accuracy is simply the ration of hitting the correct button and the total number of button requests. Duration is the total time between a button prompt being displayed and it being pressed, summed up for each test run.

To get basic results from the study, we tested for influence of four ba-

TABLE 7.2: *Mean values of accuracy and average duration per touch depending on different categories.*

Classification	Accuracy		Duration	
	Adult	Children	Adult	Children
Orientation				
Portrait	0.9638	0.8222	0.7840	3.008
Landscape	0.9538	0.7928	0.8534	2.972
Device/Thumb Mode				
iPad 2	0.9736	0.8571	0.7689	1.629
iPad Mini	0.9687	0.8115	0.7736	1.434
iPhone 4S	0.930	0.7566	0.8863	5.887
iPhone 4S Thumb(s)	0.9500	N/A	0.8440	N/A
Owns Touch Device				
No	0.9515	0.8595	0.9207	6.175
Yes	0.9631	0.7756	0.7947	1.579
Experience				
1	0.9203	N/A	0.9637	N/A
2	0.9391	N/A	0.7848	N/A
3	0.9717	N/A	0.8681	N/A
4	0.9798	N/A	0.8277	N/A
5	0.9695	N/A	0.7405	N/A
Gender				
Male	0.9557	0.7637	0.7640	3.967
Female	0.9597	0.8948	0.9341	1.648

sic factors on the accuracy. The factors are the device orientation (Landscape/Portrait), a combined factor of the device used and if the user was asked to use only their thumb, if the user actually owns a touch device, and the self-rated experience value.

The mean values are given in Table 7.2. As supposed from the related work, the accuracy results are above 90% for adults. Children's accuracy is significantly lower at values between 70% and up to more than 90%. This shows, that children are actually able to understand the concept of virtual buttons, even for eyes-free interaction. But due to the lower hit-ratio, care must be taken to allow undo-operations or similar mechanisms when designing applications for children.

Portrait orientation supports higher accuracy for both adults and children. It also influences the duration, with adults being faster in portrait orientation. Interestingly children were faster using the (less accurate) landscape orientation. More evaluation is needed in order to gain insight into this.

As expected, bigger devices allow for higher accuracy. It also seems, that the bigger device size does not cause the duration to go up (due to longer ways to move fingers), but instead seem to lower the duration. This is probably caused by users being more confident in hitting the desired area.

Using the thumbs only for interaction with the phone seems to actually increase accuracy and interaction speed. Since the alternative involves moving the whole hand to reposition the finger, this is not surprising.

Experience levels do also influence both accuracy and duration, mostly in the expected way. There is a outlier in the data, as the low experience level of 2 performed faster than levels 1, 3 and 4.

Gender does not seem to influence accuracy at all for the adults. Though male candidates performed the requested tasks quicker than females. The cause of this needs more evaluation. Even more interesting is the fact, that with children this is actually reversed. Girls were more accurate hitting the virtual buttons and even performed fasters.

Looking at the actual target button sizes, as defined by touch screen size and the grid layout yields the results seen in Table 7.3 and 7.4. For Table 7.3 the results were grouped by the minimal dimension of the button. For example, a button size of 10x15 mm would then have a minimal dimension

TABLE 7.3: *Accuracy and duration by minimal dimension of target button (i.e., a button size of 10x15mm has a minimal dimension of 10mm). Columns contain mean values and standard deviation. Size is given in milimeters and duration in seconds.*

min.	Accuracy		Duration	
Size	Adults	Children	Adults	Children
19.5	0.9349 ± 0.1103	0.7448 ± 0.2805	0.9191 ± 1.2987	5.647 ± 16.359
29.3	0.9569 ± 0.1009	0.8139 ± 0.2489	0.8223 ± 0.6596	6.238 ± 23.781
38.4	0.9464 ± 0.1071	0.7595 ± 0.2443	0.7452 ± 0.2164	5.681 ± 17.085
44.9	0.9645 ± 0.0907	0.8050 ± 0.2595	0.7843 ± 0.1672	1.524 ± 0.6258
61.9	0.9683 ± 0.0679	0.8464 ± 0.1685	0.7896 ± 0.2002	1.660 ± 0.6454
66.7	0.9665 ± 0.0749	0.7970 ± 0.2514	0.7764 ± 0.2366	1.344 ± 0.6166
67.4	0.9790 ± 0.1026	0.8735 ± 0.2013	0.7333 ± 0.1874	1.385 ± 0.6058
80.4	0.9692 ± 0.0666	0.8711 ± 0.1413	0.7591 ± 0.1591	1.678 ± 0.7999
92.9	0.9938 ± 0.0166	0.8965 ± 0.1803	0.7190 ± 0.1111	1.503 ± 0.5611

TABLE 7.4: *Accuracy and duration by size of the button. Columns contain mean values and standard deviation. Size is given in square millimeters and duration in seconds.*

Sq.	Accuracy		Duration	
Size	Adults	Children	Adults	Children
750	0.9223 ± 0.1048	0.7192 ± 0.3213	0.9038 ± 0.2660	5.8143 ± 17.620
1125	0.9397 ± 0.1203	0.7679 ± 0.2690	0.8764 ± 0.9219	6.4241 ± 24.571
1688	0.9777 ± 0.0805	0.8415 ± 0.2279	0.8240 ± 0.9643	5.6302 ± 16.959
2250	0.9538 ± 0.0918	0.7573 ± 0.2474	0.8348 ± 1.5992	5.5466 ± 16.877
2993	0.9611 ± 0.0781	0.7941 ± 0.2644	0.8260 ± 0.1703	1.6114 ± 0.5039
4490	0.9645 ± 0.0992	0.8012 ± 0.2779	0.7965 ± 0.2172	1.4944 ± 0.6702
4977	0.9625 ± 0.0836	0.8297 ± 0.1682	0.8210 ± 0.1668	1.7047 ± 0.4986
6735	0.9790 ± 0.1026	0.8735 ± 0.2013	0.7333 ± 0.1874	1.3853 ± 0.6058
7465	0.9700 ± 0.0625	0.8734 ± 0.1535	0.7905 ± 0.1977	1.8205 ± 0.8531
8980	0.9710 ± 0.0569	0.8058 ± 0.2277	0.7063 ± 0.1505	1.3203 ± 0.6133
11198	0.9938 ± 0.0166	0.8965 ± 0.1803	0.7190 ± 0.1111	1.5032 ± 0.5611
14930	0.9718 ± 0.0583	0.8527 ± 0.1573	0.7108 ± 0.1632	1.4917 ± 0.6016

Factor	Adult Group		Children Group	
Orientation	0.011	6.518	0.104	2.649
Device/Thumb Mode	0.000	14.208	0.000	10.437
Owns Touch Device	0.024	5.106	0.000	17.874
Experience	0.000	22.866	N/A	N/A

TABLE 7.5: *ANOVA Results for* Accuracy. *The first value in each cell is the significance value, the second is the F-value as calculated by the ANOVA method.*

Factor	Adult Group		Children Group	
Orientation	0.029	4.782	0.972	0.001
Device/Thumb Mode	0.009	3.887	0.000	13.620
Owns Touch Device	0.008	7.066	0.000	25.648
Experience	0.002	4.165	N/A	N/A

TABLE 7.6: *ANOVA Results for* Duration. *The first value in each cell is the significance value, the second is the F-value as calculated by the ANOVA method.*

of 10mm. In Table 7.4 the whole square size is used to group the results. Roughly the results reinforce the hypothesis, that bigger target sizes will lead to better accuracy and execution speed. Not all bigger sizes per-form better than smaller sizes. More research is needed here to find the cause of this. Generally the standard deviation (and therefore variance) of accuracy and duration decreases with increasing button size (in both tables). This might be a sign of users being more confident with bigger sizes or simply more users can reliably use the virtual buttons as their size increases. There also seems to be a point till where user performance increases faster with increasing size. Unfortunately our data is not abled to support such a hypothesis statistically, so this is also a point needing further research

We did a Pearson's test of correlation between the accuracy, user's age, the virtual button size, the minimal length or width of a button (called MinSize for the remainder of this paper) and the duration of a test. We got a significant correlation between accuracy and age, button size and MinSize for both test groups. The same goes for duration, respectively. Since Pearson's test of correlation is well known to be impacted by outliers, we confirmed our results

with an additional Spearman-correlation test. All correlations were confirmed, except for the correlation of duration and MinSize in the children group, so correlation here is improbable.

Analyses of Variance (i.e. ANOVE) was then done (Results in tables 7.5 and 7.6) to test the influence of the virtual button position on the accuracy. Separate ANOVAs were performed for each combination of device and grid layout. One of the interesting findings here is the fact, that for all devices the middle button in the 1x3 (one single row of 3 buttons) layout was significantly (on a 5% level) harder to hit, while this does not hold true for the 3x1 layout. Here no significant differences could be found. Other layouts with statistically significant inaccurate buttons are 2x3, 3x2 and 3x3.

For the 3x3 layout, the buttons in the middle column had least accuracy, with the top and bottom button in this column having even lesser accuracy than the middle one. This holds true on all devices. For this layout the middle device size has the highest accuracy for all individual buttons. In the 3x2 and 2x3 layouts, also the middle buttons had lower accuracy than the other buttons, again on all devices. This is probably due to the fact that buttons near a corner are easier to find without looking on the device.

7.4.4 Conclusion

Our evaluation shows, that there are several influence factors confirmed for eyes-free virtual button interaction on smart phones and tablets. The device size is important as one might assume as is the actual size of the target. As the improvement of accuracy when using the iPad over the iPad Mini is not significant, this hints, that there is a point, where enlarging the mobile device does not yield better results for accuracy. Since grid sizes bigger than 3x3 virtual buttons are strongly discouraged by Yi et.al. [108], this is understandable. Also we have to keep in mind that larger devices incur longer distances for the user's finger to travel above the tablet's surface. This can be seen at the missing correlation between the minimal target size and test duration in the children group, since for children the size difference of the devices are even bigger. Taking the statements of the test subjects into account, most users were most content with the middle sized iPad Mini. Since there is a trade-off between the target button size, the weight of the device (connected to the size

of the device) and the distance the user's hand has to move to acquire a target button, there is a sweet spot in device size.

For the design of virtual buttons we can conclude, that it is important to use the minimal possible amount of buttons and to move secondary functionality into other interaction mechanisms, like gestures or an on-screen menu. Especially since application designers cannot influence the factors of age and touchscreen experience of their potential users, they are well-advised to keep the user interface as simple as possible. Especially when developing for platforms that target several different device sizes (such as Android or iOS), the designers have to keep in mind, that they might not know the actual device size their application is used and thus not the actual target sizes of virtual buttons they use.

When using only three different buttons, it is advised to arrange them in column order, as there is no button with lower accuracy in that layout as found out above. In general the 3x1 and 2x2 layouts are most suited for eyes-free interaction. If more functions are needed, applications designers should take care to identify critical functionality, that causes most problems if invoked by mistake. Important functions should be assigned to a button in a corner of the device, unimportant ones can be connected with the middle buttons. The centre button in the 3x3 layout has a special role, as it has higher accuracy than the buttons to its sides, but less than the corner buttons.

We can also conclude, that virtual buttons is a concept, that is simple enough for even some small children to understand and use properly. As the number of children owning their own touch device is steadily increasing, this is an important point for application design. While the actual accuracy numbers are fairly low for the children test group, one has to keep in mind that eyes-free interaction is not easy for children at all. As future work the results mentioned in this paper need thorough evaluation to statistically prove their validity. Especially looking for a perfect size and the influence of grid layout and device size is very interesting for further evaluation. Still the results presented can be used as a hint for designing apps using the virtual button approach.

7.5 Industry Application

The work presented here in this and the previous chapters is the base for a industry project done at the time of writing with the German car manufacturer Volkswagen. They opted to have interaction through mobile devices included in ongoing project "Virtueller Meisterbock".

A Meisterbock is used in Quality Ensurance. It is a precision fixture where multiple body parts of a car can be mounted replicating the original scheme. This enables visually examination of the interplay of different parts of different manufacturers. If there are any issues concerning look-and-feel, general impression, haptic, optic, fitting, function, measurements or mountability, they can be identified in this early stage [2, 78].

The project focuses on virtualizing the physical Meisterbock. It is of course desirable to present this to others, in order to enable collaboration. As already mentioned in the initial chapter, large displays are great for presentation purposes, allowing more people to see more at the same time. It ease the interaction in this setting a mobile control is to be implemented into the software. This allows easy control of the application while presenting.

The actual benefits of a mobile control are as follows:

- **Interaction elements on the main application can be hidden:**
  Since interaction takes place through a mobile device, elements, such as buttons, can be hidden in the main application. This removes distracting graphics on the presentation screen and also increases available screen estate.
- **Interaction can be simplified:**
  Due to the huge number of functions available in the main application, the user interface is itself quite complex. Since only a part of these functions are needed for presentation purposes, only a simplified interface is needed on the mobile device. Additionally touch interaction usually feels more intuitive for the user, increasing user satisfaction. Through gesture input, the whole user experience can be further improved.
- **Multi-User interaction:**
  Naturally, in a presentation multiple users are involved. Through the usage of multiple devices, more users can be directly involved. Even if

only one device is used, a mobile device can be passed easier to another person and the simplified user interface allows untrained users to do basic interaction.

While this is no scientific evaluation, it is still described in this chapter to show, that there is interest on the side of the industry in applying this research in real-world problems and applications.

7.6 Conclusion

We have presented the evaluation results of two visualization techniques, the stereoscopic highlighting and the reflection layer extension. Stereoscopic highlighting in node-link diagrams can be used to encode ordinal data properties as depth of a 2D graph layout, making use of the capabilities of 3D displays. Our evaluation found out, that most users prefer to have 3 levels of depth, so classification might be needed to match this number. While this might seem like a severe limitation at first glance, it is not uncommon to have 3 different values e.g. when using the red-yellow-green color scheme.

The reflection layer allows to further improve the depth perception, even to a level where no stereoscopic display is necessary anymore to still perceive the depth cue. This helps to develop visualization applications, that can make use of both 2D and 3D displays.

We described why it is a viable idea to add metadata to CAD models. By looking at the model of a robot and a lab, we can see that the optimal interaction method is dependant on the nature of the CAD data. When more data describing the model, such as points-of-interest, etc is available, applications can make heavy use of that when providing interaction.

For virtual buttons we have shown, that bigger device sizes improve the accuracy and to some point the interaction speed. There seems to be a "sweet spot" at which accuracy and speed are in an optimal combination. According to the results and user comments, this device size might be near the 200mm x 134.7mm of the iPad Mini. Also we can find correlations between user's age and accuracy, which hints at an influence. Other correlations include the button size and the minimal button width/height. Buttons at the corners of the touchscreen are more accurately accessible than center buttons. These

results can be considered when designing interfaces using virtual buttons.

Chapter 8

Conclusions and Future Directions

The previous chapters gave an overview of interaction with Large Displays and Virtual Reality Environments, focused on using smart phones and tablets.

8.1 Discussion of Results

The results of this thesis will be summarized and discussed in this section. After a presentation of the various types of Large displays and the necessity for new interaction methods, an overview of the work done by other researchers in this field was given and its applicability for Virtual Reality settings was rated.

Mobile Interaction: Specialized Approaches - Proof of Concept

In chapter 3 we presented a proof-of-concept for large display interaction using smart phones to interact with four different use-cases. The use-cases are 3-DOF-positioning in three dimensions using the smart phone's touchscreen and accelerometer simultaneously, camera positioning on a 2D plane in a 3D scene using a joystick metaphor, tagging spots on a map and solving a jigsaw-puzzle. The 3D use-cases feature eyes-free interaction, while the other scenarios had their visual data transferred to the smart phone in order to allow independent examination and editing. While these interaction methods are primarily designed for the proof-of-concept application and its evaluation, they still can be applied to real-world applications.

The 3D positioning technique can be used in cases similar to the described one: when there are one or more objects in a 3D scene that need to be positioned in all three dimensions, but when rotation is not of interest. Rotation can be addressed by having positioning and rotation modes, much like in current 3D modelling applications. It can also be used in a multi-user case. But if more sophisticated functions are available, this method might not be sufficient, as it does not provide any functions beside positioning and simple 2-button selection.

The camera control using a smart phone, is less versatile and can only be used for cases where the camera follows a plane in 3D space. A possible real-world application can be found in the domain of architecture, where a house or structure can be simulated and explored using the smart phone. This could even be done through a public advertisement screen to which casual users can connect and explore the advertised house.

The 2D use-cases transfer parts of the screen to the smart phone to allow local changes. This is generally usable for any 2D interactions not too complex to be done on the touchscreen of a mobile device. With more sophisticated approaches it could also be used in a multi-user environment to partition a workload to individual users. Unfortunately this needs special implementation for each application using this design. When dealing with large datasets this approach might be unusable due to the small size of the mobile devices compared to the space needed to display the data.

But the main contribution of these use-cases is an evaluation where we show the viability of the approach of using smart phones as input devices for large display interaction.

Later in that chapter we presented a direct multi-touch interaction technique using the touchscreen of a tablet as a transparent surface into the virtual world of a CAVE. This allows intuitive interaction without the use of a flystick or a even more expensive VR glove. While not providing the haptic feedback of a VR glove, the well-known multi-touch gestures of tablets can be used and applied in the Virtual Reality of a CAVE. Generally this can be used for any cases where no special haptic feedback is needed. As added value, the tablet can be used to display additional information, especially 2D information, such as textual, this tends to be difficult to see/read in most CAVEs.

Mobile Interaction: Solving Common Issues

Common issues of large display interaction were tackled in chapter 4. We described a sensor-fusion method which allows smart phones and tablets to track their position and orientation in micro-scale without additional hardware. Using camera, accelerometer and compass data from the smart phone or tablet, we can estimate the position of the device quite accurately. Only in cases of fast movement, the camera gives blurred pictures which leads to inaccuracies with this method. But as the camera picture is restored, position estimation is accurate again. Given the amount of money saved when expensive tracking equipment is not needed, this does not limit the approach in a great way. With this method a low-cost VR environment can be created from a consumer-level 3D TV or monitor, a normal PC with WiFi connectivity and a smart phone or tablet. Since we currently are using markers to help estimating the position, there needs to be enough space to place the markers, and, of course, the markers need to be present. This limits this method to prepared spaces. With the CPU power of mobile devices still increasing rapidly, marker-less tracking and self-localization using a feature-map (i.e., based on SIFT, SURF or similar) will become possible, thus making the preparation unnecessary. Due to the reliance on the camera of this technique, its efficiency scales with camera quality and lighting parameters. But camera quality is close to a non-issue on modern smart phones and tablets, and adequate lighting can easily be provided, as the usage environment needs preparation anyway. A last limitation is give by the inaccuracy when moving the device quickly. In applications where the actual path of the device is important this might cause issues. But if only positioning without adhering to a special pathing is needed, this method stays perfectly valid. It can also be used for multiple users, since the devices can share the same markers. Scalabiltiy is hindered by occlusion of markers by co-users, though.

To address the distal-access-problem, we presented a solution using a tablet to act as a trackpad. Users can use multi-touch to adjust the mouse pointer speed while moving it. Thus pointer speed can be set to high for longer distances and to low for exact positioning. Additionally the cursor can be highlighting through a multi-touch gesture, in case the user loses track of it. While this method needs some training from the user to be used to its full extend,

it does not need any screen distorting effects, such as icons bending towards the mouse cursor. Again, this method can be applied to multiple users, the limiting factor being the number of cursors that can be displayed simultaneously on a large display without causing confusion. The method could also be extended to work on touch-capable computer mice (e.g. the Apple Magic Mouse^{TM}), or employing the mouse wheel for speed-scaling on normal mice.

Mobile Interaction: Marking Menus

Chapter (5) presented the usage of Marking Menus on smart phones and tablets as a generalized interaction method. It can be employed regardless of special hardware or software, even on standard personal computers. The principle of Marking Menus is to show a menu hierarchy that can be traversed through multiple connected touch strokes, that form a gesture. Especially for frequently used gestures, this interaction can be performed eyes-free (called expert modes in the context of Marking Menus). By allowing menu items not only to be selected, but also to act as draggable sliders, value control is possible. We extended this technique into multi-dimensional sliders, through single- and multi-touch. Also multi-touch can be used to allow clutching, re-selection of menu items and add further menus. Using this generalized method, we can solve the problem of diversity of input devices in different large display environments.

Cases where this approach is inferior include very simple applications where a menu is not needed, such as positioning of a single object, basic navigation tasks with only 2 or 3 DOF, etc. Still the menu structure can be setup to support these applications without displaying an actual menu, due to the fact value control is supported even on the top-level of a “menu”.

When direct selection is needed, this method can also be cumbersome for the user. While the smart phone can be used as a pointing device, this can lead its display facing away from the user while pointing, disallowing novice mode. By allowing to mark first and then interact, this can be overcome at the cost of interaction speed.

Mobile Interaction And Virtual Reality Visualization: Application In Embedded Systems Development

A description of an application for the visualization of safety and security issues of embedded systems is contained in chapter 6. This application is scalable in terms of display size, works in any of the large display setups as presented by Ni et al. [85], and is capable of single and multi-user support. This real-life application is developed in favor instead of creating another proof-of-concept application to stress the point of validity of the whole concept of large display interaction through mobile devices and scalable applications.

We developed this application to help safety and hardware engineers to collaborate in the development of safer embedded systems. There are two views, where one of each is familiar to either of the domain experts. The views are linked which helps each party to present their concerns about the embedded system to the other party. The application includes stereoscopic highlighting for its graph representation of the embedded system. The elements in the graph are given a different depth, depending on their importance to the user. This allows to visually encode this importance in the graph without needing to use other visual cues, such as color or shape, making it possible to encode further information into the graph. Using a reflection layer at the bottom of the graph visualization, the results can be improved. This helps even users not being able to see the depth cues (either due to the lack of a 3D display or because of visual impairment) to perceive the importance of elements in the graph. Finally we displayed the interaction method used for this application using mobile devices. It employs not only Marking Menus, but is a dual-mode app, that can display information provided by the host application in order to leverage multi-user interaction.

The stereoscopic highlighting as presented in this chapter can be applied to any node-link-diagram to be presented on a 3D display, or even on 2D displays when using the reflection layer. The only cost of the reflection layer is the screen space needed for the reflection, leaving less space for the actual graph. Also the method relies on showing importance of nodes, since having more important nodes closer to the user is intuitive. Care has to be taken when using this technique for other ordinal aspects of nodes to not confuse users.

The safety visualization can generally be extended to work with any kind of data. Its strength lies in combining two connected datasets and combining them in a single application to leverage collaboration between two parties not having the same common domain language. This happens a lot nowadays, due to the increased collaboration requirements in almost all fields, especially in engineering fields.

Evaluation

Chapter 7 contains four evaluations and their results. Two evaluations are done about techniques presented in chapter 6: The Extended Stereoscopic Highlighting and its Reflection Layer Extension. We showed the results of our evaluations, where we found out, that the techniques are useful to provide an overview of important or unimportant parts of a data set. While it takes time for users to get accustomed to this kind of visualization it is easily combinable with other visualization techniques. Through the Reflection Layer Extension, this is more accessible to both people having problems using stereoscopic displays and normal users. We can also observer the limits of stereoscopic highlighting, as most users cannot reliably recognize the depth of nodes when more than 3 layers of depth are presented.

With a case study, we argue for including metadata in 3D models to improve interaction possibilities. While some data structures already contain basic information, such as VRML containing hints, such as *NONE*, *FLY*, *WALK* and *EXAMINE*, more information can be used for complex 3D models. Possibilities include changing the interaction metaphor based on camera location (e.g., object centered for exterior views and camera-centered for interior viewpoints). While the possibilities given by metadata seems almost limitless, the problem is the increased workload for model designers when adding the metadata.

Virtual Buttons are the focus of the last study in that chapter. The interaction area of a touch device is divided into evenly spaced buttons through an arbitrary sized grid. The effect of device size, grid size, device orientation and user factors are evaluated in order to gain insight in better and worse layouts depending on target devices and audience. Results include usable grid sizes and “best” buttons that are hit more consistently than other independent of other factors. The evaluation study also included children to get an insight

into special needs of children when using smart phones and/or tablets. Additionally this broadens the user properties of our study and establishes the simplicity of the virtual buttons approach, as the children were able to use it.

An overview of an industrial application was also given. This shows the interest of the industry for these techniques and methods.

8.2 Conclusion

As asked by the goals of this thesis, we did find proof-of-concept for using smart phones and tablets as input devices, we developed a generalized input method, that does not rely on a special usage environment, hardware or software. It does scale well with the number of users. And we developed an application, that is scalable based on display size and also properties, such as 2D or 3D displays.

The additional goals were also met, as we found multiple solutions to common issues of large displays. The scalable pointer movement solves the distal-access-problem, we present possibilities for low-cost applications of mobile devices for Virtual Reality applications, such as the self-tracking, or the touch surface. Additionally we included a rating of related work using smart phones as input devices.

With the main and additional goals, we do not only present a single approach for large display interaction using smart phones and tablets. Instead, we present a collection of methods, that are not mutally exclusive and can be combined as application's or system designer's needs be. The generalized Marking Menu method can be used to control a pointer for a 2D GUI. The scalable cursor control can be applied to this method without a problem. Another example is to combine the touch surface in VR environments with Marking Menus to make them context sensitive.

8.3 Outlook

Research is never finished. From the results of this dissertation, there are several points still worth of looking into.

- **Specialized Approaches:**
  Interaction especially tailored towards their application are always a viable choice. While needing more effort than using generalized methods, the results can be more satisfying. Thus, creating special interactions for an application are always an open field of research.
- **Smart Watches:**
  The recent advent of smart watches creates the opportunity of employing these small wearable touch devices as well for interaction. They are similar enough to smart phones and tablets to use the same basic approaches for interaction, but still different to need a validation or adaptation.
- **Virtual Buttons:**
  The evaluation of virtual buttons can be extended to the new size of smart watches, where the smaller touch area will effect the viable grid sizes for sure.
- **3D Interaction:**
  Due to time constraints, the domain specific language for 3D models was never finished. This is still an open research question.

While the results collected in this dissertation might seem broad, they all focus on improving the interaction with large displays (and their special case of Virtual Reality Environments) through the usage of smart phones and tablets.We presented approaches and evaluations of techniques employable for both application and interaction designers. Especially with the recent advent of cheap consumer-level virtual reality glasses, that lack rich interaction interfaces, this research field is at least as important as it was before. Through price-drops in hardware, large displays become more common in industrial settings as well, especially when it comes to collaboration. The importance of this research is attested by the industry looking for applications and interaction methods for their large displays.

Bibliography

[1] ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 — Virtual Reality Modeling Language (VRML) , 12 2003. URL http://www.web3d.org/x3d/specifications/vrml/.

[2] February 2010. URL http://autogramm.volkswagen.de/01-02_10/wolfsburg/wolfsburg_06.html.

[3] Johnny Accot and S. Zhai. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 295–302. ACM, 1997. ISBN 0897918029. URL http://dl.acm.org/citation.cfm?id=258760.

[4] Yasmin Al-Zokari, Taimur Khan, Daniel Schneider, Dirk Zeckzer, and Hans Hagen. Cakes: Cake metaphor for analyzing safety issues of embedded systems. *Dagstuhl Follow-ups*, 2010.

[5] Yasmin I. Al-Zokari, Daniel Schneider, Dirk Zeckzer, Liliana Guzman, Yarden Livnat, and Hans Hagen. Enhanced CakES representing Safety Analysis results of Embedded Systems. In *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 791–798, USA, 2011. IEEE. ISBN 978-83-60810-39-2. doi: \url{http://fedcsis.eucip.pl/proceedings/pliks/fedcsis.pdf}.

[6] Basak Alper, Tobias Hollerer, JoAnn Kuchera-Morin, and Angus Forbes. Stereoscopic highlighting: 2d graph visualization on stereo displays. *IEEE Transactions on Visualization and Computer Graphics*, 17(12): 2325–2333, December 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2011.234. URL http://dx.doi.org/10.1109/TVCG.2011.234.

[7] Ragaad AlTarawneh, Jens Bauer, Patric Keller, Achim Ebert, and Peter Liggesmeyer. Essavis: A framework to visualize safety aspects in embedded systems. In *SIGRAD 2012, Interactive Visual Analysis of Data*, November 2012.

[8] Ragaad AlTarawneh, Jens Bauer, Shah Rukh Humayoun, Patric Keller, and Achim Ebert. The extended stereoscopic highlighting technique for node-link diagrams: An empirical study. In *14th IASTED International Conference on Computer Graphics and Imaging (CGIM 2013)*, February 2013.

[9] Ragaad AlTarawneh, Jens Bauer, ShahRukh Humayoun, Patric Keller, and Achim Ebert. The reflection layer extension to the stereoscopic highlight technique for node-link diagrams: An empirical study. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Baoxin Li, Fatih Porikli, Victor Zordan, James Klosowski, Sabine Coquillart, Xun Luo, Min Chen, and David Gotz, editors, *Advances in Visual Computing*, volume 8034 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-41938-6. doi: 10.1007/978-3-642-41939-3_1. URL `http://dx.doi.org/10.1007/978-3-642-41939-3_1`.

[10] Ragaad AlTarawneh, Jens Bauer, Shah Rukh Humayoun, Achim Ebert, and Peter Liggesmeyer. Essavis++: An interactive 2dplus3d visual environment to help engineers in understanding the safety aspects of embedded systems. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '14, pages 201–204, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2725-1. doi: 10.1145/2607023.2611453. URL `http://doi.acm.org/10.1145/2607023.2611453`.

[11] Shiri Azenkot and Shumin Zhai. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 251–260. ACM, 2012.

[12] Mathias Baglioni, Eric Lecolinet, and Yves Guiard. JerkTilts: using accelerometers for eight-choice selection on mobile devices. In *Proceedings of the 13th international conference on multimodal interfaces*, ICMI '11, pages 121–128, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0641-6. doi: 10.1145/2070481.2070503. URL `http://doi.acm.org/10.1145/2070481.2070503`.

[13] Gilles Bailly and Eric Lecolinet. Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. *AVI '08 Proceedings of the working conference on Advanced visual interfaces*, pages 15–22, 2008. URL `http://dl.acm.org/citation.cfm?id=1385575`.

[14] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. Wave menus: improving the novice mode of hierarchical marking menus. *INTERACT 2007*, pages 475–488, 2007. URL `http://www.springerlink.com/index/l4g6x72707336u51.pdf`.

[15] Rafael Ballagas, M. Rohs, and J.G. Sheridan. Sweep and Point and Shoot: Phonecam-based Interactions for Large Public Displays. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1200–1203. ACM, 2005. ISBN 1595930027. URL `http://dl.acm.org/citation.cfm?id=1056876`.

[16] Rafael Ballagas, Jan Borchers, Michael Rohs, and Jennifer G Sheridan. The Smart Phone: A Ubiquitous Input Device. *IEEE Pervasive Computing*, 5(1):70–77, January 2006. ISSN 1536-1268. doi: 10.1109/MPRV.2006.18. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1593574`.

[17] Rafael Ballagas, Jan Borchers, Michael Rohs, Jennifer G Sheridan, James Foley, Victor Wallace, and Peggy Chan. The Smart Phone : A The Input Design Space. *Shoot*, 2006.

[18] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens: combining display technology with visualization techniques. In *Proceedings of the 14th annual ACM symposium on User interface*

software and technology, UIST '01, pages 31–40, New York, NY, USA, 2001. ACM. ISBN 1-58113-438-X. doi: 10.1145/502348.502354. URL http://doi.acm.org/10.1145/502348.502354.

[19] Patrick Baudisch, Edward Cutrell, Dan Robbins, Mary Czerwinski, Peter Tandler, Benjamin Bederson, Alex Zierlinger, et al. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *Proceedings of INTERACT*, volume 3, pages 57–64, 2003.

[20] J. Bauer, S. Thelen, and A. Ebert. Using smart phones for large-display interaction. In *User Science and Engineering (i-USEr), 2011 International Conference on*, pages 42–47, Nov 2011. doi: 10.1109/iUSEr.2011.6150533.

[21] Jens Bauer and Achim Ebert. Mobile devices for virtual reality interaction. a survey of techniques and metaphors. In Guido Brunnett, Sabine Coquillart, Robert van Liere, Gregory Welch, and Libor Váša, editors, *Virtual Realities*, volume 8844 of *Lecture Notes in Computer Science*, pages 91–107. Springer International Publishing, 2015. ISBN 978-3-319-17042-8. doi: 10.1007/978-3-319-17043-5_6. URL http://dx.doi.org/10.1007/978-3-319-17043-5_6.

[22] Jens Bauer and Achim Ebert. Virtual buttons for eyes-free interaction: A study. 2015 (Accepted for Publication).

[23] Jens Bauer, Sebastian Thelen, and Achim Ebert. Using Smart Phones for Large-Display Interaction. In *2nd International Conference on User Science and Engineering (I-USEr)*, 2011.

[24] Jens Bauer, Sebastian Thelen, and Achim Ebert. Evaluation of Mobile Phones for Large Display Interaction. In Christoph Garth, Ariane Middel, and Hans Hagen, editors, *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*, volume 27 of *OpenAccess Series in Informatics (OASIcs)*, pages 103–112, Dagstuhl, Germany,

2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-46-0. doi: http://dx.doi.org/10.4230/OASIcs.VLUDS.2011.103. URL http://drops.dagstuhl.de/opus/volltexte/2012/3744.

[25] Jens Bauer, Achim Ebert, Oliver Kreylos, and Bernd Hamann. Marking menus for eyes-free interaction using smart phones and tablets. In Alfredo Cuzzocrea, Christian Kittl, DimitrisE. Simos, Edgar Weippl, and Lida Xu, editors, *Availability, Reliability, and Security in Information Systems and HCI*, volume 8127 of *Lecture Notes in Computer Science*, pages 481–494. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40510-5. doi: 10.1007/978-3-642-40511-2_35. URL http://dx.doi.org/10.1007/978-3-642-40511-2_35.

[26] Jens Bauer, Achim Ebert, Oliver Kreylos, and Bernd Hamann. Generalized eyes-free interaction for use with large displays. *Workshop on Prototyping to Support the Interaction Designing in Mobile Application Development (PID-MAD 2013)*, 2013.

[27] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33832-1. doi: 10.1007/11744023_32. URL http://dx.doi.org/10.1007/11744023_32.

[28] Pierre Bieber, Christian Bougnol, Charles Castel, Jean pierre Heckmann, Christophe Kehren, Sylvain Metge, Christel Seguin, P. Bieber, C. Bougnol, C. Castel, J. p. Heckmann, C. Kehren, and C. Seguin. Safety assessment with altarica - lessons learnt based on two aircraft system studies. In *In 18th IFIP World Computer Congress, Topical Day on New Methods for Avionics Certification*, page 26, 2004.

[29] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 73–80, New York, NY,

USA, 1993. ACM. ISBN 0-89791-601-8. doi: 10.1145/166117.166126. URL http://doi.acm.org/10.1145/166117.166126.

[30] Andrew Bragdon, Eugene Nelson, Yang Li, and Ken Hinckley. Experimental analysis of touch-screen gesture designs in mobile environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 403–412. ACM, 2011.

[31] Ulrik Brandes, Tim Dwyer, and Falk Schreiber. Visualizing related metabolic pathways in two and a half dimensions. In Giuseppe Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 111–122. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-20831-0. doi: 10.1007/978-3-540-24595-7_10. URL http://dx.doi.org/10.1007/978-3-540-24595-7_10.

[32] Richard Brath. 3d interactive information visualization: Guidelines from experience and analysis of applications. In *HCI (2)*, pages 865–868, 1997.

[33] William Buxton. Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput. Graph.*, 17(1):31–37, January 1983. ISSN 0097-8930. doi: 10.1145/988584.988586. URL http://doi.acm.org/10.1145/988584.988586.

[34] Han Chen, Yuqun Chen, Adam Finkelstein, Thomas Funkhouser, Kai Li, Zhiyan Liu, Rudrajit Samanta, and Grant Wallace. Data distribution strategies for high-resolution displays. *Computers & Graphics*, 25:811–818, 2001.

[35] Y. Chen, H. Chen, D.W. Clark, Z. Liu, G. Wallace, and K. Li. Software environments for cluster-based display systems. In *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, pages 202–210, 2001. doi: 10.1109/CCGRID.2001.923194.

[36] Andy Cockburn and Bruce McKenzie. Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 203–210, New York, NY, USA, 2002. ACM.

ISBN 1-58113-453-3. doi: 10.1145/503376.503413. URL http://doi.acm.org/10.1145/503376.503413.

[37] C. Collins and S. Carpendale. Vislink: Revealing relationships amongst visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1192–1199, Nov 2007. ISSN 1077-2626. doi: 10.1109/TVCG.2007.70521.

[38] Ricardo Costa, Davide Carneiro, Paulo Novais, Luís Lima, José Machado, Alberto Marques, and José Neves. Ambient assisted living. In Juan Corchado, Dante Tapia, and José Bravo, editors, *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008*, volume 51 of *Advances in Soft Computing*, pages 86–94. Springer Berlin / Heidelberg, 2009. ISBN 978-3-540-85866-9.

[39] Raimund Dachselt and Robert Buchholz. Throw and Tilt – Seamless Interaction Across Devices Using Mobile Phone Gestures. *Proc. MEIS 2008*, pages 272–278, 2008. URL http://wwwisg.cs.uni-magdeburg.de/uise/Forschung/Publikationen/2009-MEIS-Throw-and-Tilt.pdf.

[40] Matthias Deller and Achim Ebert. Modcontrol – mobile phones as a versatile interaction device for large screen applications. In Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2011*, volume 6947 of *Lecture Notes in Computer Science*, pages 289–296. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23770-6. doi: 10.1007/978-3-642-23771-3_22. URL http://dx.doi.org/10.1007/978-3-642-23771-3_22.

[41] Matthias Deller, Achim Ebert, Michael Bender, Stefan Agne, and Henning Barthel. Preattentive visualization of information relevance. In *Proceedings of the International Workshop on Human-centered Multimedia*, HCM '07, pages 47–56, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-781-0. doi: 10.1145/1290128.1290137. URL http://doi.acm.org/10.1145/1290128.1290137.

[42] Matthias Deller, Achim Ebert, Stefan Agne, and Daniel Steffen. Guiding attention in information-rich virtual environments. In J.J. Villanueva, editor, *IASTED International Conference on Visualization, Imaging and Image Processing*, pages 310–315. International Association of Science and Technology for Development (IASTED), ACTA Press, 9 2008.

[43] Alan Dix, Janet E. Finlay, Gregory D. Abowd, and Russell Beale. *Human-Computer Interaction (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003. ISBN 0130461091.

[44] P. Eades. Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications*, pages 10–36, 1992.

[45] Peter Eades and Qingwen Feng. Multilevel visualization of clustered graphs. pages 101–112. Springer-Verlag, 1997.

[46] Michael Ebel. Mobile device self-localization for large display interaction. Master's thesis, Distance Study Programme, TU Kaiserslautern, September 2014.

[47] Nicolas Engel. Scalable cursor control on large displays using mobile devices. Master's thesis, Department of Computer Science, TU Kaiserslautern, March 2015.

[48] Kim M. Fairchild, Steven E. Poltrock, and George W. Furnas. Readings in information visualization. chapter SemNet: Three-dimensional Graphic Representations of Large Knowledge Bases, pages 190–206. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-533-9. URL `http://dl.acm.org/citation.cfm?id=300679.300751`.

[49] Paul M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.

[50] James Foley, Victor Wallace, and Peggy Chan. Human factors of Computer Graphics Interaction Techniques. *IEEE Computer Graphics and Applications*, 4(11):13–48, 1984.

[51] Jeremie Francone, Gilles Bailly, Laurence Nigay, and Eric Lecolinet. Wavelet Menus: A Stacking Metaphor for Adapting Marking Menus to Mobile Devices. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 2–5, 2009.

[52] Jérémie Francone, Gilles Bailly, Eric Lecolinet, Nadine Mandran, and Laurence Nigay. Wavelet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 173–180, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0076-6. doi: 10.1145/1842993.1843025. URL `http://doi.acm.org/10.1145/1842993.1843025`.

[53] Haichang Gao, Xuewu Guo, Xiaoping Chen, Liming Wang, and Xiyang Liu. YAGP: Yet Another Graphical Password Strategy. *Computer Security Applications Conference, Annual*, 0:121–129, 2008. ISSN 1063-9527. doi: http://doi.ieeecomputersociety.org/10.1109/ACSAC.2008.19.

[54] Sascha Gebhardt, Sebastian Pick, Thomas Oster, Bernd Hentschel, and Torsten Kuhlen. An evaluation of a smart-phone-based menu system for immersive virtual environments. In *3D User Interfaces (3DUI), 2014 IEEE Symposium on*, pages 31–34, March 2014. doi: 10.1109/3DUI.2014.6798837.

[55] Tovi Grossman and Ravin Balakrishnan. An evaluation of depth perception on volumetric displays. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, pages 193–200, New York, NY, USA, 2006. ACM. ISBN 1-59593-353-0. doi: 10.1145/1133265.1133305. URL `http://doi.acm.org/10.1145/1133265.1133305`.

[56] Taosong He and Arie E. Kaufman. Virtual input devices for 3d systems. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, pages 142–148, Washington, DC, USA, 1993. IEEE Computer Society. ISBN 0-8186-3940-7.

[57] Niels Henze, Enrico Rukzio, and Susanne Boll. 100,000,000 taps: analysis and improvement of touch performance in the large. In *Proceedings of*

the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, pages 133–142. ACM, 2011.

[58] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January 2000. ISSN 1077-2626. doi: 10.1109/2945.841119. URL `http://dx.doi.org/10.1109/2945.841119`.

[59] Bernhard Kaiser, Peter Liggesmeyer, and Oliver Mäckel. A new component concept for fault trees. In *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33*, SCS '03, pages 37–46, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc. ISBN 1-920-68215-5.

[60] Bernhard Kaiser, Catharina Gramlich, and Marc Förster. State/event fault trees—a safety analysis model for software-controlled systems. *Reliability Engineering & System Safety*, 92(11):1521 – 1537, 2007. ISSN 0951-8320. doi: http://dx.doi.org/10.1016/j.ress.2006.10.010. URL `http://www.sciencedirect.com/science/article/pii/S0951832006002092`. {SAFECOMP} 2004, the 23rd International Conference on Computer Safety, Reliability and Security.

[61] A Karlson, B Bederson, and J Contreras-Vidal. Understanding single-handed mobile device interaction. *Handbook of research on user interface design and evaluation for mobile technology*, pages 86–101, 2006.

[62] Taimur Khan, Daniel Schneider, Yasmin Al-Zokari, Dirk Zeckzer, and Hans Hagen. Framework for comprehensive size and resolution utilization of arbitrary displays. *Dagstuhl Follow-ups*, 2010.

[63] Kenrick Kin and B. Hartmann. Two-handed marking menus for multitouch devices. *ACM Transactions on Computer-Human Interaction (TOCHI) TOCHI Homepage archive*, 18(2):16:1–16:23, 2011. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-118.pdf`.

[64] B. Knegtering and A.C. Brombacher. Application of micro markov models for quantitative safety assessment to determine safety integrity levels as defined by the {IEC} 61508 standard for functional safety. *Reliability Engineering & System Safety*, 66(2):171 – 175, 1999. ISSN 0951-8320. doi: http://dx.doi.org/10.1016/S0951-8320(99)00034-4. URL `http://www.sciencedirect.com/science/article/pii/S0951832099000344`.

[65] O. Kreylos, April 2015. URL `http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/`.

[66] Oliver Kreylos. Environment-Independent VR Development. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Paolo Remagnino, Fatih Porikli, Jörg Peters, James Klosowski, Laura Arns, Yu Chun, Theresa-Marie Rhyne, and Laura Monroe, editors, *Advances in Visual Computing*, volume 5358 of *Lecture Notes in Computer Science*, pages 901–912. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-89638-8. URL `http://dx.doi.org/10.1007/978-3-540-89639-5_86`.

[67] Gordon Kurtenbach and William Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 482–487, New York, NY, USA, 1993. ACM. ISBN 0-89791-575-5. doi: 10.1145/169059.169426. URL `http://doi.acm.org/10.1145/169059.169426`.

[68] Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, CHI '94, pages 258–264, New York, NY, USA, 1994. ACM. ISBN 0-89791-650-6. doi: 10.1145/191666.191759. URL `http://doi.acm.org/10.1145/191666.191759`.

[69] Gordon Paul Kurtenbach. *The Design and Evaluation of Marking Menus*. PhD thesis, University of Toronto, 1993.

[70] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies.

In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: 10.1145/223904.223956. URL http://dx.doi.org/10.1145/223904.223956.

[71] Jonh Lamping and Ramana Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing*, 7(1):33 – 55, 1996. ISSN 1045-926X. doi: http://dx.doi.org/10.1006/jvlc.1996.0003. URL http://www.sciencedirect.com/science/article/pii/S1045926X96900038.

[72] Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. Lee and Seshia, 1 edition, 2010. ISBN 978-0-557-70857-4. URL http://chess.eecs.berkeley.edu/pubs/794.html.

[73] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[74] I. Scott Mackenzie and R. William Soukoreff. A two-ball mouse affords three degrees of freedom. In *CHI'97 Conference Companion*, pages 303–304. Press, 1997.

[75] Anil Madhavapeddy, D Scott, and Richard Sharp. Using Camera-Phones to Enhance Human-Computer Interaction. *Proceedings of Ubiquitous*, 2004. URL http://www.cl.cam.ac.uk/research/dtg/publications/public/ubicomp2004/madhavapeddy.pdf.

[76] D.C. McCallum and P. Irani. Arc-pad: Absolute + Relative Cursor Positioning for Large Displays with a Mobile Touchscreen. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 153–156. ACM, 2009. ISBN 9781605587455. URL http://dl.acm.org/citation.cfm?id=1622205.

[77] Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. A sorting classification of parallel rendering. *IEEE Comput. Graph. Appl.*, 14(4):23–32, July 1994. ISSN 0272-1716. doi: 10.1109/38.291528. URL http://dx.doi.org/10.1109/38.291528.

[78] Bruce Morey. New metrology culture improving chrysler quality. *Manufacturing Engineering*, pages 69 – 80, September 2013.

[79] T. Munzner. H3: Laying out large directed graphs in 3d hyperbolic space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, INFOVIS '97, pages 2–, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8189-6. URL http://dl.acm.org/citation.cfm?id=857188.857627.

[80] Tamara Munzner. Drawing large graphs with h3viewer and site manager (system demonstration). In *In Proceedings of Graph Drawing'98, number 1547 in Lecture Notes in Computer Science*, pages 384–393. Springer-Verlag, 1998.

[81] Tamara Munzner and Paul Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. In *Proceedings of the First Symposium on Virtual Reality Modeling Language*, VRML '95, pages 33–38, New York, NY, USA, 1995. ACM. ISBN 0-89791-818-5. doi: 10.1145/217306.217311. URL http://doi.acm.org/10.1145/217306.217311.

[82] B Myers, C Peck, Jeffrey Nichols, and Dave Kong. Interacting at a Distance Using Semantic Snarfing. *Ubicomp 2001: Ubiquitous*, 15213, 2001. URL http://www.springerlink.com/index/4EM3F105X1UQB470.pdf.

[83] Brad A. Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A. Chris Long. Interacting at a distance: Measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 33–40, New York, NY, USA, 2002. ACM. ISBN 1-58113-453-3. doi: 10.1145/503376.503383. URL http://doi.acm.org/10.1145/503376.503383.

[84] Ken Nakayama and Gerald H. Silverman. Serial and parallel processing of visual feature conjunctions. *Nature*, 320(6059):264–265, 03 1986. URL http://dx.doi.org/10.1038/320264a0.

[85] Tao Ni, Greg S. Schmidt, Oliver G. Staadt, Mark A. Livingston, Robert Ball, and Richard May. A survey of large high-resolution display technologies, techniques, and applications. In *Proceedings of the IEEE Conference on Virtual Reality*, VR '06, pages 223–236, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 1-4244-0224-7. doi: 10.1109/VR.2006.20. URL http://dx.doi.org/10.1109/VR.2006.20.

[86] Ian Oakley and Junseok Park. Motion marking menus: An eyes-free approach to motion input for handheld devices. *Int. J. Hum.-Comput. Stud.*, 67(6):515–532, June 2009. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2009.02.002. URL http://dx.doi.org/10.1016/j.ijhcs.2009.02.002.

[87] Dan R Olsen Jr. and Travis Nielsen. Laser pointer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, pages 17–22, New York, NY, USA, 2001. ACM. ISBN 1-58113-327-8. doi: 10.1145/365024.365030. URL http://doi.acm.org/10.1145/365024.365030.

[88] Ken Perlin. Quikwriting: continuous stylus-based text entry. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pages 215–216, New York, NY, USA, 1998. ACM. ISBN 1-58113-034-1. doi: 10.1145/288392.288613. URL http://doi.acm.org/10.1145/288392.288613.

[89] Stephen D. Peterson, Magnus Axholt, and Stephen R. Ellis. Objective and subjective assessment of stereoscopically separated labels in augmented reality. *Computers & Graphics*, 33(1):23 – 33, 2009. ISSN 0097-8493. doi: http://dx.doi.org/10.1016/j.cag.2008.11.006. URL http://www.sciencedirect.com/science/article/pii/S0097849308001507.

[90] Stuart Pook, Eric Lecolinet, Guy Vaysseix, Enst Cnrs Ura, and Môquet Bp. Control Menus : Execution and Control in a Single Interactor. *CHI*

'00 extended abstracts on Human factors in computing systems, (April): 263–264, 2000.

[91] George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. Large Display User Experience. *Computer Graphics and Applications, IEEE*, 25(4):44–51, 2005. doi: 10.1109/MCG.2005.88.

[92] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: Animated 3d visualizations of hierarchical information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 189–194, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3. doi: 10.1145/108844.108883. URL `http://doi.acm.org/10.1145/108844.108883`.

[93] Michael Rohs. Marker-Based Embodied Interaction for Handheld Augmented Reality Games Marker-Based Embodied Interac-. *Virtual Reality*, 4(5), 2007.

[94] Volker Roth and Thea Turner. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1523–1526. ACM, 2009.

[95] Khoovirajsingh Seewoonauth, Enrico Rukzio, Robert Hardy, and Paul Holleis. Touch & Connect and Touch & Select: Interacting with a Computer by Touching it with a Mobile Phone. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 36. ACM, 2009. ISBN 9781605582818. URL `http://dl.acm.org/citation.cfm?id=1613905`.

[96] D.H. Stamatis. *Failure Mode and Effect Analysis: FMEA from Theory to Execution*. ASQ Quality Press, 2003. ISBN 9780873895989. URL `https://books.google.de/books?id=TTxI8jbTkVwC`.

[97] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The movable filter as a user interface tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 306–312, New York, NY,

USA, 1994. ACM. ISBN 0-89791-650-6. doi: 10.1145/191666.191774. URL http://doi.acm.org/10.1145/191666.191774.

[98] Kishore Swaminathan and Steve Sato. Interaction design for large displays. *interactions*, 4(1):15–24, January 1997. ISSN 1072-5520. doi: 10.1145/242388.242395. URL http://doi.acm.org/10.1145/242388.242395.

[99] Y. Tanaka, Y. Okada, and K. Niijima. Treecube: visualization tool for browsing 3d multimedia data. In *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on*, pages 427–432, July 2003. doi: 10.1109/IV.2003.1218020.

[100] Sebastian Thelen, Joerg Meyer, Achim Ebert, and Hans Hagen. A 3D Human Brain Atlas. *Modelling the Physiological Human*, pages 173–186, 2009. URL http://www.springerlink.com/index/046712m1u0535763.pdf.

[101] Sebastian Thrun and JohnJ. Leonard. Simultaneous localization and mapping. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 871–889. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-23957-4. doi: 10.1007/978-3-540-30301-5_38. URL http://dx.doi.org/10.1007/978-3-540-30301-5_38.

[102] Software Engineering Research Group: Dependability Kaiserslautern University. Essarel tool: Embedded systems sadety and reliability analyer. URL http://essarel.de/index.php?site=backgroundtext.

[103] J.J. van Wijk and H. van de Wetering. Cushion treemaps: visualization of hierarchical information. In *Information Visualization, 1999. (Info Vis '99) Proceedings. 1999 IEEE Symposium on*, pages 73–78, 147, 1999. doi: 10.1109/INFVIS.1999.801860.

[104] Yuntao Wang, Chun Yu, Jie Liu, and Yuanchun Shi. Understanding performance of eyes-free, absolute position control on touchable mobile phones. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '13,

pages 79–88, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2273-7. doi: 10.1145/2493190.2493215. URL http://doi.acm.org/10.1145/2493190.2493215.

[105] Colin Ware and Robert Bobrow. Supporting visual queries on medium-sized node-link diagrams. *Information Visualization*, 4(1):49–58, March 2005. ISSN 1473-8716. doi: 10.1057/palgrave.ivs.9500090. URL http://dx.doi.org/10.1057/palgrave.ivs.9500090.

[106] Colin Ware and Glenn Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Trans. Graph.*, 15 (2):121–140, April 1996. ISSN 0730-0301. doi: 10.1145/234972.234975. URL http://doi.acm.org/10.1145/234972.234975.

[107] Colin Ware and Peter Mitchell. Visualizing graphs in three dimensions. *ACM Transactions on Applied Perception (TAP*, 2008.

[108] Bo Yi, Xiang Cao, Morten Fjeld, and Shengdong Zhao. Exploring user motivations for eyes-free interaction on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2789–2792, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2208678. URL http://doi.acm.org/10.1145/2207676.2208678.

[109] Ji Soo Yi, Youn ah Kang, J.T. Stasko, and J.A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1224 –1231, nov.-dec. 2007. ISSN 1077-2626. doi: 10.1109/TVCG.2007.70515.

List of Own Publications

Book chapters

[1] Ragaad AlTarawneh, Jens Bauer, Shah Rukh Humayoun, Patric Keller, Achim Ebert The Reflection Layer Extension to the Stereoscopic Highlight Technique for Node-Link Diagrams: An Empirical Study. Presented at ISVC '13, Advances in Visual Computing, Lecture Notes in Computer Science, Springer, v. 8034, pp 1-12, 2013. [Book Chapter]

[2] Jens Bauer and Achim Ebert Mobile Devices for Virtual Reality Interaction. A Survey of Techniques and Metaphors. In Virtual Reality: Dagstuhl Seminar 2013.

Conference Papers (peer-reviewed)

[1] Jens Bauer and Achim Ebert. Virtual buttons for eyes-free interaction: A study. In Human-Computer Interaction – INTERACT 2015, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2015. (Accepted for Publication)

[2] Ragaad AlTarawneh, Jens Bauer, Nicole Menck, Shah Rukh Humayoun, and Achim Ebert assistME: A Collaborative Mobile Environment to Help Engineers in Maintaining the Factory Pipeline. MobilENG 2014 Workshop, in conjunction with 5th international conference on Complex Systems Design & Management (CSD&M) 2014, Paris, France, November 14, 2014.

[3] Ragaad AlTarawneh, Jens Bauer, Achim Ebert A Visual Interactive Environment for Enhancing Collaboration between Engineers for the Safety Analysis Mechanisms in Embedded Systems. In Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing

(CSCW Companion '14). ACM, New York, NY, USA, 125-128, 2014. http://doi.acm.org/10.1145/2556420.2556480

[4] Ragaad AlTarawneh, Jens Bauer, Shah Rukh Humayoun, Achim Ebert, and Peter Liggesmeyer Enhancing Understanding of Safety Aspects in Embedded Systems through an Interactive Visual Tool. In Proceedings of the companion publication of the 19th ACM International Conference on Intelligent User Interfaces (IUI Companion '14). ACM, New York, NY, USA, 9-12, 2014. http://doi.acm.org/10.1145/2559184.2559196

[5] Ragaad AlTarawneh, Andreas Griesser, Jens Bauer, Shah Rukh Humayoun, and Achim Ebert Poster: 3DintEx – A Tool to Explore Interactively the Structural and Behavioral Aspects of System Models in 3D Environments. In IEEE 9th Symposium on 3D User Interfaces (3DUI 2014), 141-142, March 29-30, 2014, Minneapolis (MN), USA. DOI: 10.1109/3DUI.2014.6798860 "

[6] Ragaad AlTarawneh, Jens Bauer, Shah Rukh Humayoun, Achim Ebert, and Peter Liggesmeyer ESSAVis++: An Interactive 2Dplus3D Visual Environment to Help Engineers in Understanding the Safety Aspects of Embedded System. In proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems (EICS 2014), pp. 201-204, ACM, 2014.

[7] Ragaad Altarawneh, Jens Bauer, Patric Keller, Achim Ebert ESSAVis: a 2Dplus3D visual platform for speeding up the maintenance process of embedded systems. In Proceedings of the 27th International BCS Human Computer Interaction Conference (BCS-HCI '13), Steve Love, Kate Hone, and Tom McEwan (Eds.). British Computer Society, Swinton, UK, UK, 2013, Article 43 , 6 pages.

[8] Jens Bauer, Achim Ebert, Oliver Kreylos, Bernd Hamann Marking Menus for Eyes-Free Interaction Using Smart Phones and Tablets, In Availability, Reliability, and Security in Information Systems and HCI (IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Conference, CD-ARES 2013, Regensburg, Germany, September 2-6, 2013. Proceedings). pp 481-494

[9] Jens Bauer, Achim Ebert, Oliver Kreylos, Bernd Hamann Generalized eyes-free interaction for use with large displays, In: Humayoun, S.R., Hess, S. and Ebert, A., eds., Electronic Proceedings of Workshop on Prototyping to Support the Interaction Designing in Mobile Application Development (PID-MAD 2013), 4 pages.

[10] Ragaad AlTarawneh, Jens Bauer, Shah Rukh Humayoun, Patric Keller, and Achim Ebert The Extended Stereoscopic Highlighting Technique For Node-Link Diagrams: An Empirical Study. The 14th IASTED International Conference on Computer Graphics and Imaging (CGIM 2013), February 12 – 14, 2013 Innsbruck, Austria

[11] Jens Bauer, Sebastian Thelen, Achim Ebert Evaluation of Mobile Phones for Large Display Interaction. In: Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011, pp 103-112

[12] Ragaad AlTarawneh, Jens Bauer, Patric Keller, Achim Ebert, Peter Liggesmeyer ESSAVis: A Framework to Visualize Safety Aspects in Embedded Systems, Proceedings of SIGRAD 2012, Interactive Visual Analysis of Data, November 29–30, 2012, Växjö, Sweden

[13] Bauer, J., Thelen, S., and Ebert, A. Poster: Evaluation of Large Display Interaction Using Smart Phones, IEEE Conference on Visual Analytics Science and Technology (VAST), VisWeek 2011, pp. 265-266, IEEE, 2011

[14] Bauer, J., Thelen, S., and Ebert, A. Using Smart Phones for Large-Display Interaction. In Proceedings of 2nd International Conference on User Science and Engineering, 2011

Appendix A

Curriculum Vitae

Personal Information

Name	Jens Bauer
Place of Birth	Landau
Nationality	German

Education

11/2012–present	PhD student at the Department of Computer Science, Computer Graphics and HCI Group, University of Kaiserslautern, Germany
12/2010–11/2012	Qualification Studies for PhD admission
04/2002–09/2010	Study of Industrial Engineering (Minor: Computer Science) at University of Kaiserslautern, Germany Degree: Diploma
07/1990–05/1999	Ottfried-von-Weissenburg-Gymnasium Dahn (Degree: Abitur)

Work Experience

09/2006–01/2007	Research Assistant at University of Kaiserslautern
02/2007–11/2008	Research Assistant at Fraunhofer IESE

Appendix B

Declaration

I hereby declare that my Ph.D. thesis entitled *Large Display Interaction Using Mobile Devices*, is an independent piece of research compiled by under the supervision of apl. Prof. Dr. Achim Ebert and Prof. Dr. Hans Hagen, Department of Computer Graphics and HCI, University of Kaiserslautern, Germany and Prof. Dr. Bernd Hamann, Institute for Data Analysis and Visualization, University of California Davis, USA.

This thesis has not been previously submitted for the award of any degree, diplome, associateship, fellowship or its equivalent to any other university or institution. Throughout this thesis, sources of information have been credited to the best of my abilities and knowledge.

Kaiserslautern, 26.5.2015

Ort, Datum

Jens Bauer