Implementing a Wizard of Oz Tool for Augmented Reality

Mikael Möller & Per Sörbris

Master's Thesis

Department of Design Sciences
Lund University
ISRN:LUTMDN/TMAT-5163-SE

EAT 2012

Abstract

This thesis aims to explore Wizard of Oz testing in conjunction with Augmented Reality (AR) and focus has been put on testing AR with Head Mounted Displays. The recent increase of interest in HMDs with products such as MOD Live from Recon Instruments and Google's Project Glass puts new demands and possibilities on human-computer interaction. Since the commercial market for HMDs is still in its infancy the need to explore different design approaches is very much present. One way to conduct experiments on human-machine interaction is with the help of a Wizard of Oz tool.

During the thesis we have developed such a tool to support designers in researching usability and interaction. The tool provides a user friendly framework to carry out user case studies focused on AR with HMDs. After input and feedback from stakeholders and experts we believe that, even though the tool is mainly meant to be used in conjunction with AR in HMDs, the tool can be applied to other areas as well.

Keywords
Augmented Reality, Head Mounted Displays, Visor, Wizard of Oz testing, Android

Acknowledgements

We would like to send our gratitude to our supervisors Klas Hermodsson and Günter Alce at Sony Mobile Communications for their constant support and invaluable feedback. We would also like to thank the rest of the Technology Research Department for providing hardware, help and pleasant friday-"fika"s.

Furthermore we would like to thank Joakim Eriksson, our supervisor at the Department of Design Sciences at Lund University, for his guidance and input during the project. We would also like to thank Charlotte Magnusson, Associate Professor at the Department of Design Sciences Lund University, for your valuable comments on our work.

Thank you to our families, friends and acquaintances that have followed our work, you have been very supportive.

Last but not least a big thank you to Tom Carlsson for the coffee breaks and participating as a puppet user in our simulated test.

/ Mikael & Per

Lund, November 2012

Table of Contents

1 Introduction **1**
- 1.1 Purpose and goal . . . 1
- 1.2 Tool setup . . . 2
- 1.3 Scenario . . . 2
- 1.4 Stakeholders . . . 3
- 1.5 Limiting factors . . . 4

2 Concepts of Interest **5**
- 2.1 Augmented reality . . . 5
- 2.2 Wizard of oz testing . . . 6

3 Augmented Reality Technologies **7**
- 3.1 Smartphone . . . 7
- 3.2 Head mounted displays . . . 9
- 3.3 Marker tracking . . . 10
- 3.4 Markerless tracking . . . 11
- 3.5 Gestures . . . 11
- 3.6 Facial recognition . . . 12
- 3.7 Speech recognition . . . 13

4 Methodology **15**
- 4.1 Literature review . . . 15
- 4.2 Agile software development . . . 15
- 4.3 Requirement specification . . . 16
- 4.4 Tools . . . 18

5 Design and Development **19**
- 5.1 Connection . . . 19
- 5.2 Puppet . . . 20
- 5.3 Filebrowser view . . . 21

5.4 SmartWatch extension 21
5.5 Navigation view 22
5.6 Camera view 23
5.7 Puppet view 24
5.8 Notification view 24
5.9 Tour view 26
5.10 Predefined sequence view 26
5.11 Settings view 27
5.12 Log 27

6 Tool Evaluation **29**

7 Discussion **31**
7.1 Technical discussion 31
7.2 Design discussion 32
7.3 Usage discussion 37
7.4 Future work 38

8 Conclusions **41**

9 Definitions, acronyms and abbreviations **43**

References **45**

A Comments **51**
A.1 Klas Hermodsson 51
A.2 Günter Alce 51
A.3 Charlotte Magnusson 52

B Requirements **55**
B.1 Requirements Matrix 55
B.2 Requirements Elicitation Matrix 61

List of Figures

1.1 An example of our WOz setup 2

2.1 Milgram's virtuality continuum 5

3.1 Android System architecture 8
3.2 Image illustrating the difference between video and optical see-through 9
3.3 Evolution of Steve Mann's HMDs 10
3.4 A collections of HMDs 11

4.1 Kolb's circle of learning 15

5.1 An image displayed over the camera preview on the Puppet device(video see-through). 20
5.2 Screenshot from the filebrowser view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 21
5.3 The puppet screen showing that it is ready to receive a voice command (left). The puppet loading screen showing that it is processing the command (right), i.e. the wizard decides what happens next. 22
5.4 Screenshot from the navigation view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 23
5.5 Screenshot from the camera view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 23
5.6 Screenshot from the puppet view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 24
5.7 Screenshot from the notification view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 25
5.8 Displaying a small notification (left) and a big notification (right), in this case SMS, on the puppet device. 25
5.9 Screenshot from the tour view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 26

5.10 Screenshot from the predefined sequence view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) . . 27
5.11 Screenshot from the settings view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S) 28

7.1 Screenshot of the tour view in editing mode 34
7.2 Screenshot of the tour view in tour mode enabled 35
7.3 Screenshot from the notification view showing the add custom notification section 35
7.4 Displaying the spinner pop-ups 36

Chapter 1
Introduction

Head Mounted Displays (HMD) with Augmented Reality (AR) is under research and development. Smartphone manufacturer *Sony Mobile Communications* is one of many corporations interested in its development and does research in the field e.g. hosted master thesis projects on AR related interaction design (*Social AR* [1] & *Visor Concept* [2]). Sony is also part of the European Union funded project VENTURI aimed at improving AR experiences and development. Alongside with development of the hardware, interaction designers have a need to research usability and develop a user interface. How could this be arranged? One approach is to use a technique called Wizard of Oz testing, that allows for evaluation of interaction design even when the product being tested is not yet developed. Wizard of Oz testing can simulate missing functionality, both hardware and software related, to enable early feedback on the direction of the work. The focus of our thesis is to implement such a Wizard of Oz (WOz) tool for use on an Android device.

1.1 Purpose and goal

The overall goal of this master thesis is to investigate the use of Wizard of Oz testing with Augmented Reality. Our work will focus on the use of Head Mounted Displays (HMDs) and a tool for Wizard of Oz testing will be created. The WOz tool will provide a user friendly experience while executing user case studies on Augmented Reality in Head Mounted Displays. The thesis aims at providing following features in the tool:

- Present media such as image, video and sound
- Easy navigation and location based triggering
- Capability to log test results and visual feedback

- User friendly UI and Sony SmartWatch [3] interaction

1.2 Tool setup

The Wizard of Oz setup for the WOz tool is shown in Figure 1.1. The *puppet device* is connected to the HMD and is used by the *puppet user*. The puppet device communicates via WiFi with the *wizard device*. The wizard device is operated by the human *wizard operator* who can control the puppet device. The puppet user is able to interact with a wrist worn SmartWatch which is connected to the wizard device via Bluetooth.

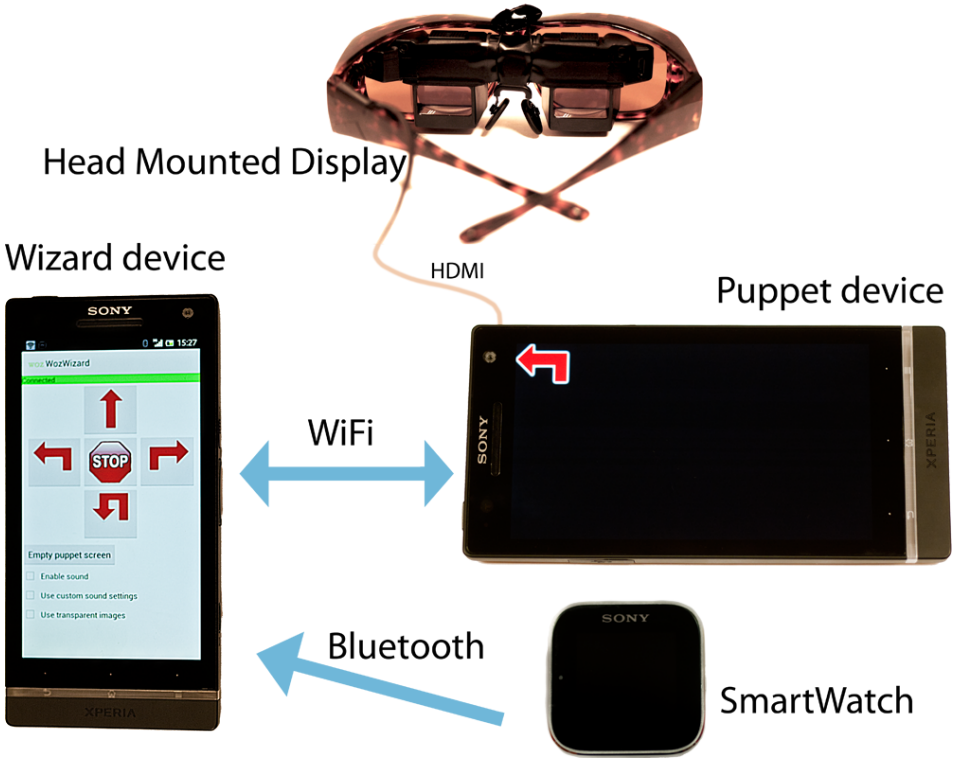

Figure 1.1: An example of our WOz setup

1.3 Scenario

A typical setting in which the tool can be useful is when a design planner has created a user interface for an augmented reality product early in the development. Since the project is still in its early stages all functionality

required to support a test of the design is not yet in place. To be able to still test the user interface the planner can use a Wizard of Oz tool. With the help of a WOz tool the planner can prepare a test for the user interface even though the functionality is not yet fully implemented. The test could follow these steps:

Planning The planner determines what to test and decides what functionality to fake. The planner also decides when events will trigger during the test.

Preparation After the planner has decided what part of the functionality to test, the planner can start to create the test by preparing the tool with the required resources such as images, sounds and video. When the resources needed have been created, the planner can bind the resources to the desired triggers, for example, GPS coordinates or place them at a certain order in the list of commands to run.

Execution The puppet users role is to use the system while the wizard operator controls the experience. If the planner chooses to use someone else as wizard operator the planner can take more of a monitoring role. This allows the planner to oversee the process rather than focusing all of his attention on the puppet user.

Processing When the test has been executed the planner process the test results with help from notes, images and logs provided by the WOz tool etc.

1.4 Stakeholders

Our stakeholders are not our customers, they are partners and recipients who have interest in the product we developed. We were not bound to follow their every directive or wishes, but we highly valuated their opinions and ideas.

1.4.1 Sony Mobile Communications

Sony Mobile Communications is a global mobile phone manufacturer and their smartphones and tablets are using Android. This master thesis is an initiative from Sony and our connections to the corporation are:

- Klas Hermodsson (supervisor)
- Günter Alce (assistant supervisor)

Sony Mobile Communications is the main stakeholder of the thesis.

1.4.2 VENTURI project

VENTURI stands for *immersiVe ENhancemenT of User-woRld Interactions* and the VENTURI project is a European Union founded project in the 7th Framework Program with eight participating partners [4]. VENTURI's purpose is to improve and provide means to create a experience,*" a fully integrated, immersive, easy to use, truly mobile and high performing AR-based enviroment"*. The project stretches over three years and focus on an AR indoor gaming scenario, an indoor personal-assistant/navigator and an outdoor tourist guide and museum demonstrater.

VENTURI is more of a recipient than a stakeholder. Sony is interested using the WOz tool for research and prototyping in the VENTURI project. The intent to apply the tool in the VENTURI project was introduced half way through the project.

1.5 Limiting factors

The foremost limiting factor was time. Since we only had twenty weeks to do the Master Thesis, including the report, we were not able to add all functionality we and our stakeholders might have wanted to. Since there was no way to circumvent this we could only mitigate the problem by letting our stakeholders rank what functionality to implement in order of importance. The design and development time in the thesis was thirteen weeks.

Another limiting factor is the hardware and software libraries on the puppet device. Since the purpose of the tool is to test interaction on unfinished products it is safe to assume that the device being tested might lack libraries or sensors otherwise considered standard parts of an Android device. Factors that have limited us are availability of sensors, camera, Voice-to-Text, Text-to-Speech etc. To circumvent the problems arising from this, as much functionality as possible resides on the wizard device since we can assume that the choice of wizard device is more open considering it is not the device being tested.

Chapter 2
Concepts of Interest

2.1 Augmented reality

Augmented reality (AR) is related to Virtual Environment (VE), or more commonly Virtual Reality (VR), but as where in VR the user is put in an entirely virtual world, AR superimposes virtual objects on the real world and supplements it. The relationship (see Figure 2.1 [5]) is often described with the help of a *Virtuality Continuum* where *reality* is found on one side and *virtual reality* on the other and in between there exists various degrees of Mixed Reality(MR) [6].

"AR can be thought of as the "middle ground" between VE (completely synthetic) and telepresence (completely real)" [7]

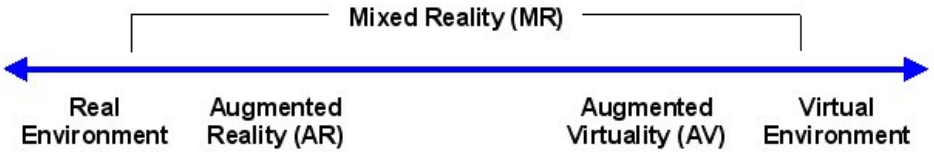

Figure 2.1: Milgram's virtuality continuum

Augmented reality is defined by R.T. Azuma as a system with the following three characteristics: [7]

1. Combine real and virtual
2. Interactive in real time
3. Registered in 3-D

In 1965, Ivan E. Sutherland stated that the ultimate display would be *"a room within which the computer can control the existence of matter"* [8]. Sutherland's thoughts about computers not having to follow the ordinary rules

of physical reality and users interacting with the VE, led him to create the first HMD (Head Mounted Display) which is considered by many the birth of AR [9]. AR can be used as a tool to help simplify tasks that would otherwise be hard to do. Starner et al [10] for example, present a system with a Remembrance Agent that, among other things, collects information for the user to be presented later when needed. Information collected can also be made public so other people can ask the system for information.

2.2 Wizard of oz testing

Wizard of Oz Testing, alternatively Wizard of Oz experiment or OZ paradigm is a method for testing a non complete system on users with the help of a human *wizard* who acts as the system where the functionality is missing. It was initially developed by J.F. Kelley in 1983 to develop a natural language application [11]. The task of the wizard varies between different implementations [12]. In one end we have the supervisor who observes the system and acts as a safeguard and if the need arises intervenes. On the other side of the spectrum we have the controller. The controller takes the role of a missing part of the system or the entire system and emulates it. The role of the wizard can also change depending on the phase of the development. When more and more functionality is added the role of the wizard goes from being a controller to a supervisor. Preferably the tools used to create the user experience should be the same as the ones used to create the wizard interface to streamline an iterative WOz testing process throughout development [13]. Wizard of Oz testing is a powerful tool for uncovering design ideas in limitedly evolved technologies, especially for systems perfoming in physical enviroments, since the designers are less constrained by technical specifications [14] [13]. It is very important that a WOz interface is supporting the wizard in performing their assigned tasks by observing the user, observing the environment, simulating sensors and/or controlling content [13]. The tool should be useful to the dedicated wizard operator and could, if possible, have automated functionality. S. Dow et al. opinion is that Wizard of Oz testing is *"leading to more frequent testing of design ideas and, hopefully, to better end-user experiences"*.

Chapter 3
Augmented Reality Technologies

This chapter will cover some of the AR technologies and related equipment available today. The covered areas are mostly those that are in relation to our work, such as smartphone, HMD, gestures, speech recognition.

3.1 Smartphone

There is no exact definition of what separates a *smartphone* from a feature phone but generally the smartphone is an open platform that can be extended by installing new software and feature phones are a closed platform with a predefined set of functionality [15]. The smartphone is usually equipped with more processing power as well and provides advanced capabilities in computing and connectivity.

Smartphone users use their phones in many aspects of their daily life. Besides from calling, texting and e-mailing smartphone users, through applications, search the Internet, navigate, handle bank services, play games, use social communities and so forth. Some applications support Mixed Reality (MR) activities but it is up to the user to decide when and where to integrate data. E. Barba, B. MacIntyre and E.D. Mynatt state that *"we are fast approaching the limits of this model both in terms of the cognitive capacity of users to context shift, but also in terms of what kinds of MR experiences this model can support"* [16]. Today's usage of smartphones with MR support requires the user to actively stop, pull up the phone, unlock it and initiate the MR activity. New and more comfortable interaction possibilities will have to evolve and research and development should be focused on capabilities in Augmented Reality and Mixed Reality technologies. The next, most important, step to take is according to Barba, MacIntyre and Mynatt: *"new forms of display(such as head-worn displays paired with a user's smartphone) must be developed".*

3.1.1 Android

The use of Android in this project came naturally, primarily from the work being done at Sony Mobile Communication who utilizes Android in their phones and tablets and secondly from both of us having some prior experience with the platform. Android is an open source software stack for mobile devices that provides an operating system, middleware and applications [17]. At the bottom, see Figure 3.1 [18], a Linux kernel is used as a hardware abstraction layer [19]. The kernel provides processes, security and a network stack. It also provides a model for hardware manufacturers to write drivers to. The kernel has been patched to better match the demands on a mobile device. This includes support for sensors not regularly found on desktop machines and laptops such as gyroscopes and GPS. It also includes optimizations aimed to improve performance on mobile devices such as improved power management. Above the kernel you have the native libraries written in C/C++. This part contains a custom *libc* implementation optimized for mobile devices. This section also contains the Surface Manager responsible for combining all surfaces into a frame buffer. Above that you have the Android runtime. This contains the *Dalvik Virtual Machine* and the core java libraries. All libraries available in Java are not present in Android.

Android was unveiled in 2007 and has since its inception powered various devices with different sizes and hardware [20].

3.2 Head mounted displays

A head mounted display is just as the name says a display worn on the head. It can have monocular, biocular or stereoscopic image presentation [21]. It can also have either optical see-through or video see-through (see Figure 3.2). In video see-through the user is presented with a video of the real world with augmentations applied on top of a video of the real world whereas with optical see through the HMD present the image on top of the real world directly [7].

HMDs come in many shapes and sizes. They vary from large robust military versions to smaller monocular versions like the Google Glass [22]. One of the pioneers in the field is Steve Mann and Figure 3.3 [23] illustrates the evolution of his work with HMDs under a 20 year period.

HMD received a lot of attention in the early 90's, but the interest did not last [24]. However recent progress in HMD technology have attracted much attention again. Manufacturers have now proposed lightweight HMDs that are only a few millimeters thick which could have a commer-

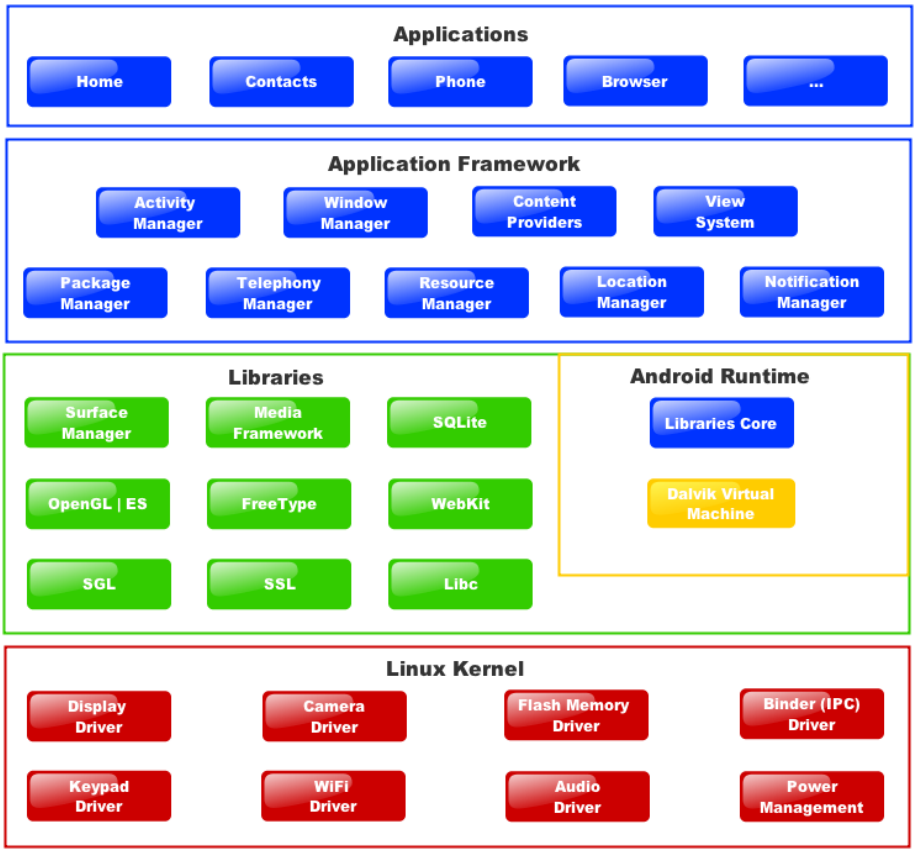

Figure 3.1: Android System architecture

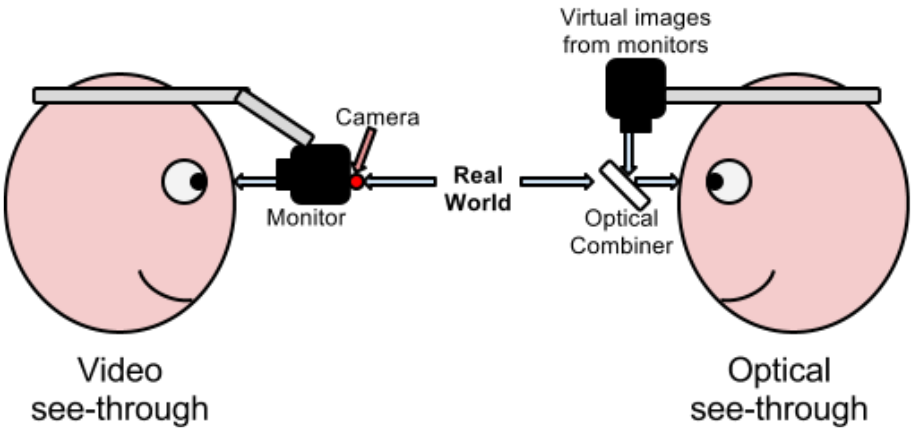

Figure 3.2: Image illustrating the difference between video and optical see-through

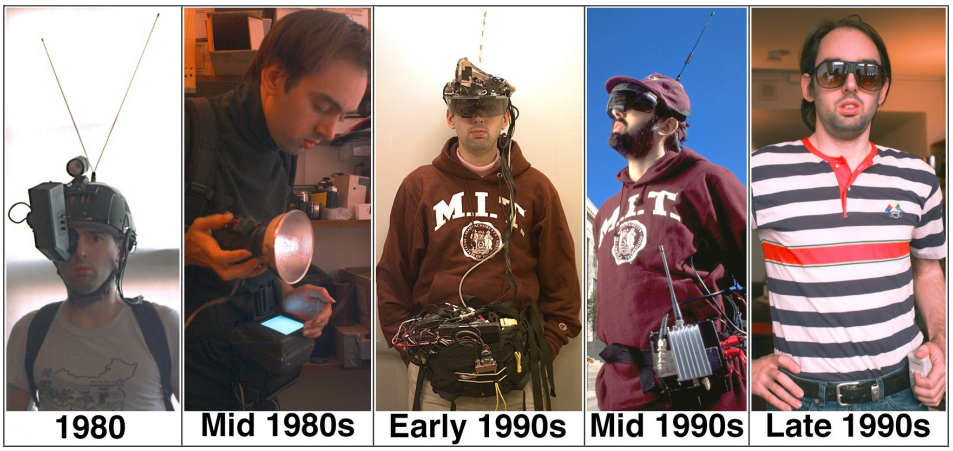

Figure 3.3: Evolution of Steve Mann's HMDs

cial interest. There are still some issues that needs to be resolved in order for a HMD commercial market to surface though. First there are some technical aspects that needs to be handled such as brightness, contrast and battery life [25]. However since the device is meant to be used outside in public the social aspect has to be taken in to consideration as well [26]. The design has to be non-intrusive and unostentatious to allow everyday use, e.g. size and contemporary spectacle design. Lastly the price of the products must become affordable to a bigger crowd.

HMDs could also be used to help people with impaired vision. They could, for example, be used to magnify certain points of the field of view to help with reading or they could be used to warp certain areas to circumvent blind spots [10].

There are several companies that are developing and researching in the area of Head Mounted Displays. For example:

Vuzix Manufactures portable high quality wearable displays for virtual and augmented reality [27]. Vuzix corporation is the leading supplier of video eyewear products in the consumer, commercial and entertainment market. We have partially used one of the older see-through models from Vuzix, the Vuzix STAR 1200 revision 1, mounted on a pair of sunglasses (see Figure 3.4).

Google Glass Google's Project Glass was presented and demonstrated on Google I/O 2012. The functionality presented on Google I/O was recording and streaming video. A development version of the phone is to be released during 2013. Currently the technical specifications of the product are not known but it appears to be a monocular device with a camera (see Figure 3.4 [28]). A concept video has been released but how much of the content presented in the video that will actually be reality is impossible to say [29].

Recon Instruments Recon instruments manufactures displays for use on the inside of ski goggles (see Figure 3.4 [30] [31]) [32]. The device contains, among other things, GPS, accelerometer, gyroscope, thermometer and bluetooth connectivity. The user interacts with the application with the help of a watch-like bluetooth remote. Recon Instruments works with many major ski goggle manufacturers.

3.3 Marker tracking

Marker tracking uses fiducial markers to track a point in the field of view to overlay a virtual object. The markers increase speed and precision

Figure 3.4: A collections of HMDs

but on the other hand they require markers which might not always be wanted [33]. There is also no standard for marker tracking so there is a big risk that each application requires its own distinct marker [34].

There are quite a few marker tracking applications available currently, the following two , for example, are available in the Google play store: Nerdherder and AR Defender.

3.4 Markerless tracking

A device using *Markerless tracking* does not use *fiducial markers* to determine its position in relation to the surroundings [35]. Instead it uses either *sensors* or *natural features* of the surrounding to establish the direction and position of the device. Visual systems can generally be divided into two different subgroups, *corner detection* and *blob detection*. Corner detection uses corners in the surrounding to track the devices movement and position while blob detectors tracks image regions with similar color intensity. Generally corner detection is superior in precision but blob detection is better at scaling.

3.5 Gestures

Gestures is a way of communicating with your body without using verbal communication. Because of advances in size, power and cost of computer hardware it is now possible to distinguish and recognise swipes, flicks, body movements, hand gestures and even finger movements [36]. This will, according to Norman, lead to that *"gestures will find an appropriate place in the repertoire of interaction systems"* and be a valuable addition of interaction techniques. But it will not be established effortlessly since gestures as a language is globally nonstandard that can create confusing meetings when cultures intermix. For example are simple headshakes and hand-waving gestures for *hello, goodbye* and *come here* are performed differently in different cultures. Therefore standard conventions must be developed in order for gestures to mean the same thing in different systems. It is not all ways easy to learn gestures and standardization will entail a learning curve. But our opinion is that children of today will have an advantage in adding gestures to interact with future system. Many children are all ready using gestures as a tool for computer interacting via video games. Products as Nintendo Wii [37] and Microsoft Kinect [38] are using gestural interaction for gaming experience and are very popular and globally spread. Old people and people with cognitive impairment might be the users with the most difficulties of applying this standard of interaction.

SixthSense Pranav Mistry et al. are researching about gestural interactions. The WUW (Wear ur world), also known as *SixthSense* [39], is a wearable gestural interface which consist of a camera and a small projector [40]. The hardware is mounted on a hat and connected to e.g. a smartphone. The software sees what the user sees through the camera and the projector can visually augment surfaces or physical objects. The user can then interact with the project information through hand gestures, arm movements and object manipulation by tracking color marked fingertips. Mistry, Maes and Chang believe that SixthSense could replace system interaction through hardware resulting in that the hardware components of SixthSense can be miniaturized without usability concerns. An interesting thought is that this could entail manufacturers to create products, like smartphones, without a lower limit in size regarding to interaction. Leaving the components and power supply to determine the size of the device while outsourcing interaction to products like SixthSense, a HMD or even a combination of them.

OmniTouch Another system that in many ways is similar to SixthSense is *OmniTouch* [41]. But unlike SixthSense it does not require the user

to color code the fingers. Just as SixthSense it uses a pico projector to augment the surroundings. Instead of the regular camera used by SixthSense it uses a depth camera. This difference allows OmniTouch to differ between a finger hovering over a surface and an actual click.

3.6 Facial recognition

As the name states, face recognition is used to recognise faces in images. The technique is fairly new and was first developed during the 1960s. In the beginning the process required an operator to manually identify features and it was not until 1991 that a method for automatically detecting faces in images was discovered [42]. The method for automatic face detection was developed by Mathew Turk and Alex Pentland and made use of what they named *Eigenfaces*. Eigenfaces are the eigenvectors of a set of faces [43]. With this method it was possible to determine if an image contained a face without the use of an operator in near real time.

Since then the increase in computational power have made it possible to do real time face recognition on mobile devices and Android contains a library for real time face detection(Package: android.hardware.Camera.Face) from API level 14 and up and face detection in images (Package: android.media.FaceDetector) from API level 1 [44].

Facial recognition can be used in augmented reality to present information about people around us. For example, the interactions described by Övferholm and Petrén [1] make use of facial recognition to visualise icons around a person. The icons are connected to information and social media related to the person. Another example is the TAT augmented ID concept video [45].

Face recognition is not only interesting from an augmented reality stand point. Much research in the area if not most comes from the perspective of using it as a biometric similar to fingerprints or an eye scan. The first usage that captured the public was a trial at the 2001 Super Bowl where surveillance images where matched against mugshots [42].

Face.com The Israeli founded company Face.com is operating the largest cloud based face recognition platform [46] [47]. In 2011 users and developers were able to scan 5,000 photos per hour, free of charge. The face recognition technology has improved significantly and Face.com's facial recognition algorithm has a high recognition rate of 91.3% [48]. From June 2012, Face.com is owned by Facebook.

Viewdle Viewdle is an Ukrainian founded, now based in Silicon Valley,

facial recognition company which is leading in imaging and gesture recognition [49]. Viewdle enables new user experiences in e.g. mobile interaction, augmented reality and social networking on Smartphones/Tablets through face, object and gesture recognition [50]. In October 2012 the augmented reality and image recognition firm was acquired by Google's Motorola Mobility unit.

The fact that large corporations like Facebook, Google and also Apple [51] acquire smaller facial recognition companies seems to indicate that the facial recognition market might be growing to be a profitable market.

3.7 Speech recognition

The way we interact with computers is changing with the increase of wearable computers. This change requires different methods of interaction since the usual tools might not always be available, such as keyboards and mouses. One form of interaction that naturally comes to mind is to use ones voice. The technology can be divided in to two separate parts, *speech engine* and *interpreter*. The speech engine is the software that transforms the speech to text. The interpreter then analyses the words and context to generate a response. An example is Apple's Siri application which tries to emulate an intelligent personal assistant [52]. Apple is using the *Nuance communications* speech engine to get plain text, from which the Siri application then producse a response or acts accordingly to the spoken command [53]. Speech engines and interpreters have evolved fast during the last twenty years. Some of the leading speech recognition related companies of 2012 are [54]:

Google Google's speech engine [55] is fast, accurate and supports a large number of languages.

Microsoft Microsoft's engine, *Tellme* [56], is like Google's engine fast and accurate, but also available on many platforms.

Nuance Communications The market leader is Nuance Communications [57]. The company has great innovation and customer satisfaction along with an engine that is very accurate. Nuance has been a part of the speech recognition business since the 90's, and has recently expanded by buying other companies.

The field of speech recognition technology is highly anchored to Wizard of Oz testing. As stated earlier(chapter 2.2 Wizard of oz testing) the

testing method was developed during the development of a natural language system.

It should be noted that speech interaction is not without its downsides. In some situations it is not possible to use it such as in noisy environments and in situations where noise is frowned upon such as a library or at a presentation.

Chapter 4
Methodology

In this chapter used techniques and development methods will be explained. The extent to which they were used will be stated. Our approach to the work generally followed the style of *Kolb's circle of learning* [58, p. 21] (see Figure 4.1). Basically we chose an idea to work with and implemented, tested and evaluated it. If the evaluation proved positive we continued to improve the idea. In case of a negative response, the result was either modified, reanalyzed or discarded.

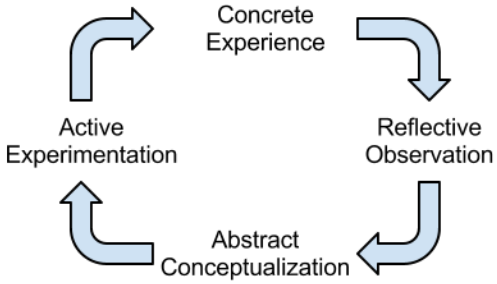

Figure 4.1: Kolb's circle of learning

4.1 Literature review

We started to study the previous thesis in the AR area that had been made at Sony Mobile Communications. Later moving on to search articles on the Internet, from sources such as Google scholar, IEEE Xplore and Summon by Lunds University library.
Keywords used: AR, MR, Wizard of Oz, HMD, Speech recognition, Android

4.2 Agile software development

Many companies have introduced agile software development methods like Scrum, XP etc into their software development process [59]. Kent Beck wrote the *"Embracing Change with Extreme Programming"* in 1999, describing the principles and patterns which should be used in a software development process to increase customer satisfaction, simplicity, efficiency etc [60]. The following principles were used during our development:

Pair programming Has been applied a few times to solve complicated problems together.

Continuous integration New functionality has been integrated to the project as soon as it has been finished.

Refactoring or design improvement At numerous occasions refactoring has been made and design improvement have been applied a few times.

Small releases At the end of every week a release has been made. The APKs for the WozWizard, WozPuppet, WOZSmartWatchExtension have been placed and shared on Google docs.

Coding standards We have strived to use the same styles and standards for the entire source code.

Collective code ownership Both programmers have always been able to add, change or remove functionality in the source code.

Simple design Had the ambition of implementing the simplest solution possible that makes it work.

Sustainable pace and no overtime We have followed the concept of software developers to work no more than 40 hours per week with no overtime.

The customer is always available We worked at the same office as our main stakeholder, it was therefore almost never troublesome to get customer input.

Leave Optimization until last This principle has been followed.

Acceptance tests are run often and the results are published Acceptance tests have been made almost every week, during the development phase, on the weekly meetings. The result have been published in form of feedback or meeting protocols on Google docs.

4.3 Requirement specification

The requirement specification document was highly valuable for the development of the WOz tool. During the implementing stages of the project, we continuously updated the requirements and integrating new requirements. We used an Excel document where the requirements was categorized in Data, Design, Feature, Function, Quality requirements. Furthermore, the requirements were described in detail and dependencies between the requirements were also documented. We time estimated the requirements and then one of the stakeholders, Günter Alce, prioritized them. The elicited requirements and related elicitation method is shown in Appendix B. The elicitation methods used are described in this chapter.

4.3.1 Document studies

A part of the document studies was covered under chapter 4.1 Literature review, but a very important study was made half way into the project. The introduction of VENTURI as a new stakeholder gave us the opportunity of studying their ideas and use cases. In particular two use cases gave us perspective of where the WOz tool could be applied and where Sony Mobile Communication have interest in using the tool. The first one is indoor guidance through arrows with or without voice. The second one is finding a product on a shelf with help of indicators.

4.3.2 Pilot experiments

We carried out some technical prototyping parallel to the document studies. Many small applications, which tested hardware and sensors possibilities and constraints, were implemented. These small applications gave us insight of the APIs and how to use the sensors. Later, when building the WOz tool, these applications were integrated into the system.

4.3.3 Stakeholder analysis

The purpose of the stakeholder analysis was to determine the stakeholders, what goals do they expect and stakeholder contribution [61, p. 339]. A part of the stakeholder analysis process has been made iteratively since a new stakeholder have emerged and goals have changed or surfaced.

4.3.4 Similar companies

A truly great technique to elicit realistic ideas is to see what similar companies or products can do [61, p. 345]. Since the Head Mounted Displays domain is relatively new with limited commercial products released we had to study visions instead of actual products. The concept video on Project Glass from Google Incorporated was a great first approach on the field of HMD which gave us many ideas of future functionality in the domain that could be tested [29].

4.3.5 Brainstorming & bodystorming

Brainstorming was used to a great extent during the start up of the project. We were both familiar with the method so it was the easiest and most natural way to start eliciting requirements. Brainstorming was applied during the whole project, not only for the requirement elicitation, e.g. design choices and specific implementation problems and solutions. Bodystorming is an extension to brainstorming which can be a very a powerful tool to elicitate requirements related to the environment and interaction design [62]. The technique involves imagining the product in the element which it will be used and from there be able to elicit the interaction requirements and realistic possibilities. It also allows us to explore requirements without having any functionality at all. We applied bodystorming frequently because it was difficult to imagine the complicated setup with glasses, phones and SmartWatch. Using this technique entailed both limitations and possibilities.

4.3.6 Interviewing

During the project there were weekly meetings with our supervisors, Klas Hermodsson and Günter Alce. Since our supervisors also are the major stakeholders, we had a lot of opportunities to get feedback through short interviews and demo sessions affiliated to these meetings. The structure of the meetings were as follows:

- Preparation
- Demonstration of the latest WOz tool version
- Stakeholder tryout
- Feedback and follow-up from the previous demonstration
- Implementation after the meeting

- (Follow-up on the next meeting)

The result of these weekly interviews was that the requirement specification was prone to change. After trying out a feature it was not unusual for its importance to change and in some cases simply be removed.

4.4 Tools

Developer tool

Eclipse an open source Java IDE [63].

Android Developer Tools for Eclipse An Android plugin for the Eclipse IDE [64].

Android Debug Bridge version 1.0.29, a tool for communicating with an Android device connected to the computer. It can be used to install, uninstall, pushing and pulling files, reading debug logs as well as opening a shell on the Android device among other things [65].

Git version control software with focus on being distributed [66].

Mobile device

Sony Xperia S A smartphone by Sony Mobile Communications [67]

Sony Tablet S A tablet by Sony Mobile Communications [68]

Other Hardware

Sony SmartWatch A programmable wristwatch connected to a smartphone via bluetooth [3].

Vuzix STAR 1200 a HMD with optical see-through [69]. The smartphone is connected via a HDMI cable. STAR 1200 comes with a camera, head tracker and sound. These features however lack support for Android so we chose not to attach them on the frame. We only used the HMD as a display.

Chapter 5
Design and Development

This section will describe the implemented features. The first version of the tool was made by Klas Hermodsson at Sony Mobile Communications. It contained a list with folders and filenames. If the user pressed an item in the list the corresponding picture, audio, movie or vibrate trigger was sent to another device. When the other device received the command it would react to it, e.g. show a picture if it could find the file on the SD card. This chapter will describe the new and improved features which were implemented during this thesis. The descriptions of the features in this chapter are a part of the Wizard of Oz Tool on the wizard device, except for when something else is stated.

Android versions supported:

- Wizard: 3.2 and up
- Puppet: 2.3.3 and up

Features that were implemented are in- and outdoor navigator, SmartWatch interaction, puppet device camera streaming etc.

5.1 Connection

The WOz tool communicate via IP over WiFi. Both TCP and UDP are used in the communication. Network communication is handled through two separate services, one using TCP and the other UDP. UDP is only used to transfer the camera feed from the puppet device to the Wizard device. The other service, using TCP, is used for all other communication. The reason we use UDP for the camera feed is that speed is more important than making sure all packages are received.

All communication except for the camera feed uses a protocol generated by protocol buffers over a TCP connection [70].

Since we use wireless communication it is hard to guarantee a stable connection. Instead of trying to guarantee a stable connection, focus was put on being able to recover in case of an unwanted drop in connection.

Another negative side effect of a shaky connection is that the wizard can not really be sure that the connection is up at a given time. To counteract this the wizard device will show its connection status in the top of the screen 5.1. To clarify whether or not the connection is established the field will change color. If the devices are connected the field will be green and red otherwise. To establish whether or not a connection is up the wizard device will send a ping every two seconds. If no answer has been received before the next ping, the wizard will consider itself disconnected. The puppet device does not send out its own pings, only answering the ones it get. If it has not received any pings in a given timeframe it will consider itself disconnected and starts to initiate a new connection.

5.2 Puppet

The user interface of the puppet is designed to not have too much point and click interaction, mainly since the available ways to interact with the HMD can be limited. The puppet user can however interact with the SmartWatch but only indirectly (see chapter 5.4). The only form of point and click interaction is a menu that contains two elements, one button to close the program and one button to enter the IP address of the wizard device. Other than that it is a black background that is replaced by what the wizard wants the puppet to see, example in Figure 5.1. The reason for there being a black background is that when using HMDs with optical see-through black is transparent.

Except for some of the easier voice commands all functionality sits in the wizard device. The SmartWatch for example is paired with the wizard device and the wizard device triggers events on the puppet when it has received events from the SmartWatch. The puppet device communicates with the wizard device on two channels, one thread using UDP for the camera stream and one thread using TCP for commands and sending files. The puppet device can also, if supported by the device, read out text strings, using the Android text to speech library.

5.3 Filebrowser view

The filebrowser was improved with preview pictures to the corresponding filename in the list, see Figure 5.2. Support for requesting missing files on

Figure 5.1: An image displayed over the camera preview on the Puppet device(video see-through).

the puppet file system was also added. To display an image or video on the puppet device the wizard merely selects it.

5.4 SmartWatch extension

A SmartWatch application was implemented to handle interaction with the Sony SmartWatch. The extension is a small program that is placed on the wizard device which is connected through LiveManager and Bluetooth to the SmartWatch. Using control extension gave us the possibility to control all the interactions and the settings for the display. The program is triggered on interaction with the SmartWatch and it will broadcast the interaction that was made. Depending on which state the WOz tool is currently in, it will act in some way or not at all. The interactions that are available and their default functionality:

Touch event No default action.

Long press event Will trigger the puppet to activate the speech recognizer, see Figure 5.3.

Swipe up event No default action.

Swipe left event Clears the puppet screen when in right hand mode.

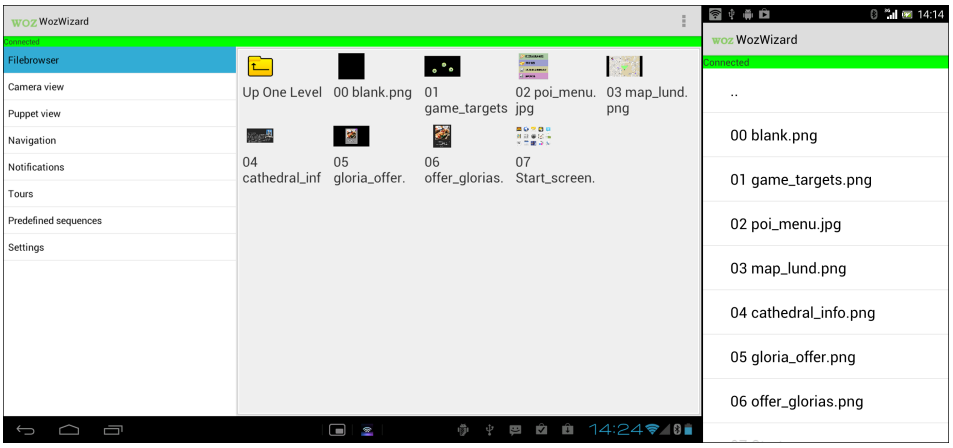

Figure 5.2: Screenshot from the filebrowser view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

Swipe right event Clears the puppet screen when in left hand mode.

Swipe down event No default action.

Pinch event Exit the application on the SmartWatch.

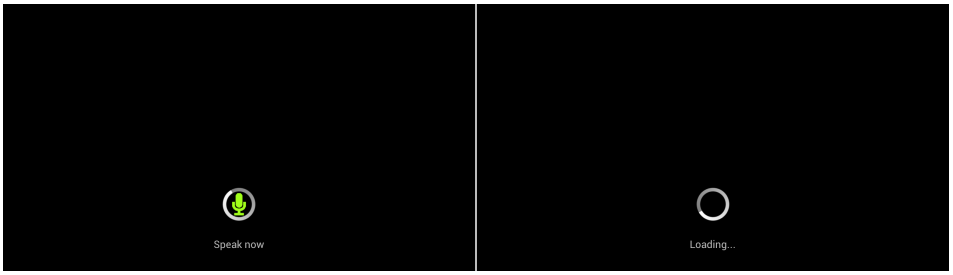

Figure 5.3: The puppet screen showing that it is ready to receive a voice command (left). The puppet loading screen showing that it is processing the command (right), i.e. the wizard decides what happens next.

The SmartWatch will remain in the dimmed state until it receives a pinch event. This is something that should be taken in to account when planning tests since it will drain the battery. The reason for this is that the user should not have to activate the SmartWatch before interacting with it.

5.5 Navigation view

The purpose of the navigation view is to supply the wizard operator with an easy way to fake the GPS navigator, but also to provide a means to navigate indoors where GPS coverage generally is lacking. The view contains buttons for right, left, forward, turn around arrows and a stop button (see Figure 5.4). There is also a button to clear the puppet screen and three check boxes. With the check boxes the wizard operator is able to:

- enable/disable the Android Text-to-Speech engine with the directions
- choose customized Text-to-Speech strings for the directions stated in the Settings view
- show transparent direction pictures. Preferable if the camera is showing on the puppet.

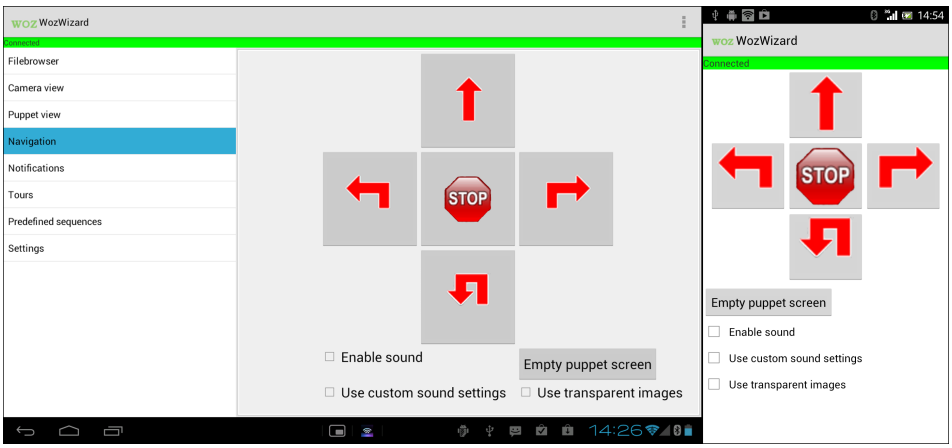

Figure 5.4: Screenshot from the navigation view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

5.6 Camera view

This is a view, shown in Figure 5.5, that handles interaction with the camera on the puppet device. From this view the wizard can start a thread that sends the camera feed to the wizard. This thread uses a separate socket that sends over UDP instead of TCP. In this view it is also possible to start taking pictures on the puppet device automatically at a given

interval. The interval is given in this view. It is also possible to send small image overlays to display over the camera feed from this view. It should however be noted that there is no need to actually display the camera on the puppet device to trigger these images. However it does require that the puppet device is streaming the camera.

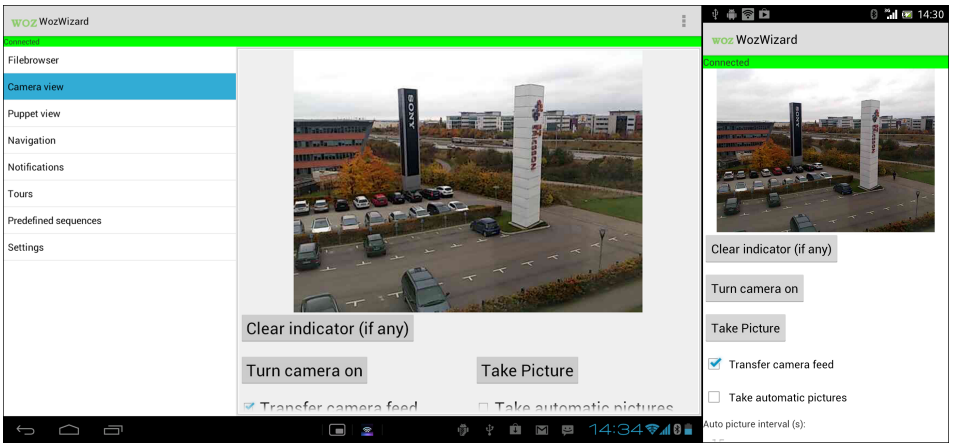

Figure 5.5: Screenshot from the camera view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

If the camera is displayed on the puppet device, pictures can be taken by the puppet user by tapping the SmartWatch (touch event).

5.7 Puppet view

This view allows the wizard to see what the puppet device is currently displaying on the screen, see Figure 5.6. It is a very important part of the tool since it allows the Wizard to confirm or recover from wrongly sent information to the puppet device [71, p. 200].

The view contains a button to request a new screenshot and a button to clear the screen of the puppet device. The update button sends a request to the puppet device for a new screen shot.

5.8 Notification view

In the first version of the tool it was possible to send notifications with the help of pictures, but Günter Alce, requested that support was added for creating new notifications from within the application. From this view (Figure 5.6) is the Wizard able to choose the notification type from ten

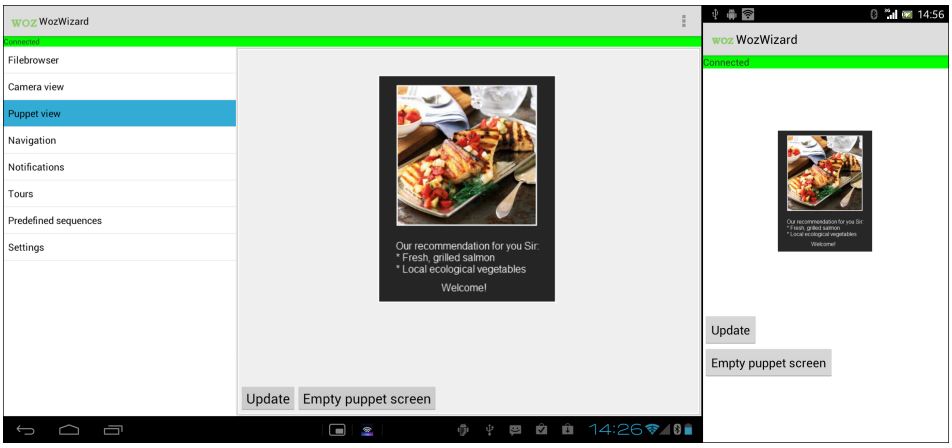

Figure 5.6: Screenshot from the puppet view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

predefined notification types. The tool comes with some example notifications to choose from and the wizard can create custom notifications. The custom notifications can be created from the tool directly or added into files on the SD card.

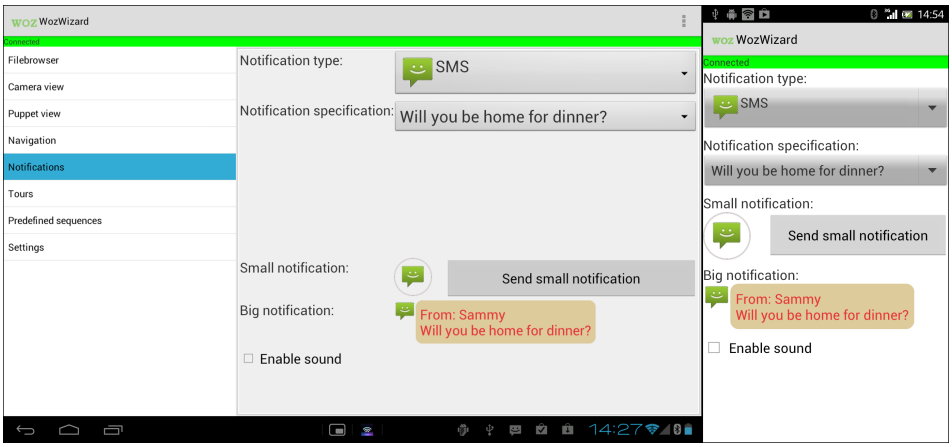

Figure 5.7: Screenshot from the notification view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

Via a button the wizard operator is able to send a small blinking notification to the puppet device to show what kind of incoming notification it is, see Figure 5.8. If the user of the glasses desires to see the notification message, the wizard can send the full notification message by tapping the preview notification message in the Wizard application. The message will

transition in to the screen from the right side and be displayed as in Figure 5.8.

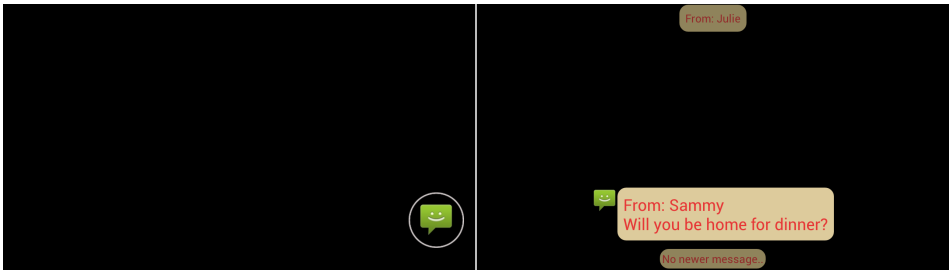

Figure 5.8: Displaying a small notification (left) and a big notification (right), in this case SMS, on the puppet device.

The functionality was later updated when the SmartWatch was introduced. If a small notification is displayed the user of the puppet device can click the SmartWatch to expand the notification or swipe to the right to discard it. When the puppet is showing SMS, older and newer text messages are placed in the top respectively in the bottom of the screen. The user can use the SmartWatch to swipe down to show older messages and swipe up to see more recent messages.

It is also possible to get the notification messages read out with the help of the Android Text-to-Speech engine on the puppet device by checking the *Enable sound* box.

5.9 Tour view

The tour view contains everything related to location triggered actions. The view consists of a map in the top and controls in the bottom, see Figure 5.9. Trigger points will be displayed on the map and if the user selects one the details of that point will be displayed (Figure 5.9). The available controls depends on whether tour mode is active or not. If tour mode is inactive the view contains tools for creating new points. The commands available are the ones used most frequently. Depending on what command is chosen the remaining fields will vary (Figure 5.9) since not all commands use the same attributes. One attribute always available though is the threshold determining how close the puppet needs to be to trigger the command.

If tour mode is active the screen fills with buttons for sending the map to the puppet device and the buttons related to creating new points are removed (Figure 7.2). The tour mode is handled by a separate service to allow the wizard operator to leave the tour view with the tour active in

the background. If the puppet device lacks means to determine its own position the wizard devices position service can be used instead. Whenever a new position is received the coordinates are compared to the list of trigger points with the help of an asynchronous task. If a hit is registered the command will trigger.

Created tours are saved as XML files which easily can be modified outside the application.

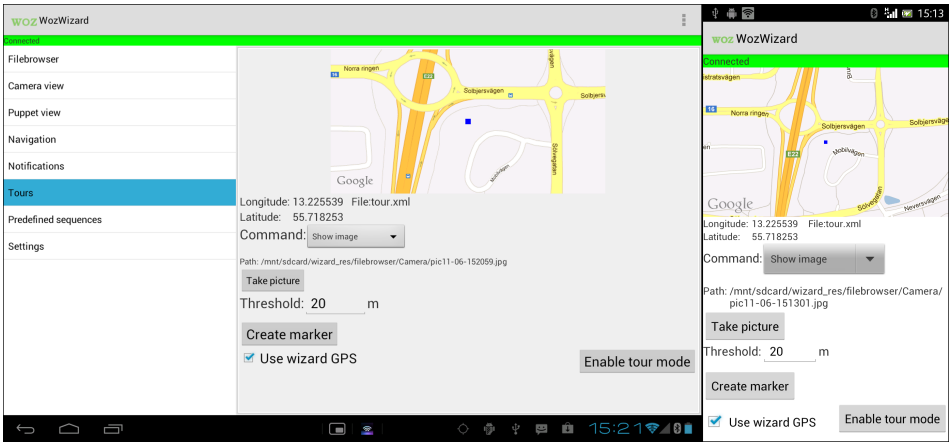

Figure 5.9: Screenshot from the tour view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

5.10 Predefined sequence view

A view for creating easy access to commands likely to be sent to the puppet. The view contains a spinner that allows you to choose a file to parse. The parsed commands are presented in a list. To simplify reading the commands it is possible to give the command an easier name, see Figure 5.10, such as "show blue image". Below the simple description the full command is displayed. The view is meant to be used as a tool to test the same use case multiple times without having to worry about missing a point. The last chosen command will be highlighted so the Wizard operator easily can recall where to resume the sequence. From a spinner the Wizard is able to choose from different predefined sequences located on the SD card. Here each file can contain a different use case and the sequences are saved as XML files.

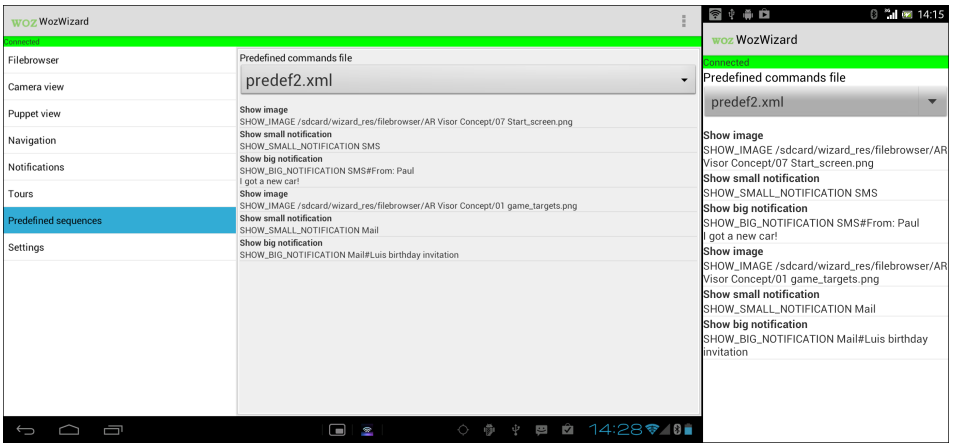

Figure 5.10: Screenshot from the predefined sequence view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

5.11 Settings view

This view allows the user to set a few settings. The wizard should check the check box if the puppet user is wearing the SmartWatch on the right hand (see Figure 5.11). Furthermore, it is possible to customize the sounds for the buttons in the navigation view. The customized sounds are not saved until the wizard operator presses the save button.

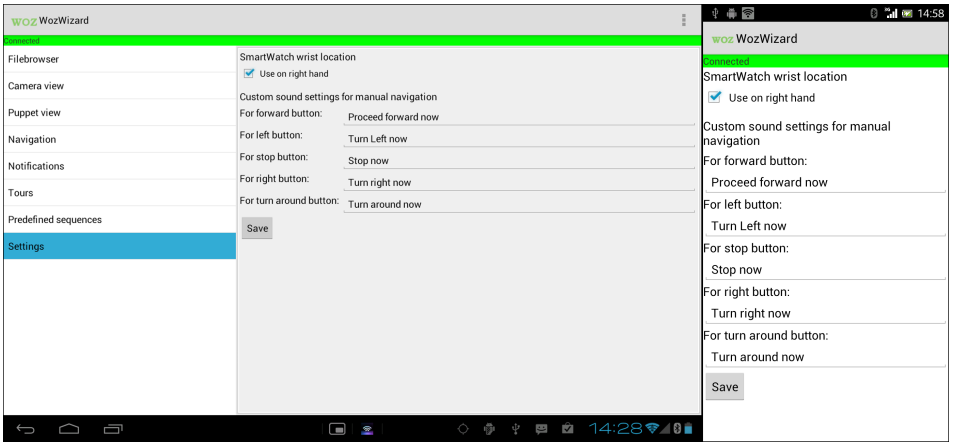

Figure 5.11: Screenshot from the settings view (left side: Sony Tablet S, right side: same screen on a Sony Xperia S)

5.12 Log

Both applications, wizard and puppet, have support for logging the activities of the involved. The logs are saved on the SD card in the *wizard_res* and *puppet_res* folders respectively as *log.txt*. All entries have a timestamp and if the position is known latitude and longitude is also saved. Examples of events that create entries are if a connection has been established, if a command has been received or sent and in some places error messages.

Chapter 6
Tool Evaluation

Klas Hermodsson, who is our main supervisor at Sony, is pleased about what the WOz tool has become. In his comments to us, he states that the feature of streaming the camera is a particularly nice addition and that the stable connection is an important improvement. Regarding the state we deliver the WOz tool, his thoughts are:

> *The current state of the tool is an excellent starting point for conducting tests and design evaluations while allowing for development of new tool features. The fact that this tool is focused on prototyping on the go situations with mobile devices makes it a great match for our research and will allow us to quickly assess user experience of new ideas.*

He explains some further improvements of making the WOz tool more tailored for video see-through in the VENTURI project. However the logging features will be useful in analysing the results of VENTURI related tests. At last he states that the usefulness of the tool and the ability to further develop it is the reason why Sony Mobile Communications will release the WOz tool as open source software. This allows both industry and academia to use and extend the tool. For the full comment see appendix A.1.

Günter Alce, our second supervisor from Sony, is the one we worked closest with during development of the tool. He has been closely involved in the design of the tool since he will very likely use it for the VENTURI project. He comments that the purpose of the tool is to act like a toolbox containing many different features to be able to manage as many different situations as possible.

> *The main goal of this master thesis was to increase the number of new features and make it as flexible as possible.*

Another point he brings up is that we should have done more testing with real test subjects during the development phase. This is something we agree wholeheartedly with. Most test we did during development where only smaller tests going through the various functions to make sure that they worked on their own. One of the most interesting things is that the *predefined sequence* functionality, which initially did not get much attention, is one of the features valued the most. Günters comment can be found in full in appendix A.2

Since Günter Alce and Klas Hermodsson have been in close contact with us during the entire development, we also wanted to get some opinions from someone who was not acquainted with our tool. We asked Charlotte Magnusson, an Associate Professor at the Department of Design Sciences, Lund University, who has experience in the field of Wizard of Oz testing for some input. We had a demo session of the WOz tool with Charlotte and she gave us some very valuable feedback on the tool. Overall she noted that there is a wide potential area of use and comments:

> *"The tool is useful as it is, but would be even more useful with a few improvements"*

She gave us a list of possible improvements and tweaks, some we had already though of and some new, e.g. providing the puppet user voice for the wizard operator and dynamic gesture responses. In the end she stated:

> *"I find the current tool a good starting point, and would like to use it in future projects myself."*

For the full comment see Appendix A.3.

Chapter 7
Discussion

This chapter covers the design choices made. Also covered, is the reasoning leading up to the selections related to interaction, implementation and technical aspects.

7.1 Technical discussion

Network
We chose to use WiFi to connect the devices. There were many reasons for this but mainly it allows us to make use of our previous experience with network programming. Another benefit of this is that theoretically the two devices do not need to be anywhere close to each other as long as the puppet device knows the IP of the Wizard device. Intead of using WiFi we could have used Bluetooth for example but that comes with its own limitations such as bandwidth. As mentioned earlier the devices use both TCP and UDP. Initially only TCP was used and all data transfer was handled by one service. We went with TCP since the nature of the protocol suited our needs. Unlike UDP you initiate a connection and packets are received in the order they are being sent. Also TCP has the benefit that all packets are acknowledged so it is possible to detect if everything has been transfered and received.

The second service, using UDP, was created when we implemented the possibility to stream the camera feed. The main reason for separating this from the regular connection was that this could possibly create a lot of traffic and we wanted the frame rate to be as high as possible. This presented other problems however. Since UDP packets have a maximum size of 65535 bytes and the images fetched from the camera preview where much larger we had to compress the images. This in turn limited the frame rate, although not to the same extent as sending the images unaltered. Unaltered the images where 460800 bytes and compressed they varied between 6000 and 16000 depending on what was filmed. It should

also be noted that even if we sent the image in the original format we would still have to encode it after receiving it on the wizard device.

For communication not related to the camera stream we chose to use *protocol buffers* (protobuf) to generate a protocol. The reason for this was mainly that it is fairly simple to expand if we wish to add more functionality and it allowed us to focus on other parts of the implementation than creating protocols. The protocol generated is also fairly efficient and it can be used to generate code in other languages than Java which can be interesting if somebody would want to port it to another platform [70].

Storage

The tours and sequences are stored as XML files on the external storage. The primary reason is that by using the external storage we can access the files from outside the application. Note that it is not necessarily external storage in the physical sense that is meant. If we had used the internal storage we would not have been able to access the files from outside the application. Considering that one of the purposes of these two functions is to allow the user to plan work ahead on a different device, this made the external storage the best approach. Another reason for using the external storage is that it was already used for images, video and audio files.

The choice of using XML instead of for example protobuf as we use for network communication is that we wanted the files to be human readable and easy to edit, hence the binary protocol created by protobuf was not an alternative. Also there is good support for building XML parsers and writers through the XML package in the java core libraries present already so implementing the functionality was straight forward.

Performance

One key aspect of the work has been to try to keep logic to a bare minimum on the puppet side. This is mostly due to the nature of the application. The tool is meant to be used during development on products that are not finished. Therefore it is safe to assume that the device running the puppet device can have limitations when it comes to supported APIs and sensors etc. As stated earlier we had to divert from this approach with the camera feed but we also did provide a function that manually converts the encoding if the format of the camera preview is not supported by the core libraries. This is a lot slower though. Another example is the location service. The wizard is able to use his own device to determine the location if the puppet device lacks GPS support.

7.2 Design discussion

Fragments
To make the program able to run on as many different screen sizes as possible as well as in both portrait mode and landscape mode we chose to use fragments for the user interface. Using fragments is great when your application needs to be flexible and dynamical in matter of screen sizes and UI design [72]. This approach might take some time to adopt, but when you acquire the knowledge, fragments become powerful building blocks for creating a user interface for multiple devices of varying size and form

UI apperance
We chose a bright theme with high contrast, dark text on white background, for the Wizard device UI instead of using the default dark theme, bright text on dark background. The reason for this was after testing the themes in outdoor sunlight, the environment that the WOz tool will be most likely used in, the bright theme was superior. It was far easier to read and was not as sensitive to sun glare as the dark theme. However in the end, the most significant factor is the mobile device's screen sunlight handling. This can vary quite a lot between different models and many factors come in to play such as luminosity and contrast [73].

Another important thing about the user interface is the placement of items. Generally we have tried to follow the same pattern in tasks. For example, if a task requires the operator to press three buttons in concession they grouped together and placed in the order that they are supposed to be used from top to bottom or left to right.

Connection text
The connection text in the top of the view is present in all layouts. It increases *visibility* in the interface and makes it possible to quickly see the connection state [71, p. 52]. The connection text also uses color coding to make it easier to evaluate the state. The color of the field is red if the devices are not connected. If a connection is established the field changes color to green. The colors used felt like a logical choice considering the two colors use in western society. Red is usually used when something requires attention, be it traffic lights or a warning sign and continuing with the traffic light parallel green means that traffic is allowed. They do however have a downside. Red and green colorblindness is one of the most common forms of color blindness [74]. However since the view also contains text that tells the connection status we did the assessment that people who will not be able to distinguish the colors will still be able to see the

state of the system. There is also the possibility of using different colors but that could also be counterproductive and confusing if the colors are not related to the situation.

Predefined Sequence

The predefined sequence view was mainly designed to allow for easily repeated tests. It allows the user to add short descriptions to the commands to make it easier to distinguish them from each other. This can be useful when many commands of the same type are in succession. It might also not be necessary for a user to know the specifics of a commands to operate it, it might be sufficient with "Right arrow" instead of SHOW_IMAGE and then the path to the specific image for the user to understand. The description is also larger than the text displaying the command in its full to further distinguish them.

Camera view

The camera view was created to be able to interact with the camera on the puppet device. If the camera is being streamed the wizard can click on feed to display an image on the puppet device. This function was created to help with fine navigation and to emulate tracking. At first we wanted to use real face tracking but it was not highly prioritized so we had to drop it because of lack of time.

This view also supports taking automatic pictures at a given interval. This was implemented to make it easier to log tests after a request from our stakeholders. To make it easier to map the images to a certain event they are timestamped.

Tour view

The tour view is a very dynamic view since it changes depending on which state it is in. The earlier versions of the view were very complex because of too many controls and input types. D.A. Norman describes the connection of complexity and controls as follows, *"To make something look like it is easy, minimize the number of controls."* [71, p. 209]. To make the view more user friendly we hide the controls and panels not being used at the moment. This minimized the appearance of complexity and made the view easier to use. In the Figures 7.1 and 7.2 are the expansions and contractions of the controls illustrated.

Navigation view

The point of this view is not to add new functionality, and initially it did not. The point of this view is to create easy access to the images related to navigation. The same functionality of the view can be achieved by se-

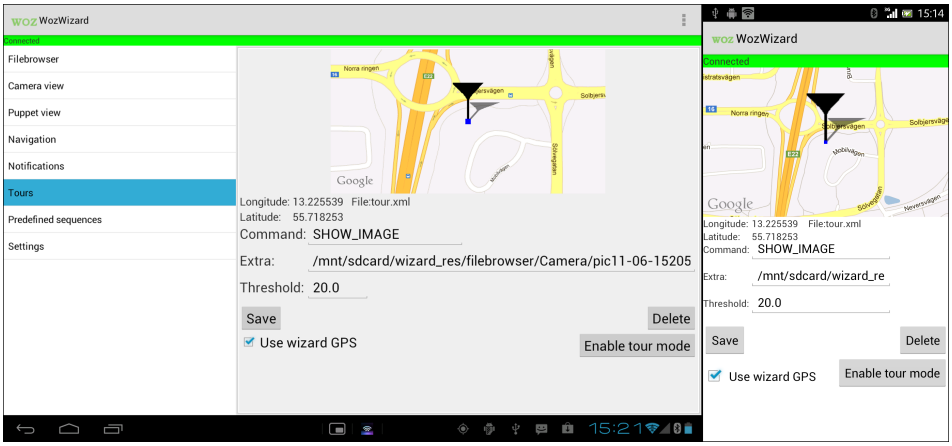

Figure 7.1: Screenshot of the tour view in editing mode

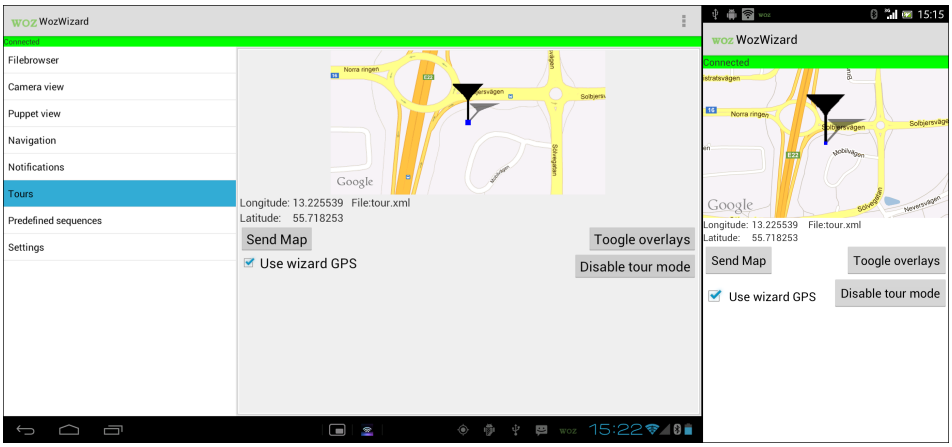

Figure 7.2: Screenshot of the tour view in tour mode enabled

lecting pictures in the filebrowser or the predefined sequence just as well but requires more work from the user. But the view groups and maps the buttons for directions in a standard and logic way, which makes the user experience much easier. The reality of having buttons containing previews of the signs facilitates the user to understand [71, p. 199].

Notification view

The notification view is also dynamic. If the Wizard user wants to create a new notification, one can choose the custom choice in the *"Notification specification"* and the extra controls will appear, see Figure 7.3 (see Figure 5.7 for difference). If a custom created notification is chosen from the list, a "delete custom notification" will appear in same area as *"Save notification"* is shown.

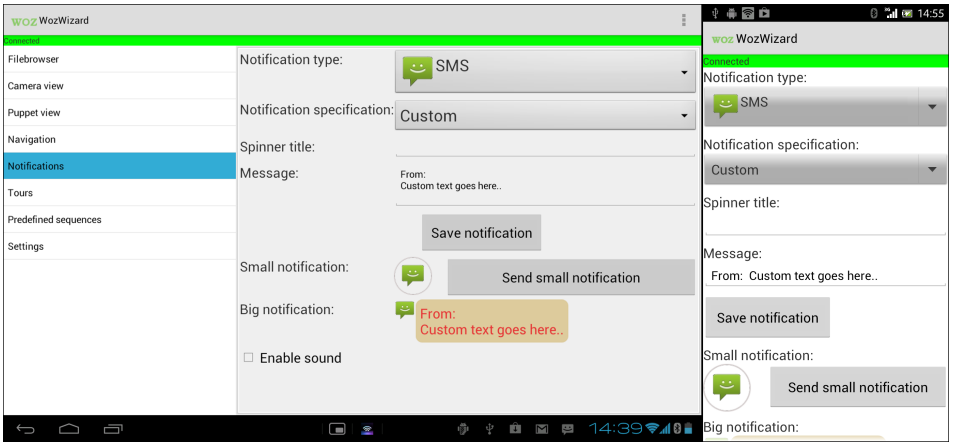

Figure 7.3: Screenshot from the notification view showing the add custom notification section

The stakeholder wanted the action of pressing the big notification to show the notification on the puppet. Instinctively was this action greater than pressing a former placed button *"Send big notification"* in the UI.

We tweaked the spinners to pop up when focused and also added icons in the notification type spinner (see Figure 7.4). These design customizations where direct visual calibrations that the stakeholder preferred.

SmartWatch

D.A. Norman explains the natural mapping as *"Natural mapping, by which I mean taking advantage of physical analogies and culture standards, leads to immediate understanding."* [71, p. 23]. We designed the interaction design for the SmartWatch with natural mapping as far as we could. The simple tap

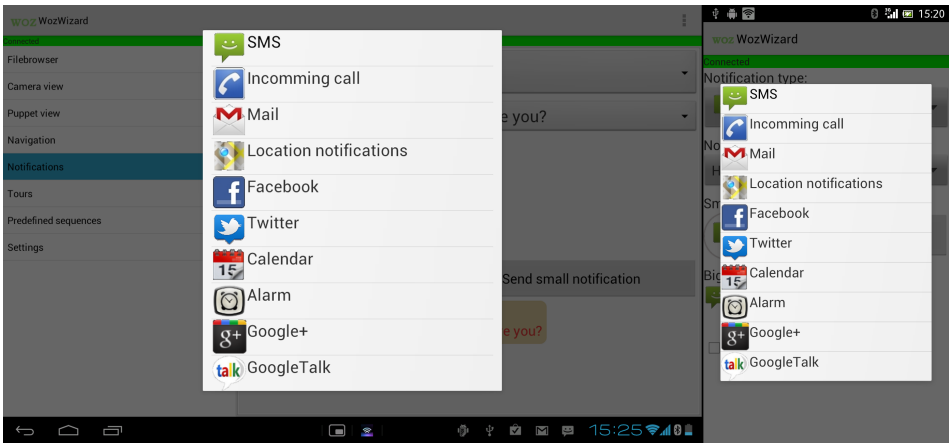

Figure 7.4: Displaying the spinner pop-ups

on the screen symbolizes a button press which is used all over the western world. The up, down, right and left swipes or not as standardized as the tap. Phones and tablets with touchscreen use swipes, but other than that the swipes are not yet widely spread. We wanted an action for the Puppet user to remove any active picture on the screen whenever he wanted to clear his field of view. The right swipe (if SmartWatch on left hand) on the SmartWatch is an action where the hand is moving away from your body. It is mapping the analogy of throwing something away from you, in this case clearing the screen. We also wanted a way for the puppet user to scroll between messages in the inbox. The puppet devices is designed so that the older messages will appear at the top of the screen and newer messages will appear at the bottom of the screen. This design was chosen because it should map the positioning of older and newer messages on the Sony smartphone. On the Sony smartphone older messages are shown if the user is scrolling down, and newer messages shown when scrolling up. So the interaction design of the up and down swipe are natural mapped in a culture standardization for smartphones from Sony Mobile Communications, Samsung, iPhone etc. Another way to design the up and down swipe would be a physical analogy of letters on a stack. The stack analogy would place the newest letters on top and the oldest at the bottom. This analogy is used in some smartphone application, e.g. mail applications. But we wanted the same representation of messages so it should not be the opposite analogy of what it is actually representing. Therefore older messages are above and newer messages below the current showing message. The long press action triggers the puppet to receive a voice command. We chose this interaction because the swipes did not feel nat-

ural to trigger this event and the press event was also occupied. We were also inspired by the Sci-Fi analogy of the Starfleet personnel in Star Trek covering the *communicator pin* on the chest to trigger the voice command receiver. The pinch action can not be overridden and will always close the active extension on the SmartWatch.

The Wizard device receives the broadcast intent from the SmartWatch in a BroadcastReceiver. The BroadcastReceiver is created when a new intent is received and destroyed when it is handled. This generated the problem of not being able to use callbacks to the current Activity. We implemented a solution by using a monitor where the the BroadcastReceiver will put incoming SmartWatch events that will trigger the active Activity to react.

7.3 Usage discussion

The tool is mainly designed to be used with HMD but can to some extent be used with a regular mobile device. Since we used a see-through HMD we can not really use the same techniques on regular mobile devices to augment reality. On the HMD everything that is black will be perceived as transparent while it would cover the camera feed on a mobile device. Since the wish to be able to use the tool on a regular mobile device was presented late in the development the support for this is rather limited. The tool has a command that allows for displaying an image over the camera feed, however it is not fully supported in the user interface. The navigation view has limited support for it and it is possible to use it through the predefined sequence and the tour.

The tool in it self is not that smart other than that it can trigger commands on certain locations and some limited automated voice recognition. The point however is not to be a complete system but rather to give the puppet the impression of a complete system. The applications can be used to simulate many of the different AR techniques it does not support, for example:

- HMD Augmented reality systems
- Gestures
- Facial and object recognition
- Speech recognition and voice commands
- Navigation

- Indoor navigation

In 2011 there were two master thesis projects in the AR area that Sony Mobile Communications hosted.

Augmented Reality Visor Concept Investigated the possibilities of AR systems of the future [2].

Social Interaction in Augmented Reality Gave an overview of how social interaction can be carried out via a pair of AR Visors [1].

The thesis by Svensson and Wozniak provides smart gestures and well thought out storyboards on Visor interaction applications, e.g. grocery shopping. The WOz tool makes it possible to carry on further user test cases on these gestures and storyboards. The resulting design concepts on social interaction via AR Visors in Öfverholm & Petrén's thesis can, with help from the WOz tool, now be tested through HMDs.

7.4 Future work

Multiple wizards It can be challenging for a wizard operator to handle the tool while keeping track of dynamically changing enviroments [14]. The WOz tool could be improved by supporting multiple wizard operators [13]. If the tool allows many wizard operators the responsibilities could be divided among them, removing the multitasking requirement on a single wizard. In this way more sensors and intelligent computer logic can be simulated.

Camera overlay To enable non HMD AR simulation all features should overlay the camera, if active, on the puppet.

Increase SmartWatch support Incorporate the SmartWatch more. E.g. supporting a *menu handling* view on the Wizard. The feedback received when using the SmartWatch has room for improvement as well.

Improved Feedback There is currently much room to improve the feedback provided to the wizard operator, for example, the application does not give any feedback about which view the operator currently resides in when using portrait mode. Further the tool could give a visual confirmation of pictures, commands etc. sent and received to the puppet device. Some commands currently do this through small text messages, *toasts*, but far from all.

Hard coded Removing the hard coded animations, SmartWatch interaction etc. and make it possible to set them during runtime.

Open Source To make the WOz tool available for further development and use outside of Sony there is a plan to release the source code under a MIT licence. This would allow interested parties to use and tailor the tool to fit their needs. It will be released on the Sony Xperia Dev account on github [75].

Recording of sound Charlotte Magnusson noted that it could be benificial to have support for voice transmission from the puppet device to the wizard device. This will allow the wizard operator to better simulate verbal commands.

Tactile Patterns This is also something that Charlotte Magnusson commented on. Currently the tool only supports a simple one second vibration from the vibrator on the puppet device. To test more advanced forms of tactile communication this has to be improved.

Improved camera overlay Currently the overlay used in the camera view is just a dot. It would improve the functionality greatly if there was an easy way to switch the image used so that the wizard operator can fake more advanced tracking.

Chapter 8
Conclusions

This master thesis aimed to provide a WOz tool with the following features:

- Present media such as image, video and sound
- Easy navigation and location based triggering
- Capability to log test results and visual feedback
- User friendly UI and SmartWatch interaction

The resulting application was successful in providing the above functionality. The development of the system progressed very well and we were able to add more functionality requested by our stakeholders. We managed to append the undermentioned features.

- Animated notifications
- Supporting pre-programmed use case scenarios
- Voice interaction testing with limited automation
- Merging images with the camera to enable use on non see-through displays

With the mentioned functionality the WOz tool can be used for design evaluation and usability studies in the following areas:

- HMD Augmented reality systems
- Gestures
- Facial and object recognition

- Speech recognition and voice commands
- Navigation
- Indoor navigation

Chapter 9
Definitions, acronyms and abbreviations

API Application Programming Interface

AR Augmented Reality

GUI Graphical User Interface

HMD Head Mounted Display

MR Mixed Reality

protobuf Protocol buffers

Puppet device The device that receives pictures, sound etc to be shown in Visors

Puppet user The user wearing the puppet device

UI User Interface

Visor A pair of glasses with provided system needed for an AR experience

Wizard device The device sending content to be shown on the puppet

Wizard user The person operating the Wizard device i.e. the one pulling the strings

Wizard operator see Wizard user

WOz Wizard of Oz

XP eXtreme Programming

References

[1] C. Öfverholm and O. Pétren. Social interaction in augmented reality, 2011.

[2] B. Svensson and M. Wozniak. Augmented reality visor concept, 2011.

[3] Sony Mobile Communications. Sony SmartWatch. `http://www.sonymobile.com/gb/products/accessories/smartwatch/`. [Online] [Accessed 12 Nov 2012].

[4] P. Chippendale, B. Prestele, D. Buhrig, P. Eisert, S. BenHimane, V. Tomaselli, H. Jonsson, G. Alce, Y. Lasorsa, M. de Ponti, and O. Pothier. VENTURI - immersiVe ENhancemenT of User-woRld Interactions. White paper, Accessed 16 Oct 2012, 2012.

[5] Russell Freeman. Virtuality continuum 2. `http://en.wikipedia.org/wiki/File:Virtuality_Continuum_2.jpg`. Public Domain. [Accessed 26 Oct 2012].

[6] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12):1321–1329, December 1994.

[7] R.T. Azuma et al. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[8] Ivan E. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress*, pages 506–508, 1965.

[9] O. Bimber, R. Raskar, and M. Inami. *Spatial augmented reality*. AK Peters, Wellesley MA, 2005.

[10] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R.W. Picard, and A. Pentland. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments*, 6(4):386–398, 1997.

[11] J.F. Kelley. An empirical methodology for writing user-friendly natural language computer applications. In *CHI '83 Proceedings of the SIGCHI Conference of Human Factors in Computing Systems*, pages 193–196, 1983.

[12] S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J.D. Bolter, and M. Gandy. Wizard of oz support throughout an iterative design process. *Pervasive Computing, IEEE*, 4(4):18 – 26, oct.-dec. 2005.

[13] Steven Dow, Jaemin Lee, Christopher Oezbek, Blair MacIntyre, Jay David Bolter, and Maribeth Gandy. Wizard of oz interfaces for mixed reality applications. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1339–1342, New York, NY, USA, 2005. ACM.

[14] Y. Li, J.I. Hong, and J.A. Landay. Design challenges and priciples for wizard of oz testing of location-enhanced applications. *IEEE Pervasive Computing*, 6:70–75, April 2007.

[15] Phone Scoop. Smartphone. `http://www.phonescoop.com/glossary/term.php?gid=131`. [Online] [Accessed 25 Oct 2012].

[16] E. Barba, B. MacIntyre, and E.D. Mynatt. Here we are! where are we? locating mixed reality in the age of the smartphone. *Proceedings of the IEEE*, 100(4):929 –936, april 2012.

[17] Android. `http://www.android.com`. [Online] [Accessed 2 Nov 2012].

[18] Alvaro Fuentez Vasquez. Diagram System architecture Android. `http://commons.wikimedia.org/wiki/File:Diagram_android.png`. [Accessed 2 Nov 2012].

[19] Google IO. Anatomy of an Android. `https://sites.google.com/site/io/anatomy--physiology-of-an-android`. [Online] [Accessed 29 Oct 2012].

[20] Open Handset Alliance. Industry leaders announce open platform for mobile devices. `http://www.openhandsetalliance.com/press_110507.html`. [Press Release] [Accessed 29 Oct 2012].

[21] O. Cakmakci and J. Rolland. Head-worn displays: a review. *Display Technology, Journal of*, 2(3):199 –216, sept. 2006.

[22] Google. Project glass. https://plus.google.com/+projectglass/posts. [Online] [Accessed 9 Nov 2012].

[23] Wearcompevolution.jpg. http://commons.wikimedia.org/wiki/File:Wearcompevolution.jpg. [Online] [Accessed 11 Nov 2012].

[24] K. Kiyokawa. Trends and vision of head mounted display in augmented reality. In *Ubiquitous Virtual Reality (ISUVR), 2012 International Symposium on*, pages 14 –17, aug. 2012.

[25] Wei-Qing Gao, Shi-E Zhou, Guo-Qiang Lv, and Hai Ming. Key-problem analysis and experimental research of stereo hmd used in augmented reality. pages 75090H–75090H–10, 2009.

[26] K.H. Park, S.H. Lim, Y. Song, and D. Park. Ufc: A ubiquitous fashionable computer, 2005.

[27] Vuzix. About vuzix. http://www.vuzix.com/corporate/index.html. [Online] [Accessed 29 Oct 2012].

[28] Antonio Zugaldia. Google glass detail. http://en.wikipedia.org/wiki/File:Google_Glass_detail.jpg, 2012. [Online] [Accessed 16 Nov 2012].

[29] Google Project Glass. Google project glass: Official concept walkthrough video, "one day". http://www.youtube.com/watch?v=5R1snVxGNVs. [Online] [Accessed 15 Jun 2012].

[30] Recon Instruments. Mod live heads-up display. http://en.wikipedia.org/wiki/File:MOD_Live_Heads-up_Display.jpg, 2012. [Online] [Accessed 16 Nov 2012].

[31] Recon Instruments. Heads-up display point of view. http://en.wikipedia.org/wiki/File:Heads-up_Display_Point_of_View.jpg, 2012. [Online] [Accessed 16 Nov 2012].

[32] Recon Instruments. http://www.reconinstruments.com. [Accessed 29 Oct 2012].

[33] Jonghoon Seo, Jinwook Shim, JiHye Choi, James Park, and Tack-don Han. Enhancing marker-based ar technology. In Randall Shumaker, editor, *Virtual and Mixed Reality - New Trends*, volume 6773 of *Lecture Notes in Computer Science*, pages 97–104. Springer Berlin Heidelberg, 2011.

[34] Tai-Wei Kan, Chin-Hung Teng, and MikeY. Chen. Qr code based augmented reality applications. In Borko Furht, editor, *Handbook of Augmented Reality*, pages 339–354. Springer New York, 2011.

[35] Jan Herling and Wolfgang Broll. Markerless tracking for augmented reality. In Borko Furht, editor, *Handbook of Augmented Reality*, pages 255–272. Springer New York, 2011.

[36] Donald A. Norman. Natural user interfaces are not natural. *interactions*, 17(3):6–10, May 2010.

[37] Nintendo. Nintendo wii official site. `http://www.nintendo.com/wii`. [Online] [Accessed 11 Nov 2012].

[38] Microsoft. Kinect. `http://www.microsoft.com/en-us/kinectforwindows/`. [Online] [Accessed 11 Nov 2012].

[39] Pranav Mistry and Pattie Maes. Sixthsense: a wearable gestural interface. In *ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation*, SIGGRAPH ASIA '09, pages 85–85, New York, NY, USA, 2009. ACM.

[40] Pranav Mistry, Pattie Maes, and Liyan Chang. Wuw - wear ur world: a wearable gestural interface. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 4111–4116, New York, NY, USA, 2009. ACM.

[41] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 441–450, New York, NY, USA, 2011. ACM.

[42] K. Smith. Face recognition. `http://www.biometrics.gov/Documents/FaceRec.pdf`. [Accessed 29 Oct 2012].

[43] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, January 1991.

[44] Android Developers. Android package index. `http://developer.android.com/reference/packages.html`. [Accessed 2 Nov 2012].

[45] TAT The Astonishing Tribe. Tat augmented id. `http://www.youtube.com/watch?v=tb0pMeg1UN0`. [Online] [Accessed 12 Nov 2012].

[46] Face.com. About. `http://developers.face.com/docs/about`. [Online] [Accessed 6 Nov 2012].

[47] R. Wauters. Face.com's api now lets developers scan 5,000 photos per hour, free of charge, February 2011. [Online] [Accessed 6 Nov 2012].

[48] Y. Taigman and L. Wolf. Leveraging Billions of Faces to Overcome Performance Barriers in Unconstrained Face Recognition. *ArXiv e-prints*, August 2011.

[49] F. Lardinois. Confirmed: Google's motorola mobility acquires image and gesture recognition company viewdle. http://techcrunch.com/2012/10/03/googles-motorola-mobility-acquires-image-and-gesture-recognition-company-viewdle, October 2012. [Online] [Accessed 6 Nov 2012].

[50] Viewdle. Company overview. http://viewdle.com/about.html. [Online] [Accessed 6 Nov 2012].

[51] J. Dalrymple. Apple buys facial-recognition firm polar rose. http://news.cnet.com/8301-13579_3-20017018-37.html, September 2010. [Online] [Accessed 6 Nov 2012].

[52] Apple. Siri. http://www.apple.com/ios/siri/siri-faq/. [Online] [Accessed 15 Jun 2012].

[53] M.G. Siegler. Siri, do you use nuance technology? siri: I'm sorry, i can't answer that. http://techcrunch.com/2011/10/05/apple-siri-nuance/, October 2011. [Online] [Accessed 31 Oct 2012].

[54] The 2012 market leaders. http://www.speechtechmag.com/Articles/Editorial/Feature/The-2012-Market-Leaders-83629.aspx, July 2012. [Online] [Accessed 30 Oct 2012].

[55] Google Inc. Google voice search. http://www.google.com/mobile/voice-search/. [Online] [Accessed 12 Nov 2012].

[56] Microsoft Corporation. Microsoft tellme. http://www.microsoft.com/en-us/tellme/. [Online] [Accessed 12 Nov 2012].

[57] Nuance Communications. Nuance speech engine. http://www.nuance.com/. [Online] [Accessed 12 Nov 2012].

[58] D.A. Kolb. *Experiential learning: experience as the source of learning and development*. Prentice Hall, Englewood Cliffs, NJ, 1984.

[59] D. West, T. Grant, M. Gerush, and D. D'Silva. Agile development: Mainstream adoption has changed agility, January 2010.

[60] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, October 1999.

[61] Soren Lauesen. *Software Requirements: Styles & Techniques*. Addison-Wesley, January 2002.

[62] A. Oulasvirta, E. Kurvinen, and T. Kankainen. Understanding contexts by being there: case studies in bodystorming. *Personal and ubiquitous computing*, 7(2):125–134, 2003.

[63] Eclipse. The eclipse foundation open source community website. `http://www.eclipse.org`. [Online] [Accessed 11 Nov 2012].

[64] Android Developers. Android developer tools. `http://developer.android.com/tools/help/adt.html`. [Online] [Accessed 11 Nov 2012].

[65] Android Developers. Android debug bridge. `http://developer.android.com/tools/help/adb.html`. [Online] [Accessed 11 Nov 2012].

[66] Git. `http://git-scm.com`. [Online] [Accessed 11 Nov 2012].

[67] Sony Mobile Communications. Sony xperia s. `http://www.sonymobile.com/gb/products/phones/xperia-s/`. [Online] [Accessed 12 Nov 2012].

[68] Sony Mobile Communications. Sony tablet s. `http://www.sony.co.uk/product/sony-tablet-s/tab/overview/`. [Online] [Accessed 12 Nov 2012].

[69] Vuzix. Star 1200. `http://www.vuzix.com/augmented-reality/products_star1200.html`. [Online] [Accessed 29 Oct 2012].

[70] Google Developers. Protocol buffers developer guide. Accessed 25 Oct 2012.

[71] D.A. Norman. *The Design of Everyday Things*. Basic Books, New York, 1988.

[72] Android Developers. Fragments. `http://developer.android.com/guide/components/fragments.html`. [Accesed 24 Oct 2012].

[73] S. Litchfield. Screen sunlight tests. `http://www.allaboutsymbian.com/flow/item/14661_Screen_sunlight_tests.php`, April 2012. [Online] [Accessed 24 Oct 2012].

[74] B. Wong. Color blindness. *Nature Methods*, 8:441–441, June 2011.

[75] Sony. Xperia developer world. `https://github.com/sonyxperiadev`. [Online] [Accessed 13 Nov 2012].

Appendix A
Comments

A.1 Klas Hermodsson

It has been my privilege to be an advisor to this Master Thesis work. The tool will be used in upcoming usability studies and prototypes for evaluating interaction design. The tool has evolved a lot from my very rough initial development effort. I am particularly happy about the feature for streaming camera images from the puppet device to the wizard device. The connectivity is also stable which is critical to a tool like this where lost connectivity between puppet and wizard devices may result in the need to redo a complicated test scenario.

The current state of the tool is an excellent starting point for conducting tests and design evaluations while allowing for development of new tool features. The fact that this tool is focused on prototyping on the go situations with mobile devices makes it a great match for our research and will allow us to quickly assess user experience of new ideas.

To enable further use for us and other VENTURI partners we will most likely add more support for video see through scenarios. The logging features are also very useful and the possibility to capture the camera view together with the user interface elements visible to the puppet user makes the tool even more useful in analysing the results of tests.

A testament to the tool's usefulness and foundation for further work is the fact that Sony Mobile Communications has decided to release this tool as open source. We anticipate interest from both industry and academia in using and extending the tool.

A.2 Günter Alce

When conducting user studies of a system which is in its early development it is essential to be able to simulate part of the system to be able to collect feedback from potential users. A wizard of Oz tool is one way of doing it. The main goal of this master thesis was to increase the number of new features and make it as flexible as possible.

The wizard of Oz tool will be used when conducting user studies e.g. for the VENTURI project. During the development of the new features for the tool we worked iteratively and changed the wizard user interface often. The interface was developed both for a tablet and phone form factor. Unfortunate too little ”real” testing was done on how flexible and useful the wizard of Oz tool is in action. However at the end of the development we performed a test with real user and found some interesting outcomes such as moving around in the tool is not so effective during a user study if the phone form factor is used.

Further interesting outcome was that the predefined sequence view will probably be used most. If we had performed real test earlier in the project we would have focused more in the predefined sequence view. That said still the main goal of the project was to add as many features as possible and that was a success. Another outcome was that the tablet is more efficient when performing user studies since it gives a better overview of the features and it is easier to change views.

A.3 Charlotte Magnusson

To be able to quickly lo-fi prototype designs is very valuable, but so far prototyping designs intended to be used on the move (like AR - augmented reality) has been problematic. One can use a human to ”play a device”, but to be able to test more technical details like specific interactions & responses more tools are needed. The current tool has quite a wide potential area of use - one can monitor the user and send both images and sounds to the device the user is holding. The tool is useful as it is, but would be even more useful with a few improvements:

1. Ability to send sounds from the user device (or an external microphone) back to the test leader. This allows better monitoring of what happens, and also allows the ”wizard” to respond to verbal input from the user.

2. Ability to send tactile patterns to the user device (possibly by tapping a pattern on the screen).

3. More ability to fake dynamic gesture responses - possibly by allowing the test leader to drag stuff on screen, adjust sound feedback volume (or modify tactile feedback as in point 2 above).

4. Allow for several test leaders, but also more devices (currently I understand things to be designed for one ”screen device” + one ”interaction device” on the wrist).

Possible future work could also involve a client server module so that one can prototype basic interactions between users (eg games, friendfinders etc where you want to respond to and interact with other persons nearby) - some of this can probably be faked if 4) above is implemented, but it might be helpful to have the possibility to specify what happens when users are close to each other, point to each other etc without having to monitor everything ”by hand”.

Still, also in the current state of development, I find the current tool a good starting point, and would like to use it in future projects myself.

Appendix B
Requirements

B.1 Requirements Matrix

The following pages contain a matrix with the requirements on the tool. They are divided into *types* and are *specified*. Dependencies are stated in the *requires* column and in some cases they have a *priority*. The priority was set by Günter Alce. Finally there is a *status* column to keep track of the progress.

Requirement	Name	Type	Specification	Requires	Prio (1=highest)	Status
D01	Wizard notification saving	Data	The Wizard device shall be able store notifications			Done
D02	Puppet picture save	Data	The Puppet device shall be able to save the automatically taken pictures in a predefined folder			Done
D03	Puppet save new data	Data	The Puppet device shall save new received data			Done
D04	Wizard save new data	Data	The Wizard device shall save new received data			Done
D05	Wizard Coordinate-Commands pairs	Data	The Wizard device shall be able to store commands for different coordinates			Done
D06	Wizard settings file	Data	The Wizard device shall save preferences on a file			
D07	Wizard custom sound directions	Data	The Wizard device shall be able to store customized settings for sound directions			Done
D08	Wizard transparent direction pictures	Data	The Wizard device shall have transparent directions pictures.			Done
D09	Wizard tour marker	Data	The Wizard device shall be able to save a marker with coordinates, command, threshold			Done
De01	Wizard Connection status bar	Design	The Wizard device's UI shall have a connection status bar			Done
De02	Wizard Grid menu	Design	The Wizard device's UI shall be presented in a Grid menu instead of list of items		Replaced with De14 and De15	Deprecated
De03	Wizard Map creation	Design	The Wizard device's UI shall have a section for map creation			Done
De04	Wizard File browsing	Design	The Wizard device's UI shall have a section for file browsing			Done
De05	Wizard User view	Design	The Wizard device's UI shall have a section for showing user view		10	Done
De06	Wizard Camera view	Design	The Wizard device's UI shall have a section showing video from the Puppet device's camera		11	Done
De07	Wizard Tour view	Design	The Wizard device's UI shall have a section for managing tours			Done
De08	Wizard Notification creator	Design	The Wizard device's UI shall have a section for notification creation	D05	8	Done
De09	Wizard Manual navigation	Design	The Wizard device's UI shall have a section for manual navigation		7	Done
De10	Wizard Settings section	Design	The Wizard device's UI shall have a section for settings	D06		Done
De11	Puppet Loading feedback	Design	The Puppet device shall show a loading icon while waiting for current location map		9	Done
De12	Wizard Manual navigation activate sound	Design	The Wizard device's Manual navigation view shall have a way to activate sound with image pairings	De09		Done
De13	Wizard Manual navigation activate transparent directions	Design	The Wizard device's Manual navigation view shall have a way to activate transparent directions images.	D08		Done
De14	Wizard filebrowser list	Design	The Wizard device shall show the filebrowser as a list if it's portrait mode			Done

Requirement	Name	Type	Specification	Requires	Prio (1=highest)	Status
De15	Wizard filebrowser grid	Design	The Wizard device shall show the filebrowser as a grid if it's landscape mode			Done
De16	Wizard alphabetical filebrowser	Design	The Wizard device's filebrowser shall show the files in alphabetical order	De14, De15		Done
De17	Wizard jump to User view	Design	The Wizard device's views shall have a menu choice to jump the User view			Done
De18	Wizard Notification sound messsage	Design	The Wizard device's Notification view shall have a check box for enable sound			Done
De19	Wizard Notification list icons	Design	The Wizard device's Notification view shall have notification type icons in the list of notification types			Done
De20	Wizard Predefined sequence view	Design	The Wizard device's UI shall have a section for predefined sequence		8	Done
De21	Wizard Camera interval	Deisgn	The Wizard device's Camera view shall have a input for picture taking interval value			Done
De22	Wizard Tour view marker input	Design	The Wizard device's Tour view shall have a section for marker creating	De07, D09		Done
De23	Wizard Tour view marker edit	Design	The Wizard device's Tour view shall have a section for marker editing	De07, D09		Done
De24	Wizard Tour view map controls	Design	The Wizard device's Tour view shall have controls for sending an image of the map	De07		Done
De25	Wizard Tour view toogle tour view	Design	The Wizard device's Tour view shall have controls for starting the tour	De07		Done
Fe01	Puppet Reconnect	Feature	The Puppet device shall be able to reconnect if the connection is dropped.			Done
Fe02	Wizard Reconnect	Feature	The Wizard device shall be able to reconnect if the connection is dropped.			Done
Fe03	Receive SmartWatch signals	Feature	The Wizard device shall be able to receive signals from a SmartWatch			Done
Fe04	Interpret SmartWatch signals	Feature	The Wizard device shall be able to interpret signals from a SmartWatch			Done
Fe05	Wizard Audio transmission	Feature	The Wizard device shall be able to send audio clip			Done
Fe06	Puppet Audio transmission	Feature	The Puppet device shall be able to send audio clip			
Fe07	Wizard Video transmission	Feature	The Wizard device shall be able to send video clip			Done
Fe08	Puppet Video transmission	Feature	The Puppet device shall be able to send video clip			Done
Fe09	Puppet Coordinates transmission	Feature	The Puppet device shall be able to send current location coordinates to the Wizard			Done
Fe10	Wizard text message transmission	Feature	The Wizard device shall be able to send text messages			Done
Fe11	Wizard Coordinates request	Feature	The Wizard device shall be able to request current location coordinates from the Puppet device			Done
Fe12	Puppet Coordinates calculation	Feature	The Puppet device shall be able to find coordinates for current location			Done

Requirement	Name	Type	Specification	Requires	Prio (1=highest)	Status
Fe13	Wizard Coordinates receive	Feature	The Wizard device shall be able to receive coordinates			Done
Fe14	Wizard picture command	Feature	The Wizard device shall be able to send take picture command			Done
Fe15	Wizard vibration command	Feature	The Wizard device shall be able to send a vibration command			Done
Fe16	Puppet start vibrate	Feature	The Puppet device shall be able to start vibrate after receiving a vibration command			Done
Fe17	Puppet Voice command Wizard voice command extraction	Feature	The Puppet device shall be able to record voice commands via audio			Done
Fe18	Wizard send notifications randomly	Feature	The Wizard device shall be able to extract commands from voice input			Done
Fe19	Wizard send notifications randomly	Feature	The Wizard device shall be able to send notifications randomly		Removed	
Fe20	Wizard Generate notifications	Feature	The wizard tool shall be able to generate automated notifications		Removed	
Fe21	Puppet Picture mode	Feature	The Puppet device shall be able to automatically take pictures in a given interval			Done
Fe22	Wizard Location Comparing	Feature	The Wizard device shall compare the puppets location to the stored coordinate-command pairs during tour mode			Done
Fe23	Wizard Location Monitor	Feature	The Wizard device shall continuously monitor the puppets location during tour mode			Done
Fe24	Wizard Image transmission	Feature	The Wizard device shall be able to send an image			Done
Fe25	Puppet Image transmission	Feature	The Puppet device shall be able to send an image			Done
Fe26	Puppet sound paired to image	Feature	The puppet shall be able to play a sound that has been paired with an image if a command to show the image is received			Done
Fe27	Puppet handle transparent pictures	Feature	The Puppet device shall be able to place transparent pictures on top of the the camera if it's showing			Done
Fe28	Wizard save check box states	Feature	The Wizard device shall save the states of the check boxes during run time in settings			Done
Fe29	Wizard show indicator	Feature	The Wizard device shall be able to place an indicator on top of the camera feed when the feed is pressed			Done
Fe30	Wizard send indicator coordinates	Feature	The Wizard device shall be able to send the coordinates of the indicator to the puppet device			Done
Fe31	Puppet show indicator	Feature	The Puppet device shall be able to place an indicator on the screen from the coordinates received from the Wizard device	Fe30		Done
Fe32	Wizard clear indicator	Feature	The Wizard device shall be able to clear the indicator showing on top of the camera feed			Done
Fe33	Puppet clear indicator	Feature	The Puppet device shall be able to clear the indicator showing on the screen			Done

Requirement	Name	Type	Specification	Requires	Prio (1=highest)	Status
Fe34	Puppet text to speech	Feature	The Puppet device shall be able to play voice sounds from text			Done
Fe35	Wizard read predefined sequence files	Feature	The Wizard device shall be able to parse predefined sequence files			Done
Fe36	Wizard city tour service	Feature	The Wizard device shall be able to run a tour regardless of which view is active			Done
Fe37	Wizard choose location source	Feature	The Wizard device shall be able to choose the location source provider	Fe38, Fe39		Done
Fe38	Wizard device location	Feature	The Wizard device shall be able to determined its position			Done
Fe39	Puppet device location	Feature	The Puppet device shall be able to determined its position if the hardware supports it			Done
Fu01	Wizard Connection status awareness	Function	The Wizard user shall always be aware of the connection status	De01	2	Done
Fu02	SmartWatch interaction: Next message	Function	The Puppet user shall be able to swipe to the next message by using a down-up swipe on the SmartWatch	Fe03, Fe04	12	Done
Fu03	SmartWatch interaction: Previous message	Function	The Puppet user shall be able to swipe to the previous message by using a up-down swipe on the SmartWatch	Fe03, Fe04	13	Done
Fu04	SmartWatch interaction: Show message	Function	The Puppet user shall be able to tap the SmartWatch to show a message	Fe03, Fe04	14	Done
Fu05	SmartWatch interaction: Voice command	Function	The Puppet user shall be able to long press the SmartWatch to activate the voice command receiver	Fe03, Fe04	15	Done
Fu06	Wizard Map generator	Function	The Wizard device shall be able to generate picture of a map based on received location coordinates	De03, Fe09, Fe11-Fe13, Fe24		Done
Fu07	Wizard Notification input	Function	The Wizard device shall be able to create notification messages from Wizard user input	D01, De08	3	Done
Fu08	Wizard Voice command execution	Function	The Wizard device shall be able to execute commands extracted from Puppet device voice command receiver	Fe06,Fe17,Fe18		Done
Fu09	Puppet take picture	Function	The Puppet device shall be able to take pictures after receiving a picture command from the Wizard device	Fe14, D02, D04	4	Done
Fu10	Wizard Picture mode on	Function	The Wizard device shall be able activate the puppet device to takes pictures automatically e.g. every 30 seconds	D02, Fe21	5	Done
Fu11	Wizard Picture mode off	Function	The Wizard device shall be able deactivate that the puppet device takes pictures	Fu10	6	Done
Fu12	Wizard Video playback	Function	The Wizard device shall be able to playback video from the Puppet device	De06,Fe08		Done
Fu13	Puppet Google search	Function	The Puppet user shall be able to google search to get information about something e.g. "How tall is the Turning Torso"	Fe06,Fe17,Fe18	Removed	

Requirement	Name	Type	Specification	Requires	Prio (1=highest)	Status
Fu14	Puppet Google search	Function	The Puppet user shall be able to google search for opening hours of Biltema	Fu13	Removed	
Fu15	Wizard act on Location	Function	The Wizard device shall be able to set different actions on different GPS coordinates	D05, Fe09, Fe11-Fe13, De07		Done
Fu16	Wizard activate sound	Function	The Wizard device shall be able to activate sound with image on the Puppet device	Fe26		Done
Fu17	Wizard customize sound of directions	Function	The Wizard device shall be able to customize the sound of direction buttons in manual navigation	De09, D07		Done
Fu18	Wizard transparent directions	Function	The Wizard device shall be able to send transparent pictures which will be placed on top of the puppet Camera if it's showing.	Fe27, De13 Fe22, Fe11, Fe12, Fe13, De07, De22,		Done
Fu19	Wizard Create city tour	Function	The wizard operator shall be able to create a city tour	De23		Done
Fu20	Wizard remember check box states	Function	The Wizard device shall be able to remember the state of check boxes	Fe28		Done
Fu21	Wizard camera feed indicator	Function	The Wizard device shall be able to show an indicator when the Wizard is pressing the camera feed	Fe29, Fe30, Fe31		Done
Fu22	Wizard clear all indicators	Function	The Wizard shall be able to clear all the indicators showing in the WOz tool (puppet side and wizard side))	Fe32, Fe33		Done
Fu23	Wizard portrait reach user view	Function	The Wizard shall be able to jump to User view from all the views	De17		Done
Fu24	Wizard sound notification message	Function	The Wizard shall be able to send sound notification messages	Fe34		Done
Fu25	Wizard predefined sequence	Function	The Wizard shall be able to choose commands from predefined sequence files	De20, Fe35		Done
Fu26	Wizard Picture mode interval	Function	The Wizard shall be able to specify the interval at which pictures are taken automatically on the puppet device	Fu10, De21		
Fu27	Wizard enable city tour	Function	The Wizard shall be able to enable the city tour	Fe22, Fu19, De25		Done
Fu28	Wizard edit marker	Function	The Wizard shall be able to edit city tour markers	Fu19, De23, D09		Done
Q01	Stable connection	Quality	Stable connection		1	
Q02	Wizard Proximity limit	Quality	The Wizard device shall be trigger on coordinate-command pairs within 20 meters.			

B.2 Requirements Elicitation Matrix

The following pages contains a matrix mapping our requirement elicitation methods with the elicited requirements. For every requirement, the related elicitation method that found the requirement is stated. In some cases multiple methods are the source of the requirement.

Requirement	Name	Document studies	Pilot experiments	Stakeholder analysis	Similar companies	Brainstorming	Interviewing	Bodystorming
D01	Wizard notification saving		■			■	■	
D02	Puppet picture save					■		
D03	Puppet save new data					■		■
D04	Wizard save new data					■		■
D05	Wizard Coordinate-Commands pairs		■		■			
D06	Wizard settings file					■		
D07	Wizard custom sound directions					■	■	
D08	Wizard transparent direction pictures					■	■	
D09	Wizard tour marker		■			■		
De01	Wizard Connection status bar		■			■		
De02	Wizard Grid menu		■			■		
De03	Wizard Map creation		■					
De04	Wizard File browsing		■					
De05	Wizard User view						■	■
De06	Wizard Camera view		■					
De07	Wizard Tour view		■	■				
De08	Wizard Notification creator					■	■	
De09	Wizard Manual navigation		■		■			
De10	Wizard Settings section					■		
De11	Puppet Loading feedback					■		■
De12	Wizard Manual navigation activate sound							■
De13	Wizard Manual navigation activate transparent directions			■				
De14	Wizard filebrowser list						■	
De15	Wizard filebrowser grid						■	
De16	Wizard alphabetical filebrowser						■	■
De17	Wizard jump to User view						■	■
De18	Wizard Notification sound messsage						■	
De19	Wizard Notification list icons						■	
De20	Wizard Predefined sequence view						■	
De21	Wizard Camera interval						■	
De22	Wizard Tour view marker input		■			■		
De23	Wizard Tour view marker edit		■			■		
De24	Wizard Tour view map controls		■			■		
De25	Wizard Tour view toogle tour view		■			■		
Fe01	Puppet Reconnect		■			■		
Fe02	Wizard Reconnect		■			■		
Fe03	Receive SmartWatch signals					■		
Fe04	Interpret SmartWatch signals					■		■

Requirement	Name	Document studies	Pilot experiments	Stakeholder analysis	Similar companies	Brainstorming	Interviewing	Bodystorming
Fe05	Wizard Audio transmission		■					
Fe06	Puppet Audio transmission		■					
Fe07	Wizard Video transmission		■					
Fe08	Puppet Video transmission		■			■		
Fe09	Puppet Coordinates transmission							
Fe10	Wizard text message transmission		■			■		
Fe11	Wizard Coordinates request							
Fe12	Puppet Coordinates calculation		■					
Fe13	Wizard Coordinates receive		■			■		
Fe14	Wizard picture command		■					
Fe15	Wizard vibration command		■					
Fe16	Puppet start vibrate		■			■		■
Fe17	Puppet Voice command					■		
Fe18	Wizard voice command extraction						■	
Fe19	Wizard send notifications randomly						■	
Fe20	Wizard Generate notifications					■		
Fe21	Puppet Picture mode					■		
Fe22	Wizard Location Comparing					■		
Fe23	Wizard Location Monitor					■		
Fe24	Wizard Image transmission					■		
Fe25	Puppet Image transmission					■		
Fe26	Puppet sound paired to image						■	
Fe27	Puppet handle transparent pictures	■		■				
Fe28	Wizard save check box states						■	
Fe29	Wizard show indicator							■
Fe30	Wizard send indicator coordinates					■		
Fe31	Puppet show indicator							■
Fe32	Wizard clear indicator					■		■
Fe33	Puppet clear indicator					■		■
Fe34	Puppet text to speech	■			■	■		
Fe35	Wizard read predefined sequence files					■		
Fe36	Wizard city tour service					■		
Fe37	Wizard choose location source					■		
Fe38	Wizard device location		■					
Fe39	Puppet device location		■	■			■	
Fu01	Wizard Connection status awareness							
Fu02	SmartWatch interaction: Next message					■		■
Fu03	SmartWatch interaction: Previous message					■		■

Requirement	Name	Document studies	Pilot experiments	Stakeholder analysis	Similar companies	Brainstorming	Interviewing	Bodystorming
Fu04	SmartWatch interaction: Show message							
Fu05	SmartWatch interaction: Voice command							
Fu06	Wizard Map generator							
Fu07	Wizard Notification input							
Fu08	Wizard Voice command execution							
Fu09	Puppet take picture							
Fu10	Wizard Picture mode on							
Fu11	Wizard Picture mode off							
Fu12	Wizard Video playback							
Fu13	Puppet Google search							
Fu14	Puppet Google search							
Fu15	Wizard act on Location							
Fu16	Wizard activate sound							
Fu17	Wizard customize sound of directions							
Fu18	Wizard transparent directions							
Fu19	Wizard Create city tour							
Fu20	Wizard remember check box states							
Fu21	Wizard camera feed indicator							
Fu22	Wizard clear all indicators							
Fu23	Wizard portrait reach user view							
Fu24	Wizard sound notification message							
Fu25	Wizard predefined sequence							
Fu26	Wizard Picture mode interval							
Fu27	Wizard enable city tour							
Fu28	Wizard edit marker							
Q01	Stable connection							
Q02	Wizard Proximity limit							