Pedestrian Navigation and Context Awareness using Tactile Feedback and Sonification of Spatial Data

Master Thesis
Location Aware Systems

Harald K. Jansson

March 6, 2011
Halden, Norway

Østfold University College
Mobile Applications Group

Abstract

This thesis explores the use of sound and vibrotactile computer interfaces in a mobile, navigational context. Current and historical technology is described and examined.

Furthermore, a prototype navigational aid, employing sound and vibrotactile features, is developed. The implementation is based on a consumer-grade off-the-shelf mobile device, to show that such a solution is feasible using current consumer technology.

The application is put through user testing in order to demonstrate that sound and vibrotactile interfaces provide a good alternative to conventional maps, and can work as a navigational aid for both seeing and visually impaired users. The prototype also includes a conventional, interactive map, as a reference for the user testing.

Keywords: Pedestrian Navigation, Sonification, Digital Mapping, HCI, Minimal Attention User Interface, Location Aware Applications, Mobile Devices

Acknowledgements

This thesis would not have come to fruition without the help of several people and institutions.

First and foremost, I would like to thank my supervisor, Gunnar Misund, for many inspirational challenges, and for helping me to stay on track during the writing process.

I would like to thank my dear Heidi, family and friends, for their support, and for putting up with me during my times of absence. I would especially like to extend my thanks to Hans Petter Jansson for proofreading this thesis.

Thanks goes to Jo Herstad at The University of Oslo, Karianne Flaa from The Norwegian Association of the Blind and Partially Sighted, Morten Tollefsen from MediaLT, and Ida Solvang, for providing pointers and for generally being helpful and excellent people.

I would also like to thank my test users for their help during the prototype testing, which was crucial to shed light on the problem statement.

And Majka for her wonderful blueberry smile.

Table of Contents

Chapter 1

Introduction

As the miniturization of consumer electronics progresses, computers have taken up an even greater part of our day-to-day lives. The equivalent to the bulky machines we once had in our offices or homes, can now be fitted in an unit small enough to fit in our pockets. The mobile phone is arguably the most popular of all the mobile devices, and is carried with the owner virtually everywhere he or she travels.

It is likely that we will see a convergence of many different services on the mobile phone, as it is almost ubiquitous in our society and treated as a necessity. Even though the mobile phone is still mostly used for placing calls and sending messages, the platform has potential to be used for much more. One of the recently popularized services is positioning systems. Some phones have embedded GPS receivers, but network based positioning is also used. Even though a healthy concern for the user's privacy must be taken into account, such positioning systems could ease travelling and communication for many people.

Most modern navigation systems provide both visual and aural feedback to the user, and are commonly used while travelling by car. The user is presented with a visual map as well as audio cues for when to turn, road names, time to destination, and so on. These cues are mainly spoken, and are in most cases used as a complement to the map. However, these cues can be essential to make proper use of the system while navigating in a crowded city. One of the main differences between the visual information and the spoken audio cues is the amount of information they represent. While the visual map is saturated with information about the route - often information you do not need at the moment, the spoken cues are mainly to the point, and presented when needed. The visual representation provides the user with the possibility to browse and focus on particular data, while the spoken cues are given one at a time, and at a point chosen by the system. If a system were to

play multiple spoken cues at the same time, the user would in most cases be confused.

1.0.1 Problem Statement

The purpose of this thesis is to explore alternatives to computer interfaces that use visual or spoken communication. I wish to look at what ubiquitous computing means in relation to HCI. I would also like to examine user interfaces that live in the periphery of the user's scope.

In this thesis, I would like to examine how a mobile device could be used to convey a geographical location context without using a spoken or visual interface. Furthermore, with the help of user testing, I will try to answer the following questions:

1. How can a mobile, commercial off-the-shelf device be used to convey location context, without using a visual or spoken interface?
2. Are sound based interfaces suited for mobile pedestrian navigation?
3. Does interfaces based on sound or tactile feedback provide a low level of interference with other tasks in a navigational context?
4. How can a sound based navigational interface help in a day-to-day situation for visually impaired users?

1.0.2 Research Method

The research method used in this thesis facilitates a practical approach to the problem at hand. Therefore, it is natural to devise a case, and to develop a prototype application, for testing and analysis.

The approach can be broken down into the following steps:

1. Pre-analysis. In this part of the thesis I will delve into the background of the problem, find references to similar projects, and acquire the relevant literature. I will in particular need to find studies about the use of non-visual and non-verbal communication in computer interfaces. Also, I will need to find literature about the field of HCI in general, and relevant user tests.
2. Design. In this part of the thesis I will present a technical solution to the problem at hand, and discuss the design of the prototype.
3. Implementation and testing. In this part of the thesis I will present the specifics of the implementation and prototype features. The prototype should be considered as a proof of concept,

rather than a finished application. The effort will be directed at solving the basic requirements of the design, and less time will be spent polishing and making the prototype robust on multiple platforms.

4. User testing. A real-life user test of the prototype will be conducted to gather material used for answering the thesis' central questions.

Literature Study

During the preliminary stages of this thesis, I study existing literature in order to educate myself in the relevant fields of research (i.e. HCI (Human Computer Interaction), Ubiquitous Computing, geographical positioning methods), so I can focus and get ideas for the thesis. Online resources (like Wikipedia) are used when appropriate.

Prototyping

Prototyping is used as a way to experiment with the current capability of modern mobile phones. Before settling on the Android platform, prototypes were built on traditional J2ME and Windows Mobile frameworks. The UbiMap prototype was developed using principles from Agile Development, and has gone through several revisions throughout the thesis work period.

Agile Development

Agile Development is comprised of elements from several development techniques including Extreme Programming, Scrum, Adaptive Software Development, Feature Driven Development and Pragmatic programming. Principles of agile development were used in the prototyping phase of this thesis.

The Agile Manifesto states:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - **Individuals and interactions** over processes and tools
> - **Working software** over comprehensive documentation
> - **Customer collaboration** over contract negotiation
> - **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile Development also follows twelve basic principles[13]:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity–the art of maximizing the amount of work not done–is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Field Testing

Field testing is a common technique to use when dealing with prototype applications. Field testing enables the developer to get feedback directly from the user. The developer can then use ideas and comments from the users to further develop and refine the prototype application. For this thesis, a short field test will be applied at the end of the development process. Questionnaires will be handed out before and after these tests, and the answers provided will help in answering the central questions of this thesis.

1.0.3 Outline

This thesis is made up of seven parts. Part one is the section you are currently reading, providing an introduction to the thesis. Part two is the background chapter, examining related concepts and projects. In part three I will design a prototype system, demonstrating some of the ideas in this thesis. Part four will discuss the implementation in more detail. Part five is dedicated to user tests. Part six holds the findings and discussion, while part seven concludes and gives pointers to future work.

Chapter 2

Background

In this chapter I will give a short introduction to HCI theory. I will also give an overview of some of the positioning methods currently available. Lastly, I am going to present other projects which are similar, or that provide relevant background to this project - especially the ones pertaining to non-visual navigation.

2.1 HCI

2.1.1 Introduction

Our modern society is characterized in many ways by the extensive use of user interfaces. From the basic task of answering your phone (usually this can be done with the press of a button), to the more complex one of designing a building, flying an aircraft or writing a document in a WYSIWYG editor. Our environment is littered with terminals and interfaces; vending machines, ATM terminals, electronic doors of various kinds (some with electronic passcodes), dispensers, navigation systems, and so on. Some of these systems used to require a trained operator (like the first elevators, or early computers which you interfaced with directly through machine code), but in recent years many services are offered directly to the public (at least you do not need someone to operate the lift for you anymore - that is if you are not staying at a hotel which is geared towards unpragmatic luxury). Another obvious example is online banking; this has become popular and is increasingly taken for granted.

Some public systems need to be designed in a way that allow all citizens to operate the system with as much ease as possible. Designing them can be a daunting task, as the design needs to take

into consideration the physical differences of the users, as well as the psychological nuances. This is often referred to as ”universal design”.

But how do we handle the complexity of these systems, and by which rules should we design our user interfaces? The field of HCI is preoccupied with these problems, and researchers from this field have worked on solutions that are ubiquitous in our society today. The nature of the field implies that it is both multi-disciplinary and inter-disciplinary. Some fields typically associated with HCI are: computer science, cognitive science, ergonomics, psychology and aesthetics.

HCI theory also describes the relation between human and computer; but how can we describe the interaction between computer and human in a precise manner? How is the technology physically or mentally related to the user? This is particularly interesting when investigating the use of mobile computers, like cellular phones. The mobile unit is carried with the user at all times, even though it is not needed constantly. The technology enters and leaves the user's focus whenever it is needed or not needed. To further describe this, Mark Weiser uses the term ubiquitous computing[35] (”pervasive computing” and ”calm technology” are also used). We can say that a mobile technology is visible or invisible, is in the center or in the periphery, is in the foreground or in the background, is present or ready, or is explicit or tacit. It is important to note that there are indefinite states between these pairs. As an example, a location-aware application might give the user cues about the environment. The cues may not be interesting, and can be seen as a supplement to the users other senses. While the user may not pay any attention to, or focus on, the cues in general, he or she can slide them in and out of focus.

2.1.2 HCI Mass Consumption Device History

The Memex

The HCI academic history starts with Dr. Vannevar Bush and his Memory Expander[17] (Memex), during the 1930's. Dr. Bush wanted to design a device that aided a person with storing books, records and commmunications. ”It is an enlarged intimate supplement to his memory”[17]. The system was supposed to use microfilms for data storage, and also had the ability to attach small annotations to the records while reading. This bears some semblance to the modern hypertext format.

Sketchpad

Ivan Sutherland designed the first device, called the Sketchpad[33], with which the user could interact directly with the computer screen. During his Ph.D, which was finished in 1963, he designed a system that allowed the user to draw geometrical shapes directly on the screen, using a light pen.

The mouse (1963)

Before designing the mouse[3], Douglas Engelbart also experimented with using a light pen as direct input on a CRT screen. However, the mouse, which provided the user with an input option detatched from the display, proved to be a better solution. Engelbart's mouse provided input on either x-axis or y-axis, and the concept was further enhanced by Bill English to allow for input in both directions simultanously.

Dynabook (late 1960, early 1970)

Dr. Alan Kay worked on the Dynabook[26]. It was a concept which could best be described as similar to a notebook or a tablet computer. Kay also worked on the Smalltalk computer language, which was designed to work analogously to a biological process; using small modules (cells) communicationg with each other via messages. The computers were aimed at children, and the software would evolve along with them. The Dynabook was also to rely on computer networks to do external processing and to tie units together (much like modern cellular or WiFi networks).

Xerox

The Xerox Alto (1973) is often referred to as the first personal computer. It had a graphical user interface, which the user could interact with.

Apple Lisa

In 1983 Apple released the Lisa. The Lisa was a personal computer which used a graphical user interface with icons, and a mouse for navigating the GUI.

Apple Macintosh

Released in 1984, the Apple Macintosh also used a combination of GUI and mouse to interact with the user. The system was geared towards simple users that did not have much interest in the

Figure 2.1: The Sketchpad

computer itself. It was supposed to be easy and intuitive to operate. It is the predecessor to the modern Macintosh computer.

Microsoft Windows

Like the Apple Macintosh, the Windows operating system uses a desktop metaphor, with a GUI and pointer interaction. Apple and Microsoft differ with respect to platform philosophy. Apple considers the computer to be a whole, and deliver hardware and software as an integrated packgage, while Microsoft only delivers the software, which runs on generic components.

Apple Newton

The Newton, produced from 1993-1998, was one of the early PDAs (Personal Digital Assistant). It featured a touch-sensitive screen which was operated with a stylus. The software could recognize hand-written text as an input-mode. It could also recognize shapes which could be manipulated as vector graphics (scaled, rotated etc.).

Palm Pilot

The Palm Pilot, like the Newton, was produced during the nineties. It was designed be small enough to carry in your pocket, and featured gesture-recognition called ”Graphitti”.

Apple Iphone

The Apple Iphone, announced on January 9, 2007, is considered by many as a milestone in PDA/-phone integration. It features a touch-sensitive screen with a graphical interface especially tailored for fingertip interaction. The interface makes use of animations and physics (resistance, flow etc.) to give the user a feeling of interacting with physical objects.

2.1.3 Universal Design

Universal Design is about designing interfaces that are accessible to a wide array of users. Ronald Mace is regarded as one of the founders of the field, and he describes it as follows:

> ”Universal usability is the design of products and environments to be usable by all people, to the greatest extent possible, without the need for adaptation or specialized design.”

Figure 2.2: Apple Iphone

The Principles of Universal Design

The Center for Universal Design[20] states seven principles of Universal Design [1]:

1. Equitable Use
2. Flexibility in Use
3. Simple and Intuitive Use
4. Perceptible Information
5. Tolerance for Error
6. Low Physical Effort
7. Size and Space for Approach and Use

Principle One: Equitable Use

The design is useful and marketable to people with diverse abilities.

Guidelines:

- Provide the same means of use for all users: identical whenever possible; equivalent when not.
- Avoid segregating or stigmatizing any users.
- Provisions for privacy,security, and safety should be equally to all users.
- Make the design appealing to all users.

Principle Two: Flexibility in Use

The design accommodates a wide range of individual preferences and abilities.

Guidelines:

- Provide choice in methods of use.
- Accommodate right- or left-handed access and use.
- Facilitate the user‘s accuracy and precision
- Provide adaptability to the user‘s pace.

[1] Copyright ©1997 NC State University, The Center for Universal Design. Please note that the Principles of Universal Design address only universally usable design, while the practice of design involves more than consideration for usability. Designers must also incorporate other considerations such as economic, engineering, cultural, gender, and environmental concerns in their design processes. These Principles offer designers guidance to better integrate features that meet the needs of as many users as possible.

Principle Three: Simple and Intuitive Use

Use of the design is easy to understand, regardless of the user‘s experience, knowledge, language skills, or current concentration level.

Guidelines:

- Eliminate unnecessary complexity.
- Be consistent with user expectations and intuition.
- Accommodate a wide range of literacy and language skills.
- Arrange information consistent with its importance.
- Provide affective prompting and feedback during and after task completion.

Principle Four: Perceptible Information

The design communicates necessary information effectively to the user, regardless of ambient conditions or the user‘s sensory abilities.

Guidelines:

- Use different modes (pictorial, verbal, tactile) for redundant presentation of essential information.
- Provide adequate contrast between essential information and its surroundings.
- Maximize ”legibility” of essential information.
- Provide compatibility with a variety of techniques or devices used by people with sensory limitations.

Principle Five: Tolerance for Error

The design minimizes hazards and the adverse consequences of accidental or unintended actions.

Guidelines:

- Arrange elements to minimize hazards and errors: most used elements, most accessible; hazardous elements eliminated, isolated, or shielded.
- Provide warnings of hazards and errors.
- Provide fail safe features.
- Discourage unconscious action in tasks that require vigilance.

Principle Six: Low Physical Effort

The design can be used efficiently and comfortably and with a minimum of fatigue.

Guidelines:

- Allow user to maintain a neutral body position.
- Use reasonable operating forces.
- Minimize repetitive actions.
- Minimize sustained physical effort.

Principle Seven: Size and Space for Approach and Use

Appropriate size and space is provided for approach, reach, manipulation, and use regardless of user‘s body size, posture, or mobility.

Guideline:

- Provide a clear line of sight to important elements for any seated or standing user.
- Make reach to all components comfortable for any seated or standing user.
- Accommodate variations in hand and grip size.
- Provide adequate space for the use of assistive devices or personal assistance.

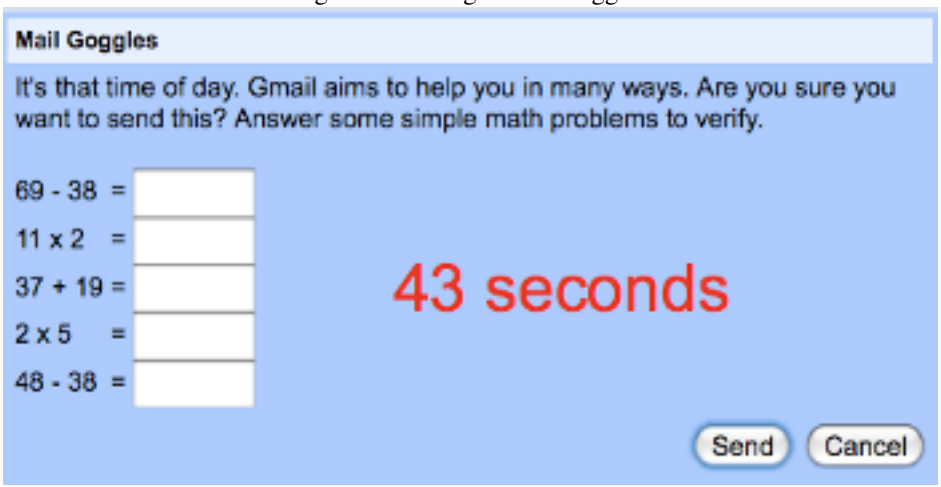
Although the field of Universal Design is geared towards people with disabilities of some kind (i.e. loss of sight/hearing, impaired movement etc.), the principles also apply to people with normal function. In computer interfaces, multiple modes of interaction are often applied to allow a user to operate a program given restrictions in the user's movement, sight or hearing. An example is a voice-activated GPS used in a car, or the tactile response of a mobile phone when receiving a message.

A recent example is Google‘s ”Mail Goggles”[2] feature. This feature requires the user to solve simple problems in order to send a mail. It can be activated to run only on certain times of the day on certain weekdays. The name implies that this feature should be used to avoid sending embarrassing e-mails while being intoxicated. This feature uses principle five - ”Discourage unconscious action in tasks that require vigilance”.

2.1.4 Ubiquitous Computing

If we take a look at computing history, we can define three distinct eras in how we use computers. First is the mainframe era; where all computing power is centralized in one location, using dumb

Figure 2.3: Google Mail Goggles



terminals to access it. The second era is the desktop era; were one user has one machine at his disposal. In contrast to the mainframes, computing power is decentralized in many, but relatively less powerful machines. In the third era - which is what the field of Ubiquitous Computing is trying to define, the notion of one personal computer becomes obsolete. The goal is to provide each user with a multitude of computers, in effect making ”the” computer invisible to the user. Mark Weiser is regarded as the father of Ubiquitous Computing. In 1988 he described it as follows:

> ”Inspired by the social scientists, philosophers, and anthropologists at PARC, we have been trying to take a radical look at what computing and networking ought to be like. We believe that people live through their practices and tacit knowledge so that the most powerful things are those that are effectively invisible in use. This is a challenge that affects all of computer science. Our preliminary approach: Activate the world. Provide hundreds of wireless computing devices per person per office, of all scales (from 1” displays to wall sized). This has required new work in operating systems, user interfaces, networks, wireless, displays, and many other areas. We call our work ”ubiquitous computing”. This is different from PDA’s, dynabooks, or information at your fingertips. It is invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere.”

And:

> ”For thirty years most interface design, and most computer design, has been headed down the path of the ”dramatic” machine. Its highest ideal is to make a computer so

> exciting, so wonderful, so interesting, that we never want to be without it. A less-traveled path I call the ”invisible”; its highest ideal is to make a computer so imbedded, so fitting, so natural, that we use it without even thinking about it. (I have also called this notion ”Ubiquitous Computing”, and have placed its origins in post-modernism.) I believe that in the next twenty years the second path will come to dominate. But this will not be easy; very little of our current systems infrastructure will survive. We have been building versions of the infrastructure-to-come at PARC for the past four years, in the form of inch-, foot-, and yard-sized computers we call Tabs, Pads, and Boards. Our prototypes have sometimes succeeded, but more often failed to be invisible. From what we have learned, we are now exploring some new directions for ubicomp, including the famous ”dangling string” display.”

In the early nineties, Ubiquitous Computing was thought of as something that belonged to the future. Although Weiser‘s vision of Ubiquitous Computing is not a reality, we can argue that in many ways that we live in a society where information flows almost seamlessly between individuals and computers. In ”Yesterday‘s tomorrows: notes on ubiquitous computing‘s dominant vision”[14] Genevieve Bell and Paul Dourish argues that the future described by Weiser and Brown is indeed here - albeit in a different form. The authors present Singapore and Korea as examples of two societies that to a large extent have implemented an Ubiquitous Computing infrastructure. In addition to multiple consumer-friendly technologies (like ordering a cab using SMS), an electronic road pricing system with variable fees was implemented in 1998 by Singapore‘s land Transport Authority[11]. The system makes use of short-band radio, vehicle identification units, smart cards, distributed data collection points and a centralized data centre.

The system is to be further enhanced with GPS-support and a traffic estimation and prediction tool (TrEPS)[6], which gives the possibility to predict congestions based on statistical analysis and realtime information.

Calm Computing

> Basically, the only ”intuitive” interface is the nipple. After that, it’s all learned. (Bruce Ediger, 1995)

The term ”Calm Computing” was coined by Mark Weiser and John Seeley Brown, and deals with computer interfaces designed to not overload the user with information. When people are surrounded by computers in their daily lives, it becomes important to understand how to make

Figure 2.4: Dangling String

interfaces that do not interfere when not needed. In ”Designing Calm Technology” [36] Weiser and Brown describes it as ”that which informs but doesn’t demand our focus or attention”. A calm interface floats between two states, which Weiser describes as ”center” and ”periphery”. Other words are frequently used to describe these states for further precision:

- foreground / background
- present / ready
- explicit / tacit
- visible / invisible

Dangling String Display

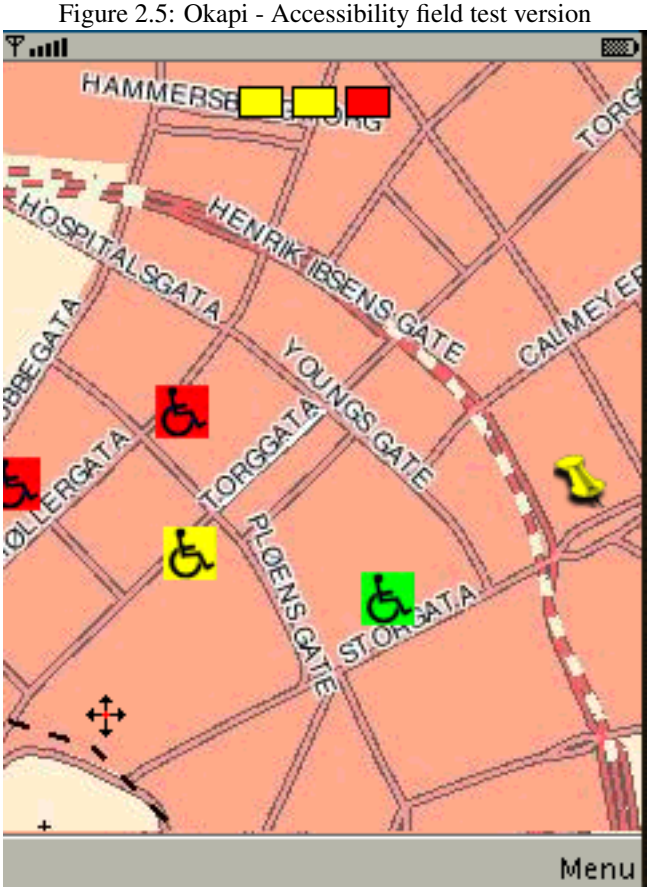
The Dangling String display[36] by artist Natalie Jeremijenko is an early example of calm technology. It uses an 8 foot string attached to an electrical motor to monitor computer network load. When there is heavy network activity the string will flutter about, and when the load is low it will be calm. The network load is thus represented by a physical object, behaving much like wind on a tree.

2.2 The Accessibility Project

The Accessibility Project was started by the Norwegian Mapping Authority[5] along with the Agency for Planning and Building Services. The project aims to provide an updated map of accessibility in Oslo city, to facilitate easier navigation for disabled people.

In 2004, the first analogue map with accessibility data was introduced. This initial version

Figure 2.5: Okapi - Accessibility field test version



got positive feedback from the users, and it was decided to make a digital version of the map. This version was implemented by Norkart AS[4], an independent GIS software house. The digital version offered extra functionality and interaction compared to the paper variant.

A mobile variant of the map is also planned. This is made in cooperation with Østfold University College, which contributes ideas and prototype solutions.

2.3 Okapi

Okapi[23] is an ongoing project at Østfold University College. It is a framework for displaying and working with geographical data on PDAs and mobile phones. The original project was started by Torbjørn Halvorsen and Harald K. Jansson in 2005, but was later worked on by Håkon A. Holmstedt and Tom Heine Nätt.

2.4 Positioning Methods

In this section I will briefly present some of the positioning methods available and look at their strengths and weaknesses. Some methods are suitable for outdoor positioning, some for indoor positioning, and some can be used in both scenarios.

GPS

During the seventies, the American Department of Defense decided to construct a satellite based global positioning system, officially named NAVSTAR GPS (Navigation Signal Timing and Ranging GPS). The goal was to acheive accuracies better than ten metres anywhere on the earth surface. The first experimental satellite was launched in 1978, and the last in the 24 satellite array was launched in 1994.

Every satellite in the array has an unique code that is constantly transmitted on two frequencies (called L1 and L2) along with current date, time and location of the satellite. The L1 and L2 frequencies cater to commercial and military applications respectively, and the commercial signal has an added noise factor that limits the accuracy.

Using four satellites (a military grade receiver only needs three), a GPS receiver can obtain its xyz position relative to the Earth's surface.

The GPS system has also spawned a couple of variants, namely DGPS (Differential GPS), and AGPS (Assisted GPS), which both provide extra accuracy and better TTF (Time To Fix). DGPS uses fixed ground-based radio emitters that broadcast the difference between the current satellite positions and itself. The GPS receiver can then use this information to acquire a more accurate position. AGPS, which has already been implemented in commercial products, uses an assistance server which the GPS receiver downloads satellite information from. This information, often in the form of an Ephemeris-file, is then used to assist the receiver in calculating the current position.

Sine I will be using GPS positioning in the implementation of the prototype, I will take a more detailed look at the GPS output format.

A GPS receiver produces multiple NMEA sentences. I have listed some of them in table 2.1 , but the most common sentence to implement is the GPGGA.

The GPGGA sentence has sixteen comma separated fields with information pertaining to the GPS fix data, as shown in table 2.2. The first field identifies the type of NMEA sentence. The second field is a time stamp, given on the form hhmmssZ in Zulu (UTC) time. The fields for latitude and longitude are compounded values, were the first two digits represent degrees, and the

Table 2.1: GPS NMEA Strings

NMEA	Description	Example
GPGGA	GPS fixed data	$GPGGA ,161229.487,3723.2475,N,12158.3416,W, 1,07,1.0,9.0,M, , , ,*18
GPGGL	Geographic position	$GPGLL,4916.45,N,12311.12,W,225444,A
GPGSA	DOP & active satellites	$GPGSA,A,3,,,,,,16,18,,22,24,,,3.6,2.1,2.2*3C
GPGSV	Satellites in view	$GPGSV,1,1,13,02,02,213,,03,-3,000,,11,00,121,,14, 13,172,05*67
GPVTG	Course and speed	$GPVTG,054.7,T,034.4,M,005.5,N,010.2,K

subsequent digits represent minutes. Thus the latitude value 4124.8963 can be read as 41 degrees and 24.8963 minutes. In a more readable form, seconds would be used instead of decimal minutes and the original value of 4124.8963 would be displayed as 41°24”58”’. To indicate the hemisphere, the latitude and longitude fields are accompanied by fields four and six.

Fix quality can be one of three values: 0, 1, or 2. 0 indicating no fix, 1 indicating GPS fix and 2 indicating DGPS fix. The HDOP (Horizontal Dilution of Precision) field gives a measure of the fix quality. This is a number that is calculated by the GPS receiver based on signal strength, number of satellites seen and signal quality. A number below four is considered to indicate a reliable fix, while a number above six is considered to indicate an unreliable fix.

Fields fourteen and fifteen are mostly used in military grade receivers. They refer to the DGPS system of using ground stations to further enhance the GPS accuracy. Using a DGPS ground station to submit extra positional data, a GPS receiver can achieve accuracies of less tan a centimetre. Field fourteen indicates the age of the DGPS data, and field fifteen is the identification of the ground station in use.

Field sixteen is a checksum which is used for error detection and correction.

While the GPS system in general is an excellent navigation aid for vehicles in open-air conditions, the system has some serious drawbacks for exact pedestrian navigation. Firstly, due to the weak GPS signal, it is only effective while travelling outdoors. For the same reason, accuracy is also affected by so-called ”urban canyons”. This refers to the canyons created by high-rise buildings in a big-city landscape. For pedestrian navigation to be effective, an accuracy of at least five metres is needed, and the current system may have large errors in accuracy, depending on the conditions[32].

The Galileo system[8] is the European counterpart to NAVSTAR GPS. Officially agreed upon in 2003, the program is intended primarily for civilian use, and will give free positioning at accuracies better than four metres. An encrypted commercial service will give accuracies of better than 1 meter.

Table 2.2: GPGGA sentence

Field	Description	Example
1	Sentence identifier	$GPGGA
2	UTC time	170834
3	Latitude	4124.8963
4	Latitude hemisphere	N
5	Longitude	08151.6838
6	Longitude hemisphere	W
7	Position fix indicator	0
8	Number of satellites in use	05
9	Horizontal dilution of precision	1.5
10	Altitude	208.2
11	Altitude unit measure	M
12	Geoidal height	-34
13	Geoidal height measure	M
14	DGPS data age	blank
15	Differential reference station id	blank
16	Checksum	*75

When operational, the Galileo system will hopefully provide for easier pedestrian navigation and greater accuracy.

RFID

RFID (Radio Frequency Identification) comes in three flavours; passive, semi-active and active. These are small radio chips which either reflect or project radio signals (or both). RFID technology is often used in indoor positioning systems, or to locate or identify goods. While passive RFID chips have a range of about 10cm to 1-2 meters, active chips can have a range of up to a hundred meters and a battery lifetime of up to ten years.

Some research has been done on using RFID tags for positioning. One advantage is that the tags can be read at great speeds (up to 150km/h)[19]. An example use would be to place RFID chips in traffic tunnels or other locations, where GPS cannot be used for positioning.

More interesting, however, is to use RFID for pedestrian navigation. RFID technology is cheap enough (5 cents per unit) to deploy on a large scale (e.g. a city). For example, chips could be implemented in traffic signs, sidewalks, stores etc., and work as an addition to other navigation methods.

The most obvious disadvantage to RFID tags is that they need to be placed on all items that

require positioning.

Infra-red

Infrared light emitters and sensors can be used in an indoor positioning system. Positioning of this kind can provide good accuracy (10-20cm)[29]. When using infra-red for positioning, a line of sight is always needed to the emitter. This makes it hard to deploy in complex environments.

Acoustic Location

All use of sound to navigate by is grouped as acoustic location. Here we find a multitude of equipment, including sonar, ultrasound and the human ear. Acoustic location is further divided into two categories; passive and active.

Passive acoustic location is well known (the human ear works this way), and widely adopted. Incoming sound waves are used to position the emitting source. Another example in this category is the passive sonar deployed in sea vessels.

Active acoustic location is based on the basic principle that sound is generated and the echo of that sound is then analysed and used for positioning. This method is used in ultrasound navigation systems like the Sonic Torch[10], or the Binaural Sonic Glasses[28]. Whales and dolphins use echolocation methods of this kind for navigational purposes.

WLAN Positioning

As of this writing, wireless LAN networks have grown in popularity and are widely deployed. These networks can be used for indoor positioning and yield a high degree of accuracy[12]. In the RADAR project[12], the position is triangulated using signal strength from multiple access points.

Cell-ID

Mobile phones are one of the most widely used technologies as of today. Most users have their own phone, which they carry with them during the day. The phone communicates with a stationary radio-tower, and one can therefore get a coarse-grained position from only that. In addition, signal strength to multiple radio-towers can be measured, resulting in a more fine-grained position.

This positioning technique is categorized as a remote positioning type, as it is normally not handled on the mobile phone. Some handsets, though, provide programmatical visibility of the current cell-id, thus a position could be retrieved without the use of external sources. The biggest

problem, however, is the coarse-grained accuracy provided by this method. As this varies with radio tower density, the accuracy provided ranges between 0.2 and 5.0 kilometres.

Lighthouse navigation

Lighthouse navigation is a well-known technique that employs light beacons to navigate by. It is mentioned here, as it provides an interesting background to sound-based navigation.

Electro Magnetism

In a city environment, a multitude of electro-magnetic emitters exist. These could range from electrical infrastructure to cell phones, alarm systems, elevators, and so forth. While these magnetic fields provide no accurate position, they could (with training) provide for greater non-visual context-awareness.

Dead Reckoning

Dead reckoning is often used when other means of positioning are unavailable. By using the last known position, this is extrapolated according to speed and bearing. Some GPS systems use this when the satellite connection is lost (e.g. in tunnels).

2.5 Sonification

Sonification is defined as the use of non-speech audio to convey information. More specifically, sonification is the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation[18].

The human being is an animal that predominantly uses visual input to navigate through its environment. We use a multitude of visual aids when we want to go from A to B, or from A via B and C to D. Humans also use aural and tactile input while navigating, although these are not considered as important as sight. However, blind and visually impaired persons must rely on sound and tactile feedback for their everyday travelling. Usually these senses are trained to be more acute in a person that lacks the ability to see. Blind use acoustic response and sound shadows to visualize their environment; however, they miss out on a lot of visual information. How can this information be presented in another way? And how can we avoid information overload to the the other functional senses? The basic idea of sonification of visual environment is not new. Dr. Leslie Kay was one

of the first to do research on the use of sonification for blind people, end developed various devices during the sixties[27]. All of these devices are based on ultrasound echolocation, and made for close environment navigation. Users of the systems could also recollect the audio environments generated by the devices, even twenty years after first hearing it.

Devices that make use of sonification are often known for using simple pitch-based feedback. However, a multitude of parameters can be used to differentiate between sounds, including, formant shift, origin of sound, rhythm and sound timbre[16]. Sonification of data is especially useful when dealing with data that has a dominant time-factor. While sonification techniques can not be used effectively to take a glance at historical data, like graphs, they have great potential for real-time feedback.

Two approaches can be chosen for sonification of spatial data. The first would be to use purely abstract sounds, like tones, rhythmical patterns, timbres etc., while the other approach would be to use real-life sounds. The latter provides for immediate recollection and less need for training, but is also bound by cultural and social factors. Using purely abstract sounds, users would need more training to grasp the system, but the sounds can be controlled more accurately and contain less abundant sound material.

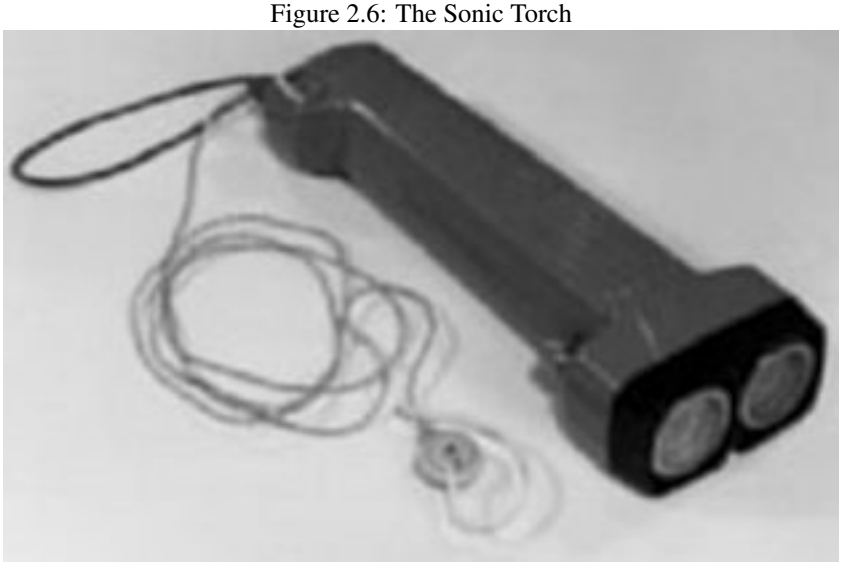
Both approaches can be combined to convey different information. For example, in a navigation application, abstract sounds could be used for directional cues, while ”realistic” sounds could be used to perceive your current context. Some examples of realistic sounds are traffic noise, church bells, drizzling water, clattering dinner plates, etc.. Additionally, these sounds could be placed in 3d space, so that they could be used to navigate by.

2.6 Tactile Feedback

Tactile (or haptic) feedback is used in interfaces were communication with the user is facilitated by using touch or other feedback which can be felt. A simple example of this is the use of small motors in mobile phones which makes the unit vibrate on incoming calls or messages. Haptic interfaces are also used in virtual reality systems and there are APIs to ease the development of such interfaces. One of them is chai3d[1], which is also open source, and thus free to use and experiment with.

Even though most mobile phones include a vibrating function, the use of haptics is still in the early phases for mass consumption technology. Some manufacturers are currently experimenting with visual interfaces that include dynamic surfaces[30]. This is used in conjunction with touch-screen interfaces, were parts of the screen can be raised or lowered to emulate physical buttons.

Figure 2.6: The Sonic Torch



2.7 Previous Work

2.7.1 The Sonic Torch

The Sonic Torch[10] is a hand held device engineered by Dr. Leslie Kay during the year 1965. The unit uses ultrasonic echolocation to generate sounds, and measures distances by sliding between high (far) and low (near) pitches. The Sonar Torch can be held and aimed with one hand, while also using a long cane. Although is was not widely adopted, The Sonic Torch was the first commercial electronic device available for blind people.

> The experimental Sonic Torch was given to 10 boys at the Worcester Grammar School for Blind Boys to find out how they would use it without any formal training. The torch enabled the boys to move along the school corridors as if they were sighted and up and down the stairs between floors, without hesitation. Outside, in the school grounds, they used the sensor to avoid objects and navigate their way around more complex areas. Some boys ventured into the local city centre without any other aid - apart from their ability to echolocate naturally. Ultra Electronics Ltd. were contracted to develop a prototype for more rigorous evaluation. Units were sent to a number of agencies worldwide who showed an interest in the project.

Figure 2.7: The Binaural Sonic Glasses

2.7.2 The Binaural Sonic Glasses

Also engineered by Dr. Kay, the Binaural Sonic Glasses[28] (later commercially dubbed the Sonicguide) became a succesful aid for blind people. Here is a testimony from a long time user[22]:

> Some O& M instructors may regard the following as blasphemy. However, it is crucial to convey the importance of the SonicGuide to my mobility. The SonicGuide is my primary aid and the cane is the secondary aid. Although I carry the cane almost everywhere, the only time I regard it as crucial is where there are dangers such as on railway stations. I usually hold it with the tip just ahead of my feet and often with the tip not contacting the ground. I stress that this is not due to foolhardiness but because I know from the SonicGuide what is ahead. On railway stations, I usually use the cane to scan the edge of the platform and the SonicGuide to watch for obstacles.

The Binaural Sonic Glasses uses the same techniques as the Sonic Torch - echolocation using ultrasound. As they are fitted in the same way as normal glasses they free up the hands for other uses like a long cane or a guide dog.

2.7.3 The 'K' Sonar-Cane

The Sonar Cane is the last in the line of mobility aids developed by Dr. Kay. It is much similar to the Sonic Torch, although much smaller in size. It is mounted on and used in conjunction with a traditional white cane. As with the Sonic Torch it uses echolocation and ultrasound to measure distance to objects.

Figure 2.8: The Sonar Cane

2.7.4 GuideCane

The GuideCane[15] is a device that physically guides a blind person along a track, or around obstacles. It is partly derived from the white cane, but uses a small robot at the far end of the cane to guide the user about.

2.7.5 BATS

Auralization of spatial data is done to some extent in traditional accessability systems. For example the sound a truck makes when backing up, sounds at zebra crossings etc.. These are simple implementations but can provide an entry point to discussion.

There is also some experimental work done in this field. One of the most interesting is the B.A.T.S. (Blind Audio Tactile Mapping System) project[31]. The aim of this project is to provide free software that makes maps and geographical data more accessible to the blind or visually impaired. There is also a prototype application associated with this project. It uses ambient sounds (traffic and crowds for urban environments, birdsong for parks, train sounds for railways etc.) to position the user on a map. It also mixes the sounds (volume, not stereo image) in realtime.

2.7.6 SIKTE navigation system for deafblind users

This project was funded by the National Insurance Administration in Norway. The goal was to find out whether GPS equipment paired with vibrotactile feedback could work as a navigational aid for deafblind users.

Cognita, an IT company specializing in assistive technology, helped in developing a working prototype for the project, and user tests were undertaken.

The use of vibrotactile feedback makes this project especially interesting. The arguments for using vibrotactile feedback are as follows:

- It is a parallell information channel.
- Users can still utilize residual hearing and vision.
- Users hand are free to operate mobility stick, or guide dog.
- Possibility to add further vibrating units, thus increasing the ”bandwidth” of information.

It is especially important to note that tactile feedback could work in conjunction with sight and hearing, without disturbing these. I will use vibrotactile feedback as an alternative to navigational sounds in my project, although using a cellular phone platform will reduce the possibilities for

precise feedback. To convey signals, a morse-like system was employed. The following table shows how the various signal were coded:

Table 2.3: SIKTE vibration codes

GPS information:	Vibration Code:
20 metres before trackpoint:	
Right	(Right)—
Left	(Left)—
Straight ahead	(Both)-
Arrival at trackpoint:(Both)—	
After trackpoint - correction of course:	
Right	(Right)—
Left	(Left)—
Slightly right	(Right)-
Slightly left	(Left)-
Straight ahead	(Both)-
U-turn	(Alternate left end right)—

2.7.7 SWAN (System for Wearable Audio Navigation)

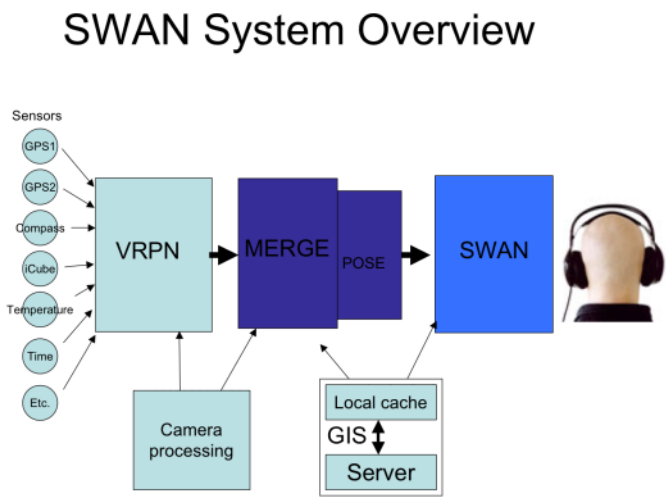
The SWAN[34] project has developed a mobile prototype that uses the following components to aid the wearer:

- Beacon sounds - guiding the user along a path.
- Object sounds - an ambient sound declaring an obstacle, or object.
- Surface transition - indicates a change in surface (e.g. grass to concrete).
- Location sounds - indicates the location of the user (e.g. building, shop, office)
- Annotations - brief messages or sounds recorded by the users themselves.

The SWAN project also utilizes so-called bonephones for delivering their audio signal. These earphones works by conducting vibration directly to the cochlea through the skull of the user, instead of through the ear. Using bone-conducting earphones for this kind of project has some advantages over normal earphones. Firstly, they do not block the outer ear so that other auditory information can pass unobtrused. Secondly, they also work for users that have a outer-ear disorder.

This project bears a strong resemblance to my own project, with the exception that my approach is to use off-the-shelf consumer components (like mobile phones) to acheive the same goal.

Figure 2.9: The SWAN Architecture



2.7.8 AudioGPS

AudioGPS[25] is a project that aims to develop a minimal attention user interface for the sighted person who is simultaneously involved in other demanding tasks. The project prototype is based on traditional computer hardware, and the system is carried in a rucksack. The use of different sound timbres, controlled via a MIDI interface, is employed to feed the user with spatial information. The project also makes use of 'open' stereo headphones in order to keep important environmental background noises.

Chapter 3

UbiMap Design

This chapter describes the basic design of the UbiMap client and server.

3.1 The basics

The UbiMap application is basically designed as a mobile map client with added functionality for proximity detection and proximity alerts. The client communicates with the user, and the server holds extra information about hotspots. The client presents the user with a pannable and zoomable map, which also displays the hotspots as circles with specified radii. Alerts are triggered when the geographical position of the user overlaps with a predefined hotspot. Alerts can be tactile, audible or both.

As for the hardware device, an off-the-shelf publicly available device will be chosen. This is discussed further in the next chapter.

3.2 Use Cases

Viewing a visual map

The user should be able to view and navigate a visual map. This is similar to most mobile mapping solutions. In this mode, the user will be able to pan the map in all directions, zoom in and out, and change the map mode between vector graphics and aerial photography. The basic map client should also include functionality for centering the map view on the current position. The reason for including a fully functional map client is to assess differences between visual map usage and sound,

and vibrotactile representations. The plain visual map is also used to assess whether users would prefer a conventional map view, and pertains to research question number two.

Downloading hotspots from a server

In addition to the basic map, the user should be able to download hotspots. These hotspots represent context and path information, and should be easily accessible for multiple users and user groups. This feature is somewhat linked to research question number four, as a day-to-day aid for visually impaired users could benefit from a central repository of hotspots.

Sonification of User Context

Much like the visual sense, hearing is used to a great extent when navigating. The noises emitted by our environment are used to navigate by and to place oneself in a geographical context. For example, traffic noise paired with the sound of bustling humans and street musicians would most likely place you in the context of a city at daytime. There are, however, several ways that the same sound could be interpreted, based on distance, timbre and quality. The very same sounds would place you in the outskirts of a town centre, or in a park nearby. Objects and landmarks may also emit characteristic noises; a park may have a fountain which produces sounds of flowing water, and the tolling of bells may be heard from a church building. There are, however, many places and objects that do not emit a characteristic sound. The audio alert can in these cases reflect the type of hotspot; whether it is a library, a concert hall or a nearby friend.

For the purpose of this thesis, the application should be able to play simple sounds whenever a user enters a predefined geographical area. This functionality is used to assess research questions number one, two and three.

Tactile feedback for User Context

Tactile feedback is used when a silent alert is appropriate, or when the user is located in a noisy environment. Alerts are differentiated by using multiple vibration patterns - much like morse code.

For the purpose of this thesis, the application should be able to alert the user with the vibrotactile feature of the phone. This functionality is used to assess research questions number one and three.

3.3 Prototype Design and implementation Phases

Using principles from Agile Development, the design and implementation process goes as follows:

1. Prototype design
2. Prototype implementation
3. Live testing of prototype
4. Redesign based on user feedback (first iteration)

3.3.1 Server Communication

The mobile client should be able to communicate with a server that provides information about alerts, and context. For this thesis, a simple geoRSS-feed with additional parameters will be used. This can be seen in figure 3.3.1.

```
<?xml version="1.0"?>
<rss version="2.0" xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
        <channel>
                <item>
                        <title>punkt1</title>
                        <link></link>
                        <description></description>
                        <icon></icon>
                        <geo:lat>59.92665100189946</geo:lat>
                        <geo:long>10.742085608546008</geo:long>
                        <ubimap:radius>10</ubimap:radius>
                        <ubimap:nudge>2</ubimap:nudge>
                        <ubimap:nudgesound>1</ubimap:nudgesound>
                </item>
        </channel>
</rss>
```

Figure 3.1: GeoRSS feed with additional fields

Chapter 4

UbiMap Implementation

This chapter describes the implementation of the UbiMap client and server.

4.1 Client Hardware

When working with mobile devices, you need to make many choices. First and foremost is choosing the hardware you would like to use. For this thesis, the hardware needed to include a GPS unit and a good sound system, in addition to the usual mobile phone features. The HTC G1 was chosen, along with the operating system Android, from Google.

4.1.1 HTC G1

The HTC G1 (also referred to as ”The Googlephone”) is a typical consumer grade mobile phone. It has a capacitive touchscreen, a built-in GPS, and a compass, along with the usual bits of technology like Wi-Fi, bluetooth, and data traffic services.

4.2 Software Platform

4.2.1 Android

The Android mobile operating system[7] is a fairly new addition to the mobile OS ecosystem. It is developed by Google, and has been available as open source since 21 October 2008. Since the release it has seen five major revisions, and at the time of writing it has reached version 2.2.

Figure 4.1: The HTC G1 running UbiMap

The Android OS is based on a Linux kernel, and although it supports native development, most applications run in a virtual machine called Dalvik. The preferred programming language is Java, but the virtual machine does not run Java bytecode; at compile time, Java classes are converted to the Dalvik .dex-format. Dalvik just recently (as of Android 2.2) acquired support for just-in-time compilation.

4.3 Architecture

4.3.1 Overview

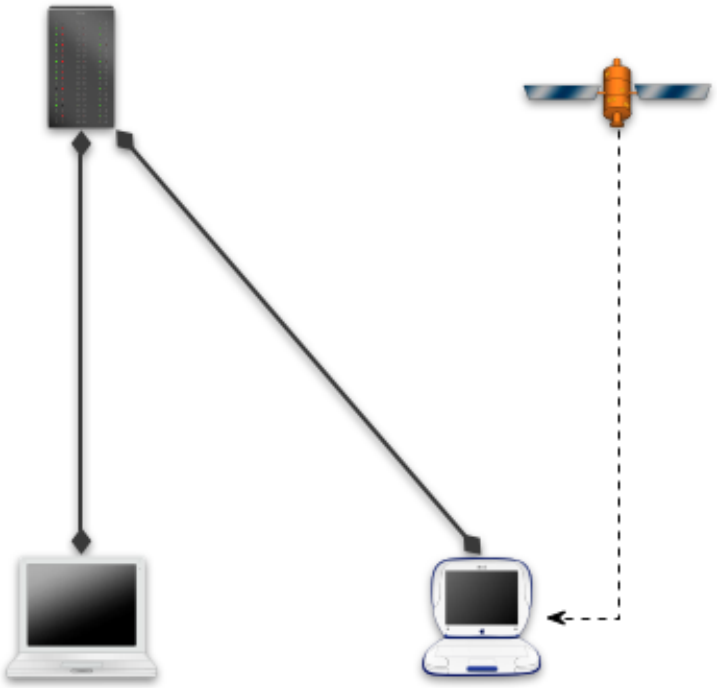


Figure 4.2: Architecture overview

4.4 Server

For the purpose of this thesis, the server only needed to serve GeoRSS-files to the client. As this task only requires a basic web server of some sort, a standard Linux distribution was set up to serve

these files.

4.5 UbiMap Client

The UbiMap client is developed and deployed on an HTC G1 device flashed with an ADP (Android Developer Phone) ROM. This ROM facilitates debugging and testing while developing.

The UbiMap client is roughly made up of four parts:

- A location-aware back-end.
- A Visible map view.
- Hotspots loaded over the net from a GeoRSS-server.
- A Nudge engine that issues alerts when the user enters a hotspot.

4.5.1 Location Back-end

The location back-end retrieves positions from the Android OS, using either cell-based location, or the built-in GPS unit. Every change in location is sent to the main UbiMap application.

4.5.2 Visible Map View

The Map view draws a visible representation of the map and its hotspots. It is built much like a normal map client, and can be panned, zoomed and centered on the current position. There are also two map styles available:

- Vektor.
- Ortho.

The vector map, while rendered as bitmaps, is the most common type to use. Like most maps it shows walkpaths, streets, streetnames, buildings, building numbers, and other information like park areas and street-lane information when relevant (e.g. in one-way streets). The orthophoto-based map view is useful when looking for information that the vector map does not contain (like trees or building features), and is often used as a supplement to the vector-based map.

The visible map view has buttons for zooming in and out, and centering on current position. It also has a graphic for indicating north direction, and a menu for toggling map style and fetching hotspots.

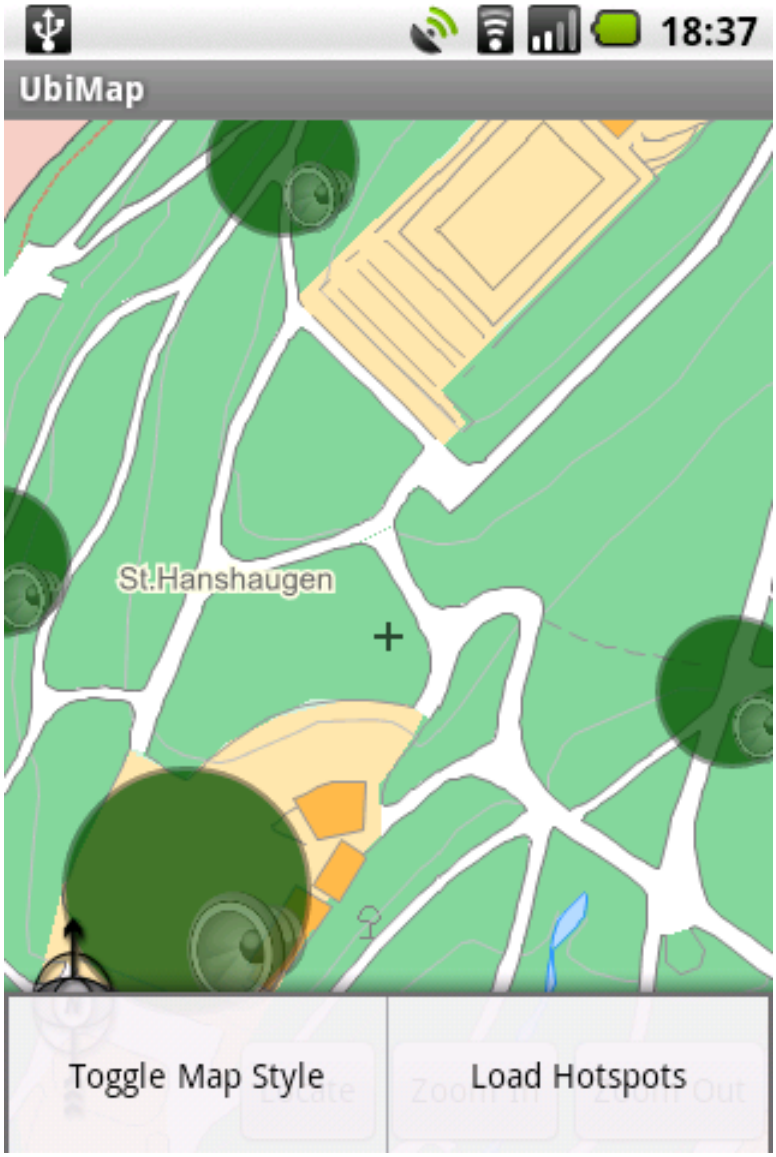


Figure 4.3: The Ubimap view, with menu enabled.

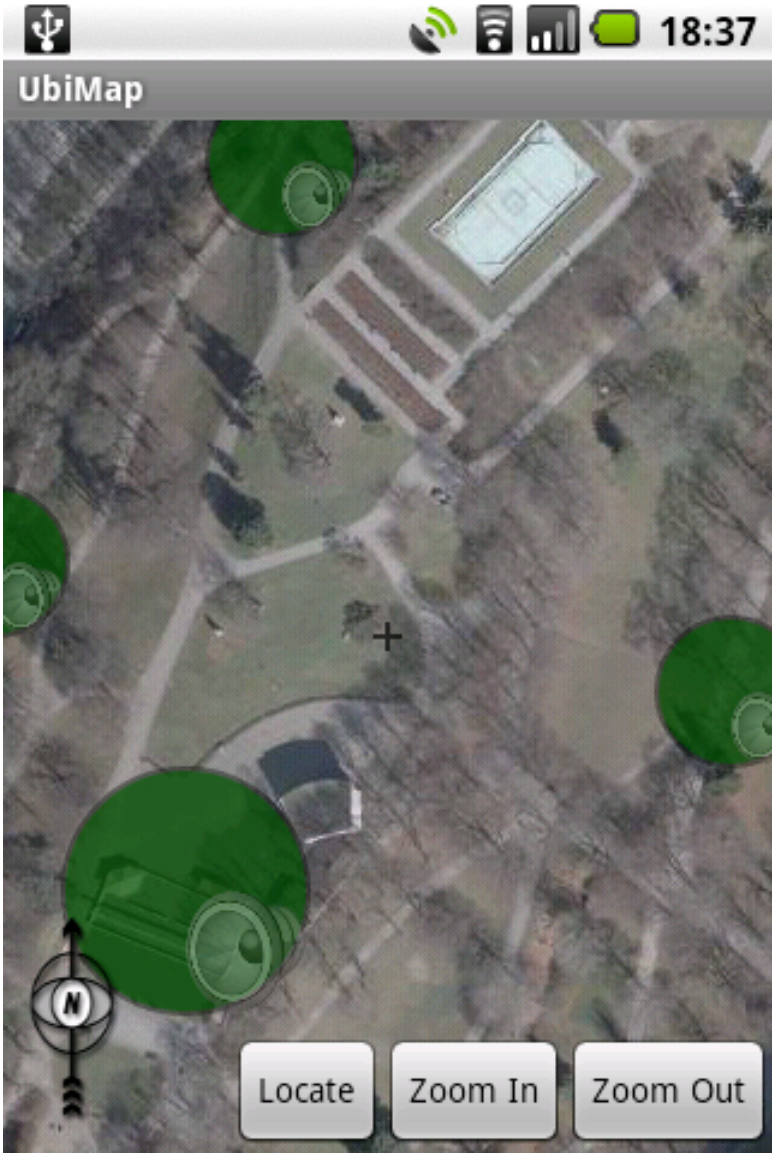


Figure 4.4: The Ubimap view, with orthophoto style enabled..

4.5.3 Hotspots

The hotspots are loaded from a geoRSS server. This implementation paves the way for easy deployment of new hotspots; for example, the user could theoretically subscribe to a feed with hotspots in a particular category. The hotspot server can be operated and maintained by a third party, for example Association of the Blind.

4.5.4 Nudges

When the user enters a hotspot, a nudge is activated. In this implementation, nudges can be one of three types:

- Sound nudge.
- Vibrate nudge.
- Combined nudge.

The sound nudge plays a sound when entering a hotspot. This nudge also has an added sound-number field, which can be used to trigger different sounds in the application. In this implementation, such sounds are preloaded onto the device, and are numbered from zero to nine. The vibrate nudge triggers a silent, vibrating alert, whereas the combined nudge triggers both a sound and the vibrating alert. A visual cue is also shown, displaying the name of the hotspot.

4.5.5 Class Overview

Client-side, the application consists of the following classes (as seen in figure A.1, and figure A.2):

- MapTile - represents a single map tile.
- TileSet - represents a group of tiles.
- TileCache - loads and stores tile images.
- AsyncImageLoader - used to retrieve map images from the tileserver.
- CoordinateConverter - converts coordinates between coordinate systems.
- ProximityIntentReceiver - listens for proximity changes and triggers nudges.
- UbiMap - main class.
- UbiMapView - a screen that represents the main map.
- FeedReader - reads RSS-feeds and creates hotspots.
- HotspotDB - stores hotspots in a local database.
- Hotspot - represents a geographical hotspot.

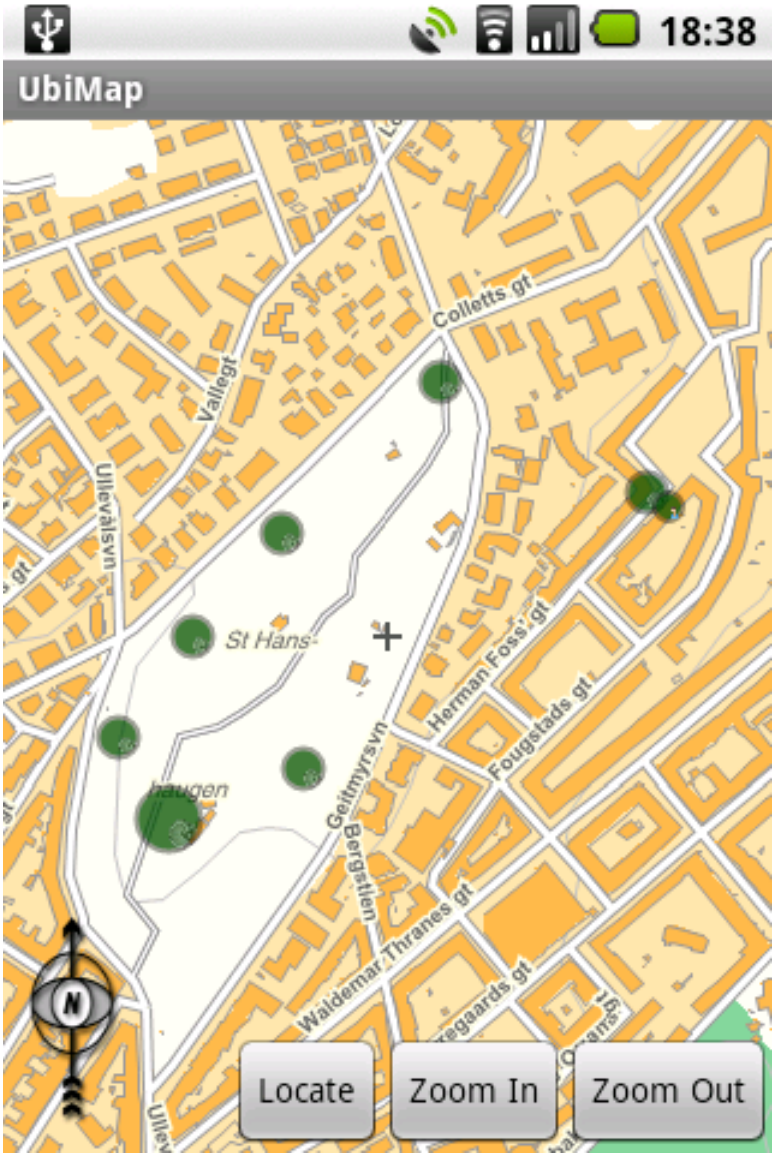


Figure 4.5: The Ubimap view, showing multiple hotspots.

Figure 4.6: The Ubimap view, entering a hotspot.

- POI - represents a single point of interest.
- Settings - stores settings for the UbiMap application.

Chapter 5

Testing

This chapter describes the live user tests, questionnaires, and results of the tests.

5.1 Test environment

All tests will be performed outdoors, as this thesis focuses on outdoor navigation and is GPS-based. To avoid dangerous situations while navigating, the user tests will be performed in a public park, with pathways and no vehicle traffic.

5.1.1 Navigation and Context awareness test

Before the test each participant is familiarized with the prototype application. The main features of the software are explained, and if the user has no sight impediments, the participant will be shown and explained the outline of the test route on a visual map. If the user is unable to view the visual map, he or she will get a brief explanation of the route. The test leader will then walk alongside the participant for the duration of the test. As the focus of the test is to reveal how the participants use sound while navigating in a normal environment, and exploring the use of peripheral interfaces, chatting while walking is encouraged.

Figure 5.1: One of the prototype users

5.2 Questionnaires

5.2.1 Pre-test questionnaire

Before the tests are executed, each participant fills out the pre-test questionnaire. This asks questions regarding how the participant uses sound in everyday navigation, and is used to understand what kind of users the test group is composed of.

Questions one and two have to do with visual impairment, as one of the questions this thesis tries to answer, is whether a sound and vibration driven user interface can be used as a navigational aid for blind or visually impaired users. Questions three to six deals with mobile phone experience, and whether the user is accustomed to mobile navigational applications. These questions pertain to research question number one, which deals with off-the-shelf mobile devices. Question seven pertains to research question number four, and asks the participant if he or she uses any navigational aids other than a map/GPS. Question eight asks the user to describe his or her day-to-day environment with regard to sound noise levels and traffic. This is useful when trying to find the right sounds and feedback levels for the application, and provides some background information for research question number three. Questions nine through eleven ask the user how they rely on their senses while navigating outdoors. These questions are used as background for research questions number three and four, and while this question relates to visually impaired users, it is also interesting to know how users without sight impediments judge their use of sensory input while navigating. Question twelve reveals if the user has other on-the-go listening habits, and is used to shed some light on research question number two.

1. Do you have a visual impairment?

a. No.
b. Yes.

2. If applicable; to what degree are you visually impaired?

a. Blind.
b. Partially blind.
b. Percentage of eyesight:

3. How can you best describe your experience with mobile phones?

a. Experienced.

b. Intermediate.

c. Novice.

4. Does your mobile phone have navigational features or applications?

a. Yes.

b. No.

5. Do you use your mobile phone for navigational purposes?

a. Often (3-4 times per week).

b. Once in a while (1-2 times per month).

c. Seldom (1-2 timer per six months).

d. Never.

6. If applicable; which feature of the navigational application do you use the most?

a. A visual map.

b. Spoken turn-by-turn aid.

c. Tactile aid.

d. Not applicable.

7. Do you use other navigational aids?

a. Cane.

b. Dog.

c. Other - please state.

d. Not applicable.

8. How would you describe your day-to-day environment?

a. Quiet environment with little traffic.

b. Moderate noisy environment with some traffic.

c. Noisy environment with mixed traffic (cars, buses, trams etc.)

9. How important is your sight to you when navigating outdoors?

a. It is my most important guide.
b. I use it as a supplement to my other senses.
c. I do not rely on sight while navigating.

10. How important is your hearing to you when navigating outdoors?

a. It is my most important guide.
b. I use it as a supplement to my other senses.
c. I do not rely on hearing while navigating.

11. How important is your tactile sense to you when navigating outdoors?

a. It is my most important guide.
b. I use it as a supplement to my other senses.
c. I do not rely on tactile information while navigating.

12. Do you listen to music, or use headphones while navigating?

a. Yes.
b. No.

5.2.2 Post-test Questionnaire

Shortly after each test is concluded, each participant fills out a questionnaire regarding how they perceived the application, and which areas could be improved upon. These questions are central to answering the thesis problem statement.

Question one asks the user to describe the test environment with regard to ambient sound levels and general traffic. As this often changes throughout the day, it can affect the application user experience, and the question is posed in order to establish some background information for research question number three. Questions two to four pertain to the application sounds, and are posed to answer research question two and three. Question two and three ask the user if he or she had any problems with the generated sounds; if these were easy to differentiate from other noises, and if the

generated sounds were otherwise disturbing. As this thesis focuses on non-verbal communication, question four asks the user if he or she would prefer spoken directions instead of the generated sounds. Question five and six pertains to the vibrotactile features of the application, asks if these were used, and if these features were interfering with the sound features. These questions pertains to research question number three. Questions seven and eight ask the user if he or she used the visual representation of the route, and if he or she would prefer a plain visual map instead of the sound and vibrotactile features. This question is linked to research question number two. Question nine asks if the user would like to use a similar application as a day-to-day navigational aid, and is linked to research question number two and four. Question number ten asks the user for ideas or comments on improving the application. This question pertains to research question number one, and will hopefully bring in some fresh ideas about non-visual, non-verbal computer interfaces.

1. In what type of environment did you use the prototype application?

a. Quiet environment with little traffic.

b. Moderate noisy environment with some traffic.

c. Noisy environment with mixed traffic (cars, buses, trams etc.)

2. Did you find it easy to differentiate the application sounds from other ambient noises?

a. Yes.

b. No.

3. Did you find the generated sound to be interfering with the task at hand?

a. Yes.

b. No.

4. Would you prefer spoken directions instead of sound queues?

a. Yes.

b. No.

5. Did you make use of the vibrotactile features in the prototype application?

a. Yes.

b. No.

6. Did you find the vibrotactile features in the prototype application to be interfering with the sound features?

a. Yes.

b. no.

7. Did you use the visual map?

a. Yes.

b. no.

8. Would you prefer a visual map when navigating?

a. Yes.

b. no.

9. Could you use a similar application as a day-to-day navigational aid?

a. Yes.

b. No.

c. Maybe.

10. In what way do you think the application could be improved? Please describe in what way you would alter the application.

description:

5.3 Test results

5.3.1 Pre-test answers

Subject	**Do you have a visual impairment?**
1	b
2	a
3	a
4	a
5	b
6	b

Table 5.1: Pre-test question one answers

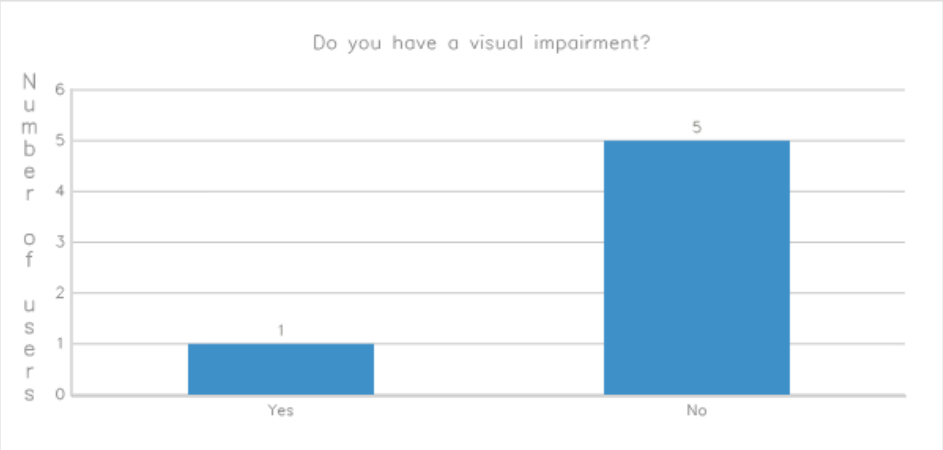


Figure 5.2: Pre-test question one chart

Subject	**If applicable; to what degree are you visually impaired?**
1	a
2	na
3	na
4	na
5	na
6	na

Table 5.2: Pre-test question two answers

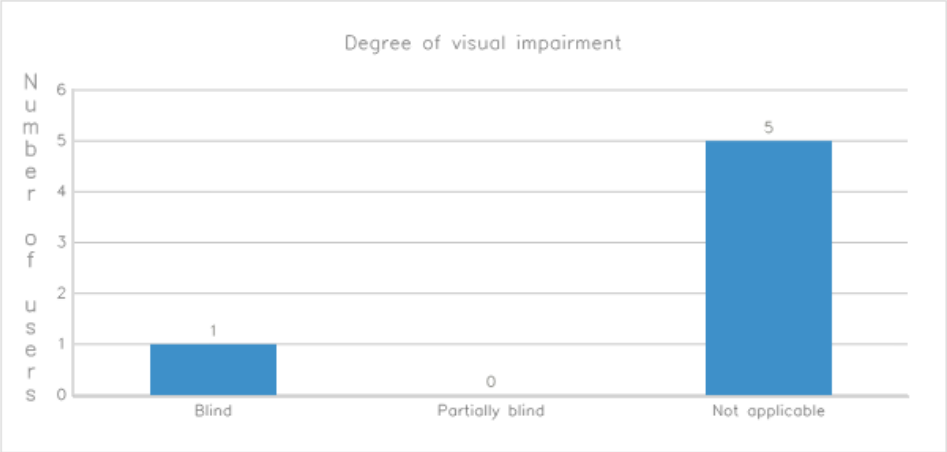


Figure 5.3: Pre-test question two chart

As shown in figure 5.3, one of the test users was blind, while the other five users had no sight impediments.

Subject	**How can you best describe your experience with mobile phones?**
1	b
2	a
3	b
4	b
5	b
6	b

Table 5.3: Pre-test question three answers

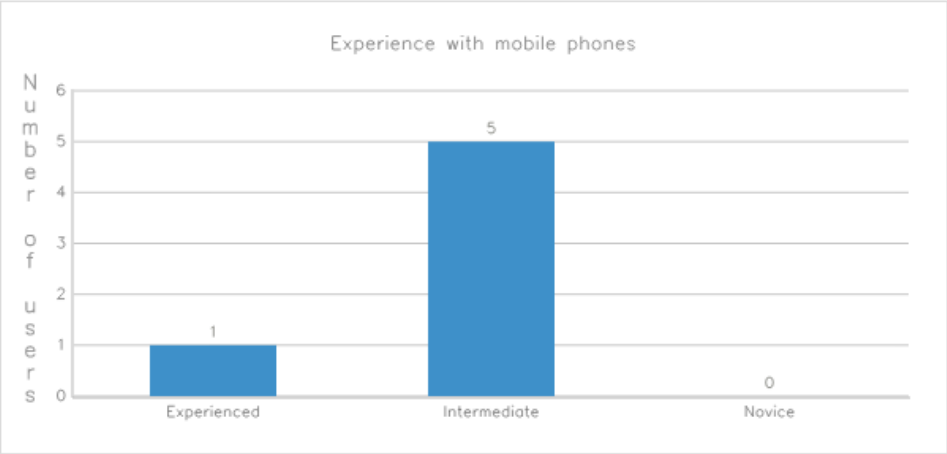


Figure 5.4: Pre-test question three chart

Nearly all of the users describe themselves as intermediate users of mobile phones, as shown in figure 5.4

Subject	**Does your mobile phone have navigational features or applications?**
1	a
2	b
3	a
4	b
5	a
6	b

Table 5.4: Pre-test question four answers

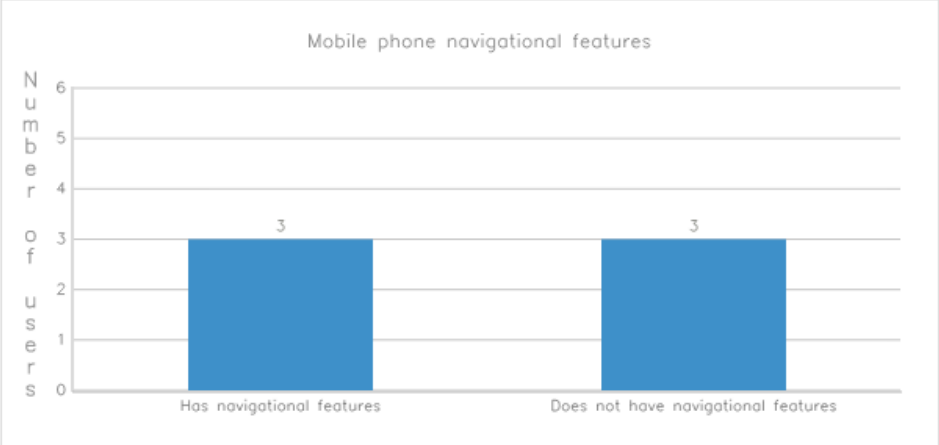


Figure 5.5: Pre-test question four chart

As seen in figure 5.5, half of the test users had navigational features included in their mobile phones. This is a typical figure, as GPS systems are increasingly popular in mobile phones.

Subject	**Do you use your mobile phone for navigational purposes?**
1	d
2	d
3	b
4	d
5	c
6	d

Table 5.5: Pre-test question five answers

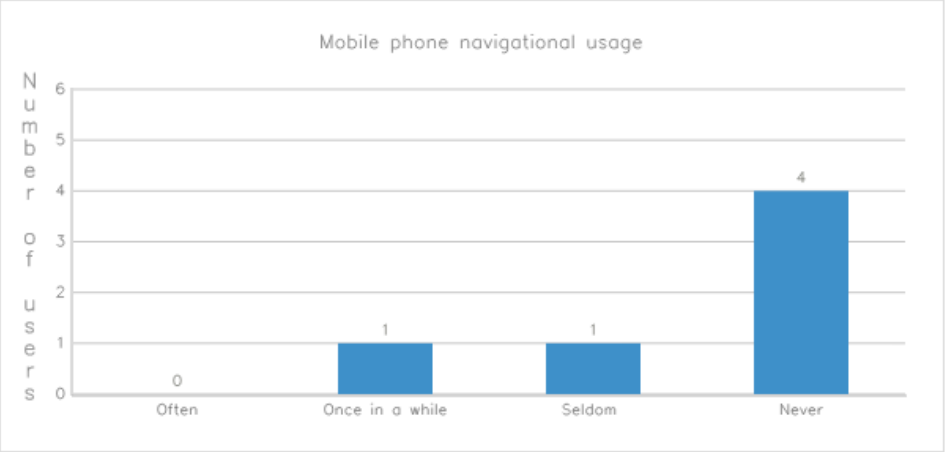


Figure 5.6: Pre-test question five chart

As seen in figure 5.6, the test users do not use navigational features often.

Subject	**If applicable; which feature of the navigational application do you use the most?**
1	d
2	d
3	a
4	d
5	a
6	d

Table 5.6: Pre-test question six answers

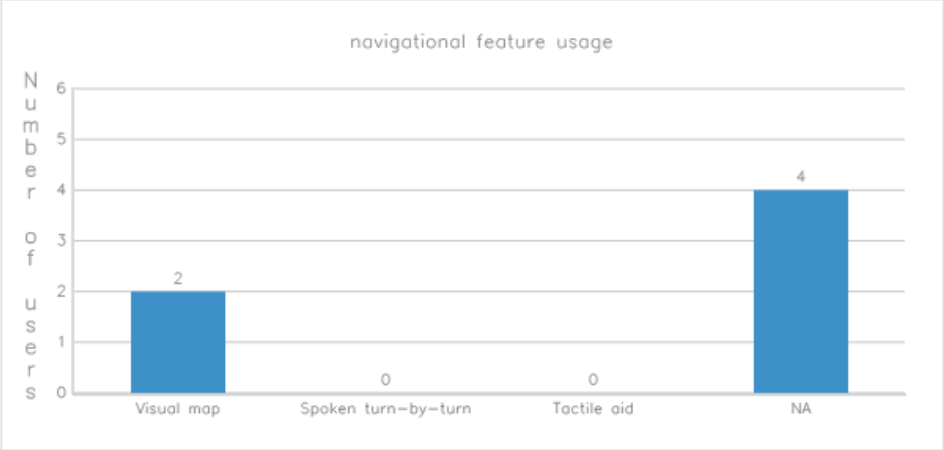


Figure 5.7: Pre-test question six chart

The users that use the navigational features of their cellphones tend to use a visual map, as seen in figure 5.7.

Subject	**Do you use other navigational aids?**
1	a
2	d
3	d
4	d
5	d
6	d

Table 5.7: Pre-test question seven answers

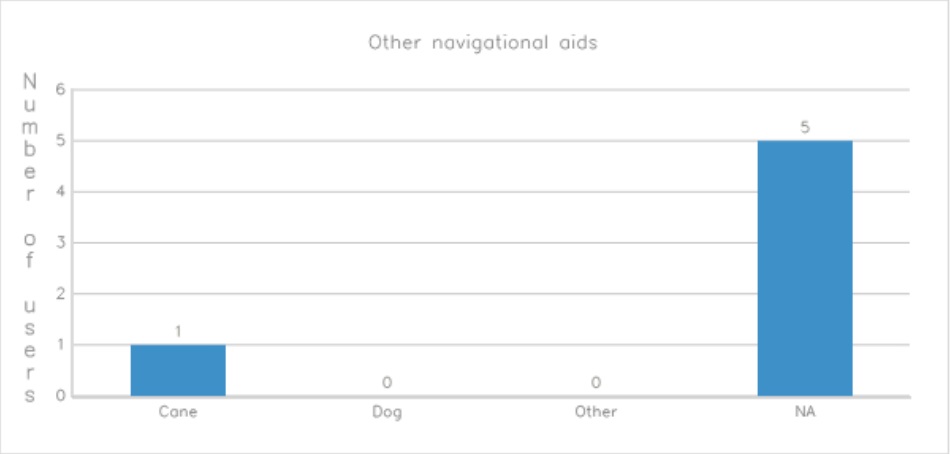


Figure 5.8: Pre-test question seven chart

One of the users, as seen in figure 5.8, uses a cane when navigating. The five remaining users do not use extra aids.

Subject	**How would you describe your day-to-day environment?**
1	c
2	c
3	c
4	b
5	b
6	b

Table 5.8: Pre-test question eight answers

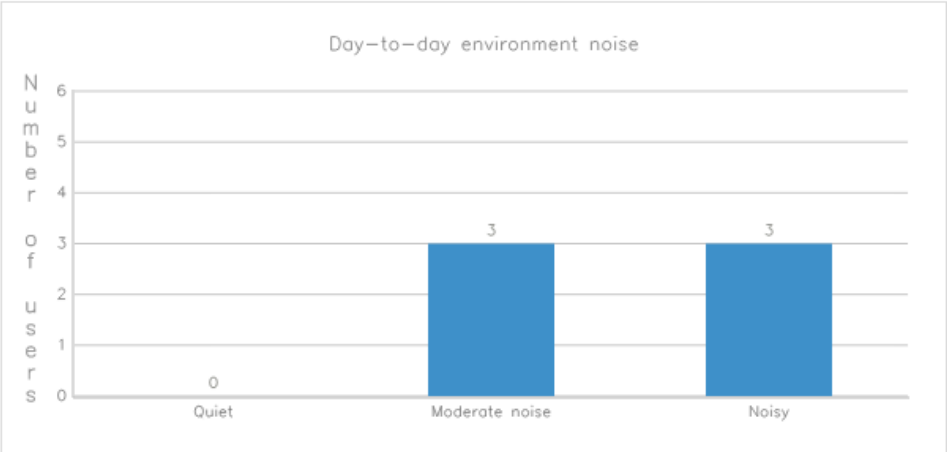


Figure 5.9: Pre-test question eight chart

All of the users described their day-to-day environment as either noisy or moderately noisy, as seen in figure 5.9.

Subject	**How important is your sight to you when navigating outdoors?**
1	c
2	a
3	a
4	a
5	a
6	b

Table 5.9: Pre-test question nine answers

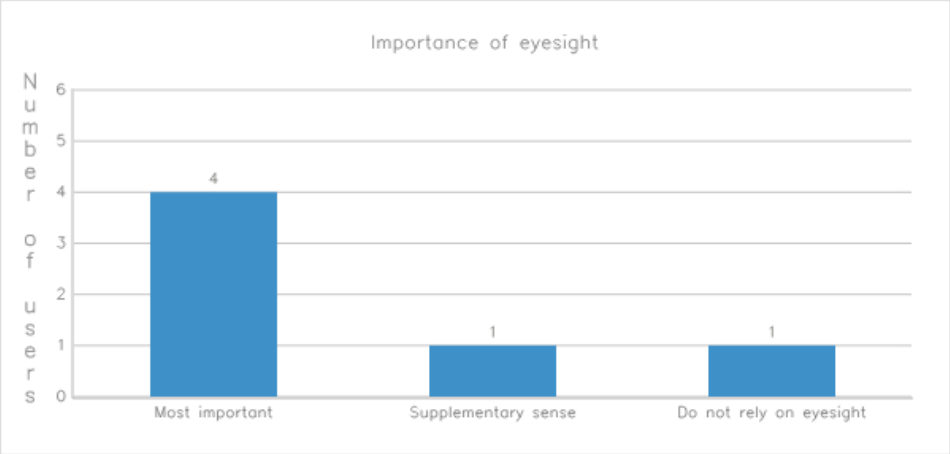


Figure 5.10: Pre-test question nine chart

As most of the test users were not visually impaired, they relied mostly on their visual sense when navigating. Seen in figure 5.10.

Subject	**How important is your hearing to you when navigating outdoors?**
1	a
2	b
3	b
4	b
5	b
6	b

Table 5.10: Pre-test question ten answers

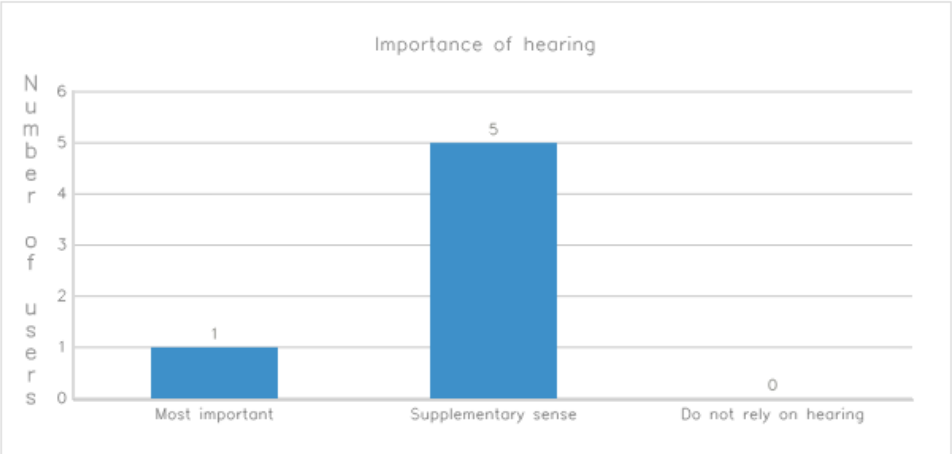


Figure 5.11: Pre-test question ten chart

Most users stated that they used hearing as a supplementary sense when navigating, as seen in figure 5.11.

Subject	**How important is your tactile sense to you when navigating outdoors?**
1	b
2	c
3	c
4	b
5	c
6	b

Table 5.11: Pre-test question eleven answers

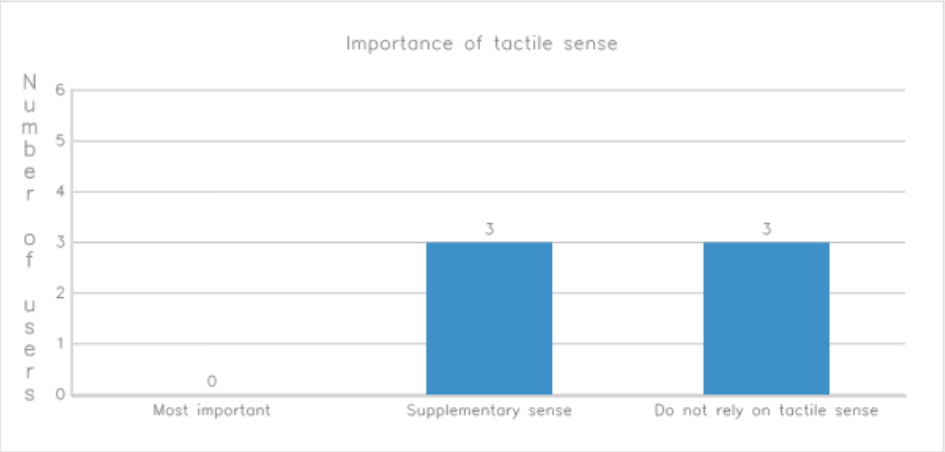


Figure 5.12: Pre-test question eleven chart

As seen in figure 5.12, none of the users deemed their tactile sense the most important when navigating.

Subject	**Do you listen to music, or use headphones while navigating?**
1	b
2	a
3	b
4	a
5	b
6	a

Table 5.12: Pre-test question twelve answers

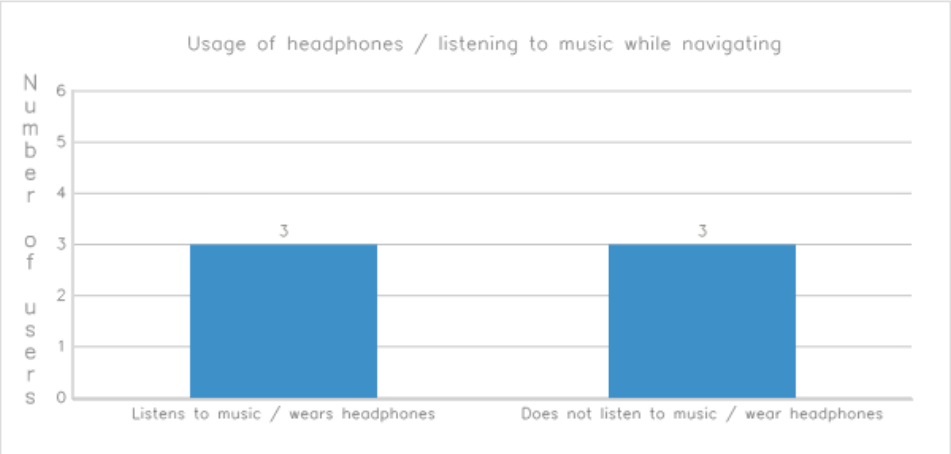


Figure 5.13: Pre-test question twelve chart

Figure 5.12 shows that half of the group of users listens to music while navigating.

5.3.2 Post-test answers

Subject	**In what type of environment did you use the prototype application?**
1	b
2	a
3	a
4	b
5	a
6	a

Table 5.13: Post-test question one answers

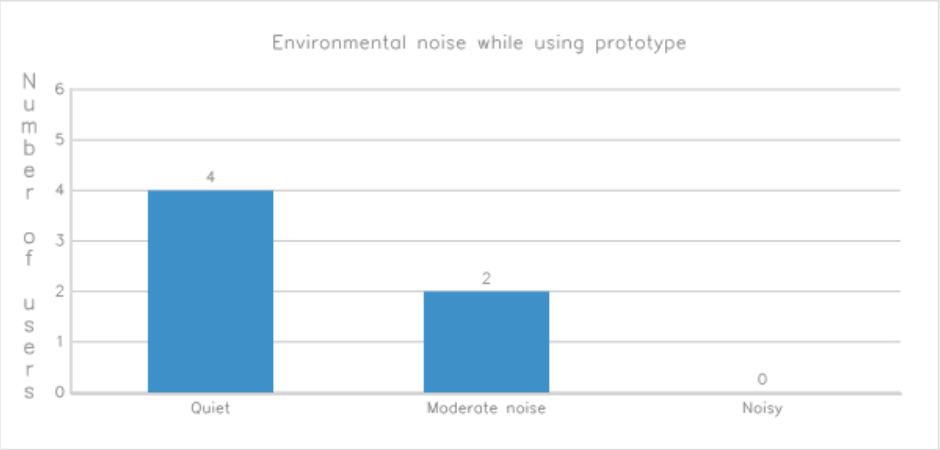


Figure 5.14: Post-test question one chart

As seen in figure 5.14, users described the test environment as either quiet or moderately noisy.

Subject	**Did you find it easy to differentiate the application sounds from other ambient noises?**
1	a
2	a
3	a
4	a
5	a
6	a

Table 5.14: Post-test question two answers

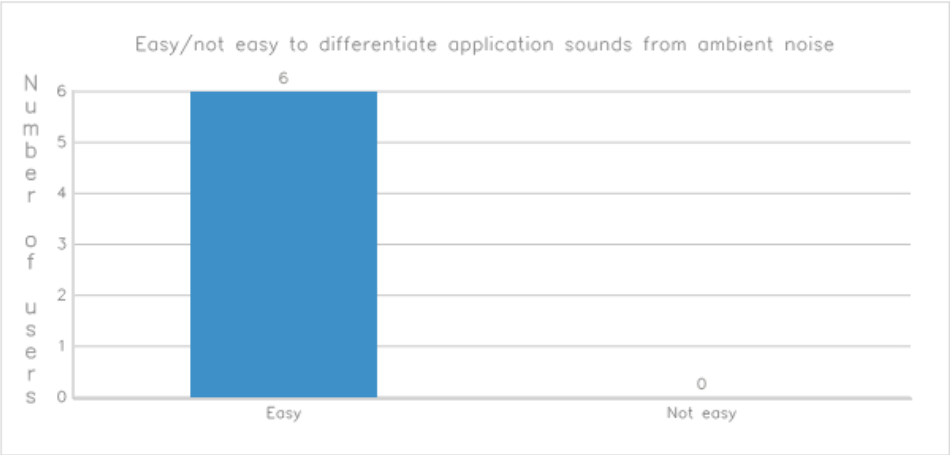


Figure 5.15: Post-test question two chart

All of the users found the application sounds to be easy to differentiate from ambient sounds. This can be seen in figure 5.15.

Subject	**Did you find the generated sound to be interfering with the task at hand?**
1	b
2	b
3	b
4	b
5	a
6	b

Table 5.15: Post-test question three answers

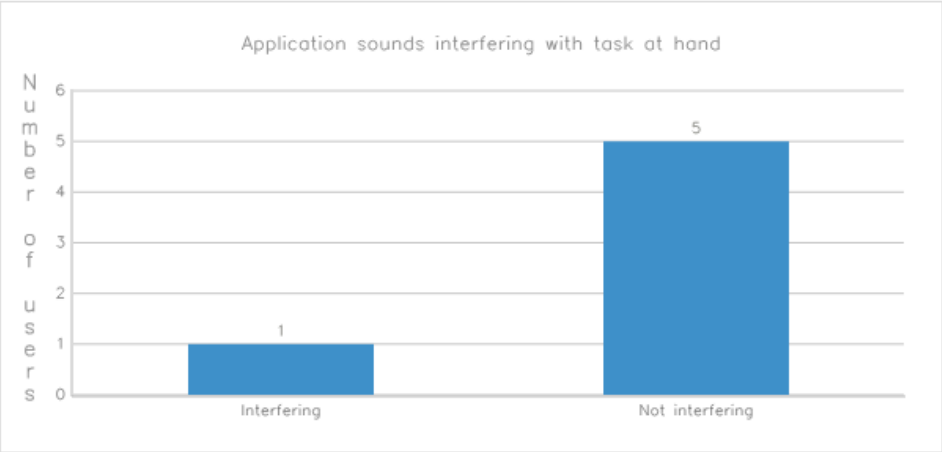


Figure 5.16: Post-test question three chart

One of the users found the generated sound to be interferring with the task at hand, as seen in figure 5.16.

Subject	**Would you prefer spoken directions instead of sound queues?**
1	b
2	b
3	b
4	a
5	b
6	b

Table 5.16: Post-test question four answers

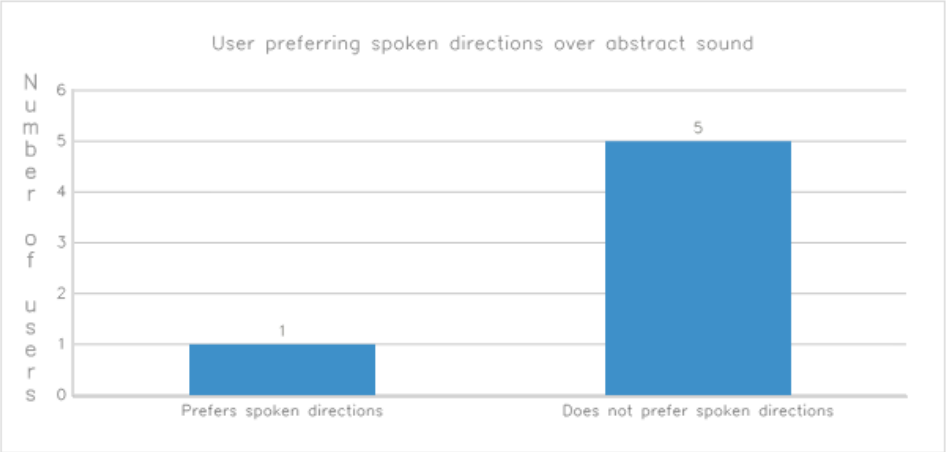


Figure 5.17: Post-test question four chart

One of the users would prefer spoken directions instead of abstract sounds, as seen in figure 5.17.

Subject	**Did you make use of the vibrotactile features in the prototype application?**
1	a
2	b
3	b
4	b
5	a
6	a

Table 5.17: Post-test question five answers

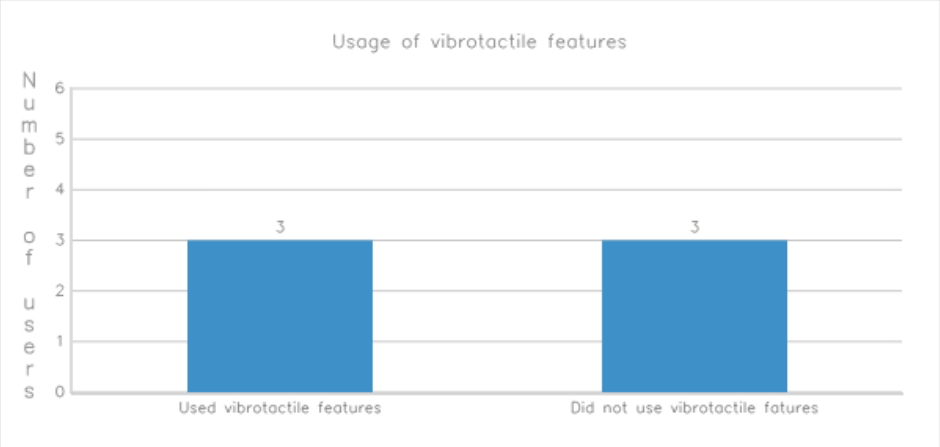


Figure 5.18: Post-test question five chart

Three of the users tried out the vibrotactile features of the application, while the remaining three did not make use of this feature (figure 5.18).

Subject	**Did you find the vibrotactile features to be interferring with the sound features?**
1	b
2	b
3	b
4	b
5	b
6	b

Table 5.18: Post-test question six answers

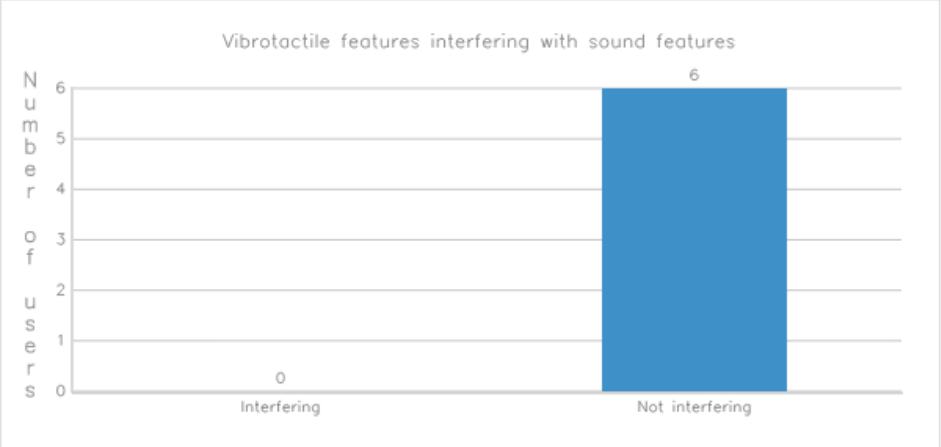


Figure 5.19: Post-test question six chart

None of the users found the vibrotactile features to be interferring with the sounds from the application, as seen in figure 5.19.

Subject	**Did you make use of the visual map?**
1	b
2	a
3	a
4	a
5	a
6	b

Table 5.19: Post-test question seven answers

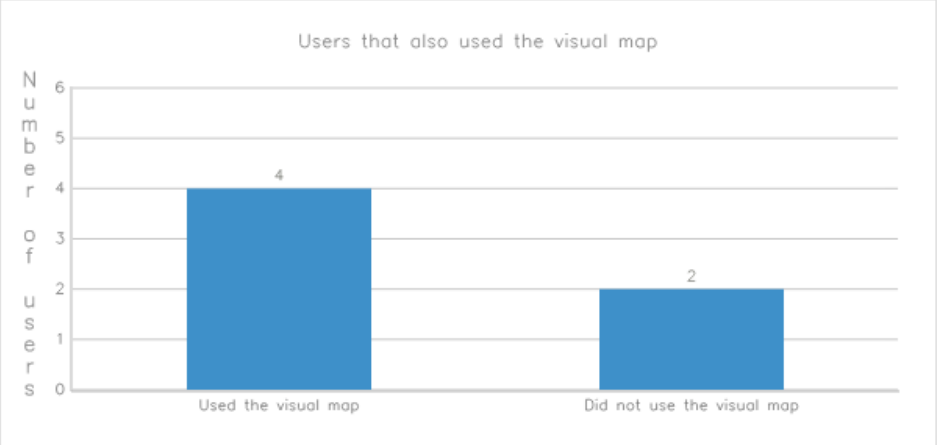


Figure 5.20: Post-test question seven chart

As seen in figure 5.20, four of the users also made use of the visual representation of the path.

Subject	**Would you prefer a visual map?**
1	b
2	b
3	a
4	a
5	a
6	b

Table 5.20: Post-test question eight answers

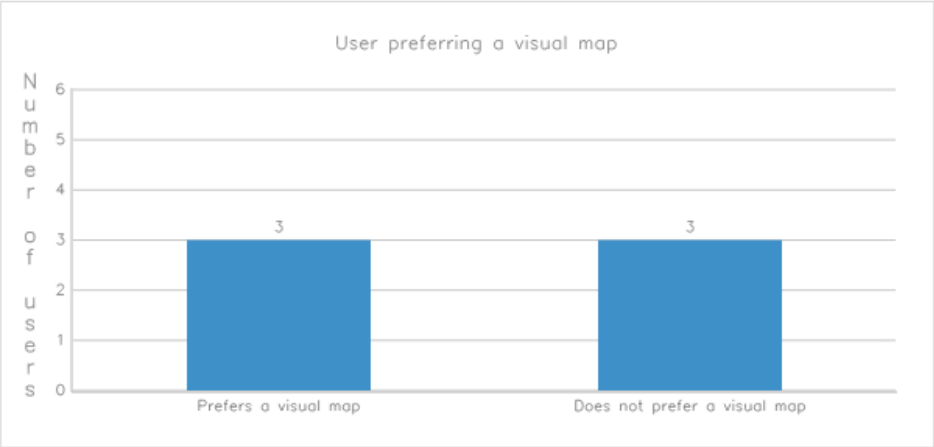


Figure 5.21: Post-test question eight chart

Half of the users would prefer a visual map over the sound and vibrotactile features, as seen in 5.21.

Subject	**Could you use a similar application as a day-to-day navigational aid?**
1	a
2	a
3	c
4	a
5	a
6	a

Table 5.21: Post-test question nine answers

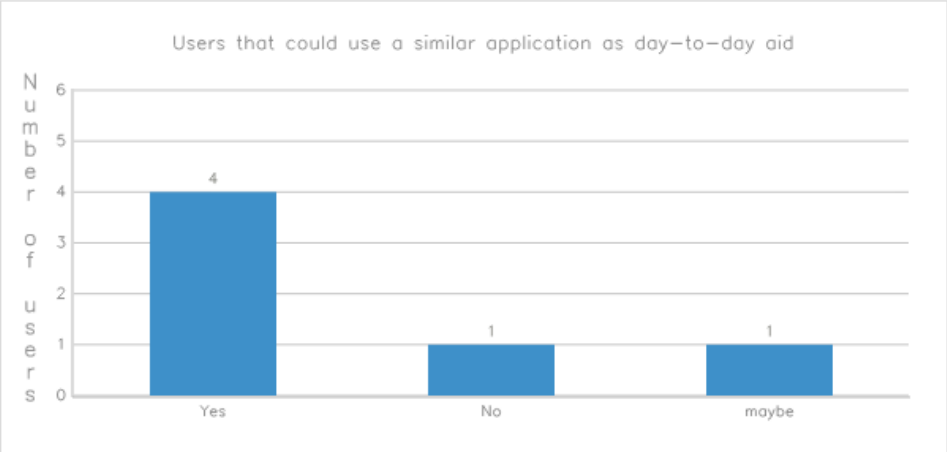


Figure 5.22: Post-test question nine chart

As seen in figure 5.22, most of the users stated that they could use a similar application on a day-to-day basis. One user would not use such an application.

Subject	**In what way do you think the application could be improved? Please describe.**
1	Non-blocking speaker. Better point precision, and obstacle/route identification.
2	Define the sounds better, otherwise a good application.
3	Some sounds are disturbing, needs a comfortable headphone
4	Alert when going in wrong direction. Also if route has been modified.
5	Shorter, more concise sounds.
6	NA

Table 5.22: Post-test question ten answers

As seen in table 5.3.2, the test users had several comments and ideas to further improve the prototype application. One of the most interesting was the need for an open-air speaker solution. This would allow for ambient noises to come through better, while still providing good feedback from the application itself. Many comments were made on the sounds, and several users commented that these should be shorter, and more concise. Other users wanted better precision, to allow for obstacle detection and avoidance.

Chapter 6

Findings & Discussion

This chapter describes and discusses the thesis findings. The background for this thesis, along with the empirical information from the user tests, will be examined and analysed, in order to answer the thesis' central questions.

6.1 Question One

In section 1.0.1 of the introductory chapter, four questions were presented. The first one was this:

1. How can a mobile, commercial off-the-shelf device be used to convey location context, without using a visual or spoken interface?

In section 2.7 of this thesis, I have presented several projects that are intended as non-visual navigational aids. A lot of interesting work has already been done, like the SWAN-project[34]. However, most of these projects focus on equipment that is not commonly found amongst the public. In this thesis I have tried to use off-the-shelf, common mobile phones to address the problem. As mobile technology has matured, and the hardware has become more powerful, it has become easier to develop rich navigational applications. I have demonstrated this by implementing a prototype application that uses sound and tactile feedback to provide the user with context awareness and navigational aid. The implementation is based on a consumer-grade device and mobile operating system, that is easily available to the public. I have relied on GPS features to provide the basic location, and a central service to deliver nudge and hotspot information. The implementation also

included a conventional visual map, to explore the basic differences between the interface types.

The projects presented in section 2.7 include The Sonic Torch, The Binaural Sonic glasses, The 'K' Sonar Cane, GuideCane, BATS, SIKTE and SWAN. With the exception of BATS, which relies on standard computer hardware, the common denominator for these projects is that they are all based on specialized equipment. Assistive technology has a long history of specialized solutions, and with good reason. Until recently, off-the-shelf hardware did not include the sensory equipment to facilitate such solutions. Furthermore, hardware that did include usable sensory equipment was often bulky and unfit for pedestrian day-to-day use.

The test group used for this thesis consisted of six users. Although this is a small group, the feedback from the users sheds some light on the thesis problem statement and central questions. Although all of the users answer that they are either experienced or expert mobile phone users (figure 5.4), only half of them have phones with navigational features (figure 5.5). The trend in mobile phones is towards tight location integration, but mobile phones with these features have yet to become ubiquitous. With strong incentives for mobile phone manufacturers to provide solid location features, used for example in location-aware advertising, it should be safe to say that during the next few years, phones without location features will become a small minority. This paves the way for assistive technology implemented on off-the-shelf commercial hardware.

Another aspect from the user tests is that none in the test group uses the navigational features of their phone often (figure 5.6). This could be explained by the user group consisting of people that do not need navigational aids frequently. Most people don't need navigational aids often, as they tend to frequent the same routes and places on a day-to-day basis. It would be interesting to make the same observations on a user group that relies on navigational aids for professional use. Furthermore, all of the test subjects that utilize a navigational application tend to use a visual map. Turn-by-turn navigation is associated with motorized transportation, and is not widely supported for pedestrians.

6.2 Question Two

The second question posed in section 1.0.1 was:

2. Are sound based interfaces suited for mobile pedestrian navigation?

Pedestrian navigation is often complex, compared to driving a car, or using other means of transportation. While in a crowded city environment, pedestrians often have designated areas that separate them from other traffic; in reality they have few limits as to what kind of routes that can be used. It is therefore difficult to design good solutions that address this problem and make sense to the user. The prototype application developed for this thesis uses sound and vibrotactile features to alert the user to navigational choices and environmental context. The basic idea is that a sound based interface could be used alongside normal pedestrian navigation without being too taxing on the user.

As the post-test question number twelve reveals, 50% of the users listen to music or wear headphones when navigating outdoors (figure 5.13). This poses a problem when using a sound based navigational application. However, as the navigational features are deemed to be more important than music listening, the problem can be minimised by manipulating the music volume when a nudge occurs. This is a frequently-used solution on phones, for example when receiving an SMS message, but it is important that the application sounds are short and to-the-point, in order to not irritate the user.

It should be noted that while the UbiMap prototype demonstrated that an auditory display coupled with vibrotactile feedback provided the users with directional and contextual nudges, another problem that should be addressed is that of obstacle detection. In the prototype application, all nudges must be digitally prepared in advance, and this poses a problem for real-time obstacle detection. Currently, off-the-shelf mobile solutions does not include ultra-sound emitters that can be used to provide obstacle detection and real-time environment sonification, as seen in The Sonic Torch. To a certain degree, image analysis could be used to spot obstacles, and generate an environment sonification, but this would most likely prove cumbersome for many users.

The test users also provided suggestions for improvements to the prototype application (table 5.3.2). The most frequent suggestion for improvement pertained to the generated sounds. Some found these distracting or not to their liking. Others would have liked shorter, more concise sounds. As with any data presentation, it is important to find a pragmatic solution, that also takes into account the aesthetics of the interface. That is, the system should be able to efficiently convey information to the user at the same time as not being perceived as 'ugly'. Question ten of the post-user test questionnaire points to future work, and will be addressed further in the last chapter.

The user tests show that 50% of the users preferred a visual map over the sound and vibrotactile

interface (figure 5.21). This could be because the users are familiar with conventional maps, and also because the prototype sound implementation did not offer as much detail as the visual view. It is also important to note that there are some fundamental differences in approach between a conventional map and a sound based representation. Sounds are excellent for providing short nudges and context information instantly, also while the user navigates, but the approach does poorly at describing a whole route or an area.

6.3 Question Three

The third question posed in section 1.0.1 was:

3. Do interfaces based on sound or tactile feedback provide a low level of interference with other tasks, in a navigational context?

Before answering this question, it is important to have some background information about the user group. As all the test participants live in an urban environment, all of them described their day-to-day environment as either 'moderately noisy' or 'noisy' (figure 5.9). While a noisy environment can be seen as negative, the noise is often useful in a navigational context (if you can hear the bus approaching down the street, you know when to start running in order to catch it). It is therefore essential to keep as much of the ambient noise as possible, and provide additional sound or tactile nudges when appropriate.

As most of the users did not have a visual impediment, nearly everyone in the test group deemed their visual sense to be their most important one when navigating outdoors (figure 5.10). One of the users deemed the visual sense to be supplementary, while the blind participant did naturally not rely on visual information. Except for the blind participant, hearing was deemed to be a supplementary sense (figure 5.11). The tactile sense was either deemed to be unimportant, or supplementary to the other senses (figure 5.12). It is interesting that 50% of the test group stated that they do not rely on their tactile sense.

Four users described the test environment as quiet, and two users described it as moderately noisy (figure 5.14). This stands in contrast to the answers given in the pre-test questionnaire, question eight(figure 5.9). The park was perceived to be mostly quiet, with a low level of ambient noise.

One of the most obvious advantages to sound and tactile based interfaces is that these can be

operated while staying in the periphery of the user. For users with no visual impediments, this type of interaction can facilitate effective use of computer interfaces while the user is concentrating on other tasks. As we see from the user tests, none of the users found the generated sounds hard to differentiate from ambient noises (figure 5.15). Also, only one of the users found the sounds to be interfering with the task at hand (figure 5.16), and only one of the users would like to get spoken directions instead of the generated sounds (figure 5.17). While only three persons in the test group made use of the vibrotactile features of the prototype application (figure 5.18), they did not find these to be interfering with the sound features (figure 5.19). This is encouraging for the feasibility of sound and vibrotactile based interfaces for use in pedestrian navigation, as it shows that such interfaces can indeed be used without interfering with other tasks.

6.4 Question Four

The fourth question posed in section 1.0.1 was:

4. How can a sound-based navigational interface help in a day-to-day situation for visually impaired users?

When designing a sound based interface for blind or visually impaired users, it is important to remember that sound is one of their most important senses. Test subject number one, who is blind, stated that hearing was the most important sense when navigating (table 5.3.1). Also the tactile sense was considered important (table 5.3.1). Therefore it is crucial to not overload the remaining senses with additional information. One of the complaints from the users was that the sounds used in the test were too long, and could be shortened without losing context recognition. Test subject number one also noted the need for an open-air speaker that would not interfere with normal hearing (table 5.3.2).

For blind or visually impaired users, a system like the prototype application could provide for some added flexibility. It does not replace any of the usual aids like a cane or a guide dog, but it can provide the user with extra information about the environment or the route. Obstacle detection or interest-based contextual nudges are two ways that could aid blind or visually impaired users. It should be noted that GPS accuracy is limited, and should be used with caution when navigating by foot. Obstacle detection in the prototype application is also limited to digitally prepared obstacles.

Chapter 7

Conclusion and Future Work

This chapter sums up the thesis, and tries to answer the central questions posed in the introductory chapter.

7.1 Conclusion

The purpose of this thesis was to examine the use of non-visual and non-verbal computer interfaces. Furthermore it was to be set in the context of a mobile, navigational application. The thesis also focuses on visually impaired users, with the possibility of developing a day-to-day aid for context awareness and pathfinding. The first research question that was posed in section 1.0.1 was:

1. How can a mobile, commercial off-the-shelf device be used to convey location context, without using a visual or spoken interface?

The thesis answers this question by examining existing technology, the field of sonification, HCI theory and consumer devices. Furthermore, a prototype was developed as a solution to this question. The prototype application shows that non-visual, non-spoken interfaces can be implemented on consumer-grade, off-the-shelf mobile devices. This is partly possible thanks to the rapid development in hardware features, and the integration of location end movement sensors into modern mobile phones.

The second question posed in section 1.0.1 was:

2. Are sound based interfaces suited for mobile pedestrian navigation?

The thesis answers this question by using the prototype application in a live user test. Six users tried out the application, and answered questionnaires before and after the test. Based on the feedback from these tests, it seems that sound based interfaces, thanks to a low level of distraction, are suitable for mobile pedestrian navigation.

The third question posed in section 1.0.1 was:

3. Do interfaces based on sound or tactile feedback provide a low level of interference with other tasks, in a navigational context?

The thesis answers this question by using the prototype application in a live user test. The feedback provided from this test showed that users perceived the audio and vibrotactile interface to provide a low level of interference with other tasks.

The fourth question posed in section 1.0.1 was:

4. How can a sound based navigational interface help in a day-to-day situation for visually impaired users?

The thesis answers this by examining existing assistive technology for visually impaired users. Features like obstacle detection, context sonification, vibrotactile feedback and current location technology have been presented.

The prototype developed for this thesis were also tested on a blind user, with positive feedback. Although the application can provide assistance for blind users, it is no substitute for traditional aids, like a white cane or a guide dog.

7.2 Future Work

The prototype application developed for this thesis is a basic implementation of a non-visual navigational application. Given the features of the hardware platform, such as a compass and an accelerometer, the application could be developed further. With inspiration from the Sonic Torch[10] and others, and leveraging the new hardware features, the application could be improved in many ways.

Directional Sound

Given the new directional hardware features of modern phones, these sensors could be used to position sounds in 3D space. This could allow the user to point the phone in any direction in order to discover obstacles, waypoints or even user defined objects. This could also be used while walking, to provide the user with path correction using left or right speakers.

Collaboration and Integration

A collaborative system could be implemented to allow for sharing of hotspots, or routes. This would allow for user groups to share hotspots, or routes within their category of interest. An alternative to this would be some sort of integration with an existing map service, like OpenStreetMap[9], which has a lot of user-generated content already.

Standardize Sound Nudges

Like visual cartography, a larger scale system would need some kind of standardization of sounds. For example, public buildings of different categories could be given their own distinct sounds.

Ambient noise detection

The possibility to adapt the user interface to local ambient noises is an interesting one. In a particularly noisy environment, a different set of sounds could be used, or the sound volume could be raised. Such an implementation would require use of the phone's microphone to analyse the ambient sound levels while using the application.

References

[1] Chai3d. http://www.chai3d.org.

[2] Google mail goggles. http://gmailblog.blogspot.com/2008/10/new-in-labs-stop-sending-mail-you-later.html.

[3] Mouse (computing). *Wikipedia*.

[4] Norkart as. http://www.norkart.no.

[5] Statens kartverk. http://www.statkart.no.

[6] Electronic road pricing. *Wikipedia*, September 2008.

[7] Android (computing). *Wikipedia*, 2010.

[8] Galileo (satellite navigation). *Wikipedia*, November 2010.

[9] Openstreetmap, 2010. http://www.openstreetmap.org/.

[10] The sonic torch, 2010. http://www.batforblind.co.nz/history.php.

[11] Menon APG and Keong CK. The making of singapores electronic road pricing systemscale collaborative systems. International conference on transportation into the next millennium, September 2006.

[12] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. IEEE Infocom, March 2000.

[13] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Principles behind the agile manifesto. http://agilemanifesto.org/principles.html.

[14] Genevieve Bell and Paul Dourish. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Pers Ubiquit Comput*, April 2006.

[15] Johann Borenstein and Ivan Ulrich. The GuideCane - A Computerized Travel Aid for the Active Guidance of Blind Pedestrians. IEEE International Conference on Robotics and Automation, Apr 1997.

[16] John Brabyn. Technology as a support system for orientation and mobility - Orientation and Mobility for Blind People. *American Rehabilitation*, 1997.

[17] Vannevar Bush. As we may think, July 1945.

[18] Gregory Kramer; Chair, Bruce Walker; Project Coordinator, Terri Bonebright, Perry Cook, John Flowers, Nadine Miner, and John Neuhoff. Sonification Report: Status of the Field and Research Agenda. 1997.

[19] Hae Don Chon, Sibum Jung, and Sang Won An. RFID for Accurate Positioning. *Journal of Global Positioning Systems*, 3:32–39, 2005.

[20] Bettye Rose Connell, Mike Jones, Ron Mace, Jim Mueller, Abir Mullick, Elaine Ostroff, Jon Sanford, Ed Steinfeld, Molly Story, and Gregg Vanderheiden. The center for universal design (1997). the principles of universal design, version 2.0. raleigh. http://www.design.ncsu.edu/cud/.

[21] Øystein Dale. GPS technology a viable orientation aid for people with dual sensory impairment? 6th DbI Conference on Deafblindness, 2005.

[22] Andrew Downie. A consumers view of electronic navigational technology, 2000. http://www.wayfinding.net/consumerview.htm.

[23] Torbjørn Halvorsen and Harald K. Jansson. Geographitti using mobile devices. Technical report, Østfold University College, 2005.

[24] Thomas Hermann, Jan M. Drees, and Helge Ritter. Broadcasting Auditory Weather Reports - A Pilot Project. International Conference on Auditory Display, Jul 2003.

[25] Simon Holland, David R. Morse, and Henrik Gedenryd. Audiogps: Spatial audio navigation with a minimal attention interface. *Personal and Ubiquitous Computing*, 6:253–259, 2002.

[26] Alan Kay. About the dynabook computer. http://www.artmuseum.net/w2vr/archives/Kay/01_Dynabook.html.

[27] L. Kay. Ultrasonic Mobility Aids for the Blind. *Proc. Rotterdam Mobility Research Conference*, 1956.

[28] L. Kay. The sonic glasses evaluated. *New outlook for the Blind*, 67:7–11, 1973.

[29] Albert Krohn, Michael Beigl, Mike Hazas, Hans-Werner Gellersen, and Albrecht Scmidt. Using Fine-Grained Infrared Positioning to Support the Surface-Based Activities of Mobile Users. 5th International Workshop on Smart Appliances and Wearable Computing (IWSAWC) 2005, 2005.

[30] Paul Marks. Microsoft develops shape-shifting touchscreen. *New Scientist*, November 2010.

[31] Peter Parente and Gary Bishop. Bats: The blind audio tactile mapping system. Technical report, University of North Carolina at Chapel Hill, Department of Computer Science, 2003.

[32] Jean-Baptiste Prost, Baptiste Godefroy, and Stephane Terrenoir. City walk: improving GPS Accuracy for Urban Pedestrians. *GPS World*, August 2008.

[33] Ivan Edward Sutherland. Sketchpad, a man-machine graphical communication system, January 1963.

[34] Bruce N. Walker and Jeffrey Lindsay. Navigation Performance With a Virtual Auditory Display: Effects of Beacon Sound, Capture Radius, and Practice. *Human Factors*, 48(2):265–278, 2006.

[35] Mark Weiser. Ubiquous computing. *IEEE Computer*, October 1993.

[36] Mark Weiser and John Seely Brown. Designing calm technology, 1995.

[37] Vasileios Zeimpekis, George M. Giaglis, and George Lekakos. A Taxonomy of Indoor and Outdoor Positioning Techniques for Mobile Location Services, 2003.

List of Figures

List of Tables

Appendix A

UbiMap UML Diagrams

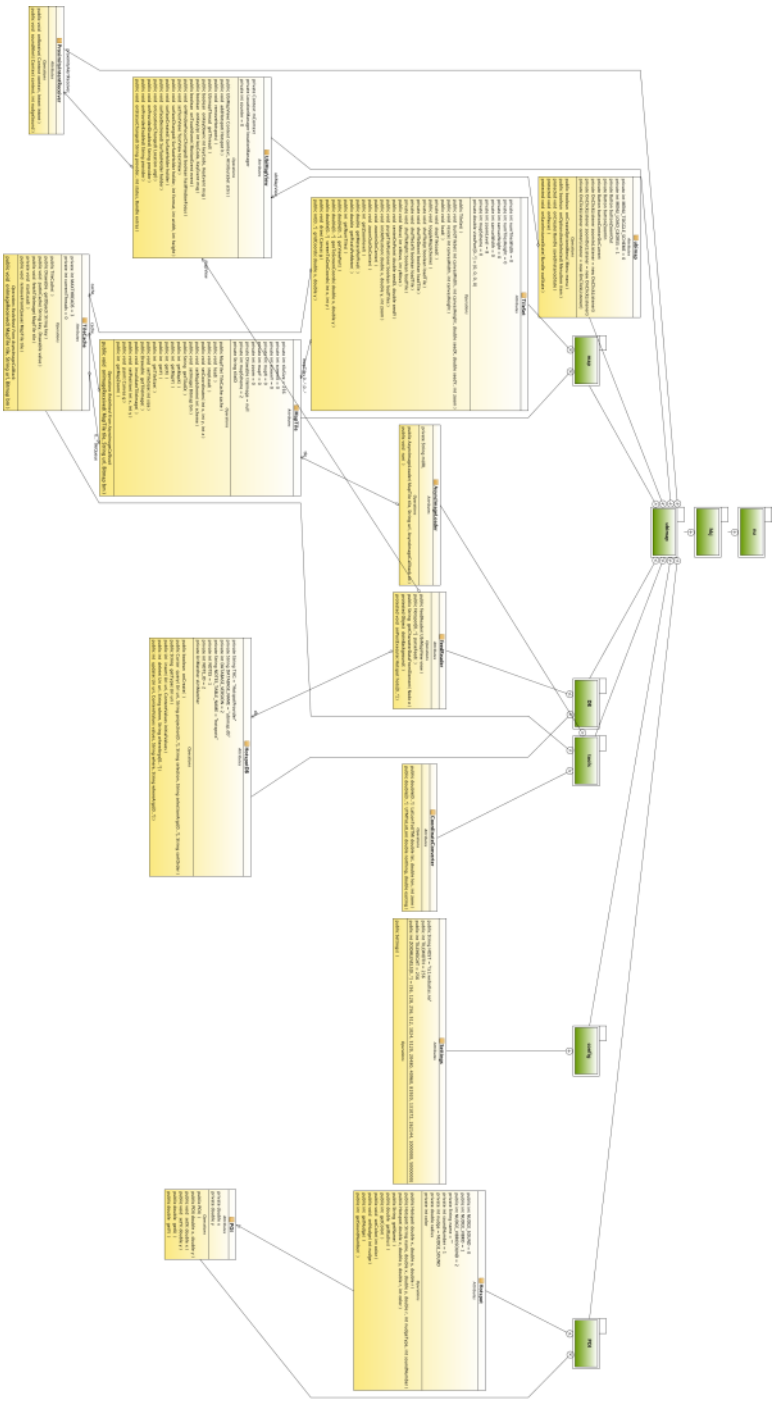


Figure A.1: Client Architecture

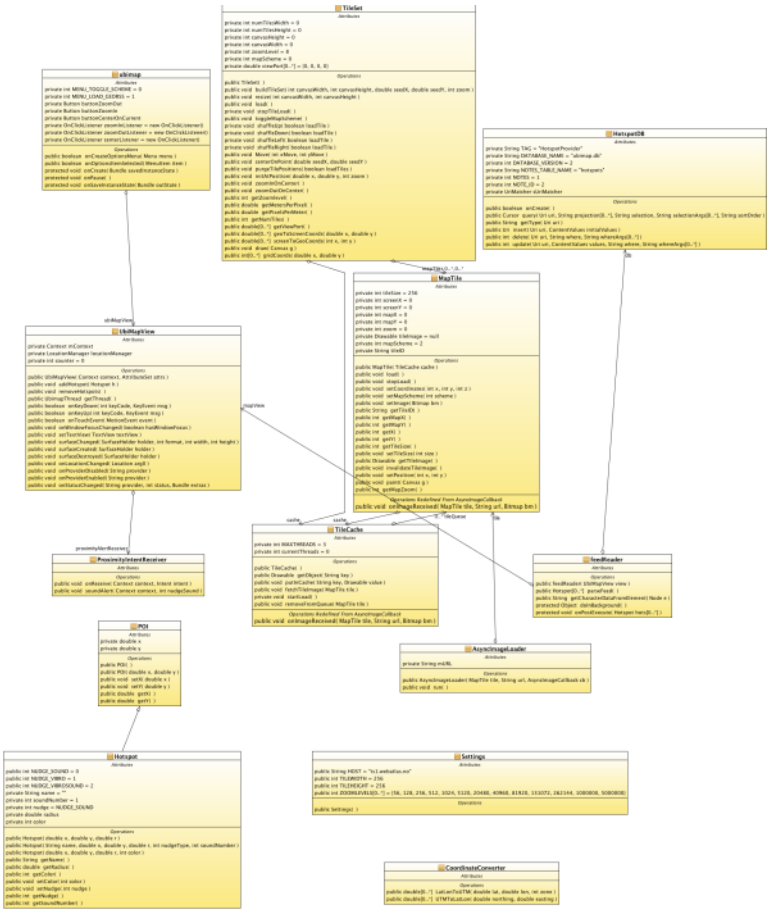


Figure A.2: Client Architecture

Table of Contents

Appendix B

Package nu.hkj.ubimap.map

Package Contents *Page*

Classes

B.1 Classes

B.1.1 CLASS **MapTile**

Represents a single map tile. Multiple map tiles are used to make up a complete map.

DECLARATION

public class MapTile
extends java.lang.Object
implements nu.hkj.ubimap.tools.AsyncImageLoader.AsyncImageCallback

CONSTRUCTORS

- *MapTile*
  `public` **MapTile**`( nu.hkj.ubimap.tools.TileCache` **cache** `)`
  - **Usage**
    - Creates a new instance of MapTile

METHODS

- *getMapX*
  `public int` **getMapX**`( )`
  - **Usage**
    - Gets the tile easting.
  - **Returns** - Tile easting.

- *getMapY*
  `public int` **getMapY**`( )`
  - **Usage**

    * Gets the tile northing.
  - **Returns** - Tile northing.

- *getMapZoom*
  `public int` **getMapZoom**`( )`
  - **Usage**
    * Gets the tile zoomlevel.
  - **Returns** - Tile zoomlevel.

- *getTileID*
  `public String` **getTileID**`( )`
  - **Usage**
    * Gets the tile id.
  - **Returns** - MapTile id.

- *getTileImage*
  `public Drawable` **getTileImage**`( )`
  - **Usage**
    * Gets the tile image.
  - **Returns** - Tile image.

- *getTileSize*
  `public int` **getTileSize**`( )`
  - **Usage**
    * Gets the tile size.
  - **Returns** - Tile size.

- *getX*
  `public int` **getX**`( )`
  - **Usage**

    * Gets tile screen x.
  - **Returns** - Tile screen x-position.

- *getY*
  `public int` **getY**`( )`
  - **Usage**
    * Gets tile screen y.
  - **Returns** - Tile screen y-position.

- *invalidateTileImage*
  `public void` **invalidateTileImage**`( )`
  - **Usage**
    * Invalidates and removes the tile image.

- *load*
  `public void` **load**`( )`
  - **Usage**
    * Tells the MapTile to load its image.

- *onImageReceived*
  `public void` **onImageReceived**`( nu.hkj.ubimap.map.MapTile` **tile,** `java.lang.String` **url,** `Bitmap` **bm** `)`
  - **Usage**
    * Callback method for receiving an image.

- *paint*
  `public void` **paint**`( Canvas` **g** `)`
  - **Usage**
    * Paint the tile.
  - **Parameters**
    * `g` - The canvas to use when painting.

- *setCoordinates*

  `public void` **setCoordinates**`( int` **x,** `int` **y,** `int` **z** `)`

  - **Usage**
    - Sets the MapTile coordinates.
  - **Parameters**
    - `x` - Easting.
    - `y` - Northing.
    - `z` - Zoomlevel.

- *setImage*

  `public void` **setImage**`( Bitmap` **bm** `)`

  - **Usage**
    - Sets the tile image.
  - **Parameters**
    - `bm` - Bitmap to use.

- *setMapScheme*

  `public void` **setMapScheme**`( int` **scheme** `)`

  - **Usage**
    - Sets the tile mapstyle.
  - **Parameters**
    - `scheme` -

- *setPosition*

  `public void` **setPosition**`( int` **x,** `int` **y** `)`

  - **Usage**
    - Sets the tile position.
  - **Parameters**
    - `x` - Screen x-position.
    - `y` - Screen y-position.

- *setTileSize*
  `public void` **setTileSize**`( int` **size** `)`
  - **Usage**
    - Sets the tile size.
  - **Parameters**
    - `size` - Tile size.

- *stopLoad*
  `public void` **stopLoad**`( )`
  - **Usage**
    - Tells the MapTile to stop any loading.

B.1.2 CLASS **TileSet**

Represents a set of tiles used to view a map image.

DECLARATION

public class TileSet **extends** java.lang.Object

CONSTRUCTORS

- *TileSet*
  `public` **TileSet**`( )`
  - **Usage**
    - Creates a new instance of TileSet

METHODS

- *buildTileSet*
  `public void` **buildTileSet**`(` `int` **canvasWidth**`,` `int` **canvasHeight**`,` `double` **seedX**`,` `double` **seedY**`,` `int` **zoom** `)`
  - **Usage**
    - builds a new tileset, given screen dimensions and start coordinates
  - **Parameters**
    - `canvas` - Width Map canvas width
    - `canvas` - Height Map canvas height
    - `seedX` - Start x coordinate
    - `seedY` - Start y coordinate

- *centerOnPoint*
  `public void` **centerOnPoint**`(` `double` **seedX**`,` `double` **seedY** `)`
  - **Usage**
    - Centers on the given coordinates
  - **Parameters**
    - `seedX` - Easting
    - `seedY` - Northing

- *draw*
  `public void` **draw**`(` `Canvas` **g** `)`
  - **Usage**
    - Draws the tileset using a Canvas.
  - **Parameters**
    - `g` - Canvas used for painting.

- *geoToScreenCoords*
  `public double` **geoToScreenCoords**`(` `double` **x**`,` `double` **y** `)`
  - **Usage**

* Takes UTM coordinates as input and gives screen coordinates back.

- *getMetersPerPixel*
  `public double` **getMetersPerPixel**`( )`
  - **Usage**
    * Gets current resolution in meters per pixel
  - **Returns** -

- *getNumTiles*
  `public int` **getNumTiles**`( )`
  - **Usage**
    * Gets the total number of tiles in the tileset
  - **Returns** -

- *getPixelsPerMeter*
  `public double` **getPixelsPerMeter**`( )`
  - **Usage**
    * Gets the current resolution in pixels per meter
  - **Returns** -

- *getViewPort*
  `public double` **getViewPort**`( )`
  - **Usage**
    * Gets the current viewport (the boundingbox coordinates of the current view)
  - **Returns** -

- *getZoomlevel*
  `public int` **getZoomlevel**`( )`
  - **Usage**
    * Gets the current zoomlevel

  - **Returns** -

- *gridCoords*
  `public int` **gridCoords**`( double` **x**`, double` **y** `)`
  - **Usage**
    - Conform coordinates to grid.
  - **Parameters**
    - `x` - The X-coordinate to transform.
    - `y` - The Y-coordinate to transform.

- *initAtPosition*
  `public void` **initAtPosition**`( double` **x**`, double` **y**`, int` **zoom** `)`
  - **Usage**
    - Inits the tileset at a given position
  - **Parameters**
    - `x` - East coordinate
    - `y` - North Coordinate
    - `zoom` - Zoomlevel

- *load*
  `public void` **load**`( )`
  - **Usage**
    - Fetches images for every tile in the tileset.

- *Move*
  `public void` **Move**`( int` **xMove**`, int` **yMove** `)`
  - **Usage**
    - Moves the tileset in XY direction.
  - **Parameters**
    - `xMove` - Number of pixels to move in x direction (positive or negative values).
    - `yMove` - Number of pixels to move in y direction (positive or negative values).

- *purgeTilePositions*
  `public void` **purgeTilePositions**`( boolean` **loadTiles** `)`
  - **Usage**
    - Purges the tile positions to see if they should be shuffled in any direction
  - **Parameters**
    - `loadTiles` - Set to true if tiles should be loaded immediately after a re-shuffle

---

- *resize*
  `public void` **resize**`( int` **canvasWidth**`, int` **canvasHeight** `)`
  - **Usage**
    - Resizes the tile canvas, and build a new tileset.
  - **Parameters**
    - `canvasWidth` -
    - `canvasHeight` -

- *screenToGeoCoords*
  `public double` **screenToGeoCoords**`( int` **x**`, int` **y** `)`
  - **Usage**
    - Takes screen coordinates as input and gives UTM coordinates back.

- *toggleMapScheme*
  `public void` **toggleMapScheme**`( )`
  - **Usage**
    - Toggles the current mapscheme between vector and satellite images.

- *zoomInOnCenter*
  `public void` **zoomInOnCenter**`( )`
  - **Usage**
    - Zoom the tileset one level in, using the center coordinate.

- *zoomOutOnCenter*
  `public void` **zoomOutOnCenter**`( )`
  - **Usage**
    - Zooms the tileset one level out, using the center coordinate.

Appendix C

Package nu.hkj.ubimap.tools

Package Contents *Page*

Interfaces

Classes

C.1 Interfaces

C.1.1 INTERFACE AsyncImageLoader.AsyncImageCallback

Interface for AsyncImageCallback

DECLARATION

public static interface AsyncImageLoader.AsyncImageCallback

METHODS

- *onImageReceived*
  `public void` **onImageReceived** `( nu.hkj.ubimap.map.MapTile` **tile,** `java.lang.String` **url,** `Bitmap` **bm** `)`
  - **Parameters**
    - `url` - Same URL as the one passed to the AsyncImageLoader constructor.
    - `bm` - A Bitmap object, or null on failure.

C.2 Classes

C.2.1 CLASS AsyncImageLoader

used to retrieve map images from the tileserver.

DECLARATION

public class AsyncImageLoader **extends** java.lang.Thread

CONSTRUCTORS

- *AsyncImageLoader*
  `public` **AsyncImageLoader**`( nu.hkj.ubimap.map.MapTile` **tile,**
  `java.lang.String` **url,**
  `nu.hkj.ubimap.tools.AsyncImageLoader.AsyncImageCallback` **cb** `)`
  - **Usage**
    - Start an asynchronous image fetch operation.
  - **Parameters**
    - `url` - The URL of the remote picture.
    - `cb` - The AsyncImageCallback object you want to be notified the operation completes.

METHODS

- *run*
  `public void` **run**`( )`
  - **Usage**
    - Runs the thread.

METHODS INHERITED FROM CLASS `java.lang.Thread`

- *activeCount*
  `public static int` **activeCount**`( )`
- *checkAccess*
  `public final void` **checkAccess**`( )`
- *countStackFrames*
  `public native int` **countStackFrames**`( )`
- *currentThread*
  `public static native Thread` **currentThread**`( )`

- *destroy*
  `public void` **destroy** `( )`
- *dumpStack*
  `public static void` **dumpStack** `( )`
- *enumerate*
  `public static int` **enumerate** `( java.lang.Thread [] arg0 )`
- *getAllStackTraces*
  `public static Map` **getAllStackTraces** `( )`
- *getContextClassLoader*
  `public ClassLoader` **getContextClassLoader** `( )`
- *getDefaultUncaughtExceptionHandler*
  `public static Thread.UncaughtExceptionHandler`
  getDefaultUncaughtExceptionHandler `( )`
- *getId*
  `public long` **getId** `( )`
- *getName*
  `public final String` **getName** `( )`
- *getPriority*
  `public final int` **getPriority** `( )`
- *getStackTrace*
  `public StackTraceElement` **getStackTrace** `( )`
- *getState*
  `public Thread.State` **getState** `( )`
- *getThreadGroup*
  `public final ThreadGroup` **getThreadGroup** `( )`
- *getUncaughtExceptionHandler*
  `public Thread.UncaughtExceptionHandler` **getUncaughtExceptionHandler** `( )`
- *holdsLock*
  `public static native boolean` **holdsLock** `( java.lang.Object` **arg0** `)`
- *interrupt*
  `public void` **interrupt** `( )`

- *interrupted*
  `public static boolean` **interrupted** `( )`
- *isAlive*
  `public final native boolean` **isAlive** `( )`
- *isDaemon*
  `public final boolean` **isDaemon** `( )`
- *isInterrupted*
  `public boolean` **isInterrupted** `( )`
- *join*
  `public final void` **join** `( )`
- *join*
  `public final synchronized void` **join** `( long` **arg0** `)`
- *join*
  `public final synchronized void` **join** `( long` **arg0,** `int` **arg1** `)`
- *resume*
  `public final void` **resume** `( )`
- *run*
  `public void` **run** `( )`
- *setContextClassLoader*
  `public void` **setContextClassLoader** `( java.lang.ClassLoader` **arg0** `)`
- *setDaemon*
  `public final void` **setDaemon** `( boolean` **arg0** `)`
- *setDefaultUncaughtExceptionHandler*
  `public static void` **setDefaultUncaughtExceptionHandler** `(`
  `java.lang.Thread.UncaughtExceptionHandler` **arg0** `)`
- *setName*
  `public final void` **setName** `( java.lang.String` **arg0** `)`
- *setPriority*
  `public final void` **setPriority** `( int` **arg0** `)`
- *setUncaughtExceptionHandler*
  `public void` **setUncaughtExceptionHandler** `(`
  `java.lang.Thread.UncaughtExceptionHandler` **arg0** `)`

- *sleep*
  `public static native void` **sleep** `( long` **arg0** `)`
- *sleep*
  `public static void` **sleep** `( long` **arg0**`, int` **arg1** `)`
- *start*
  `public synchronized void` **start** `( )`
- *stop*
  `public final void` **stop** `( )`
- *stop*
  `public final synchronized void` **stop** `( java.lang.Throwable` **arg0** `)`
- *suspend*
  `public final void` **suspend** `( )`
- *toString*
  `public String` **toString** `( )`
- *yield*
  `public static native void` **yield** `( )`

C.2.2 CLASS **CoordinateConverter**

Used for converting coordinates between decimal degrees and UTM coordinates.

DECLARATION

public class CoordinateConverter **extends** java.lang.Object

CONSTRUCTORS

- *CoordinateConverter*
  `public` **CoordinateConverter** `( )`

METHODS

- *LatLonToUTM*
  `public static double` **LatLonToUTM**`( double` **lat**`, double` **lon**`, int` **zone** `)`
  - **Usage**
    - Converts a coordinate from lat/lon to north/east in a given UTM zone.
  - **Parameters**
    - `lat` - The latitude to convert.
    - `lon` - The longitude to convert.
    - `zone` - The UTM zone to convert to.
  - **Returns** - Array with lat/lon coordinates.

- *UTMToLatLon*
  `public static double` **UTMToLatLon**`( double` **northing**`, double` **easting** `)`
  - **Usage**
    - Convert a coordinate from UTM to decimal degrees.
  - **Parameters**
    - `northing` - Northing of the coordinate to convert.
    - `easting` - Easting of the coordinate to convert.
  - **Returns** - Array with east/north coordinates.

C.2.3 CLASS **TileCache**

Represents a tile cache, responsible for loading and caching tile images.

DECLARATION

public class TileCache
extends java.lang.Object
implements AsyncImageLoader.AsyncImageCallback

CONSTRUCTORS

- *TileCache*
  `public` **TileCache**`( )`
  - **Usage**
    - Creates a new instance of TileCache

METHODS

- *fetchTileImage*
  `public void` **fetchTileImage**`( nu.hkj.ubimap.map.MapTile` **tile** `)`
  - **Usage**
    - Fetch imagedata for a tile.
  - **Parameters**
    - `tile` - MapTile.

- *getObject*
  `public Drawable` **getObject**`( java.lang.String` **key** `)`
  - **Usage**
    - Retrieve an image from the memory-cache.
  - **Parameters**
    - `key` -
  - **Returns** - Object from cache.

- *onImageReceived*
  `public void` **onImageReceived**`( nu.hkj.ubimap.map.MapTile` **tile**`,`
  `java.lang.String` **url**`, Bitmap` **bm** `)`
  - **Usage**

    * Callback from fetcThread

- *putInCache*
  `public void` **putInCache**`( java.lang.String` **key,** `Drawable` **value** `)`
  - **Usage**
    * Put a tile image in the memory-cache
  - **Parameters**
    * `key` - Id to use when caching.
    * `value` - Image to cache.

- *removeFromQueue*
  `public void` **removeFromQueue**`( nu.hkj.ubimap.map.MapTile` **tile** `)`
  - **Usage**
    * Removes a tile from the load queue.
  - **Parameters**
    * `tile` - Tile to remove.

Appendix D

Package nu.hkj.ubimap

Package Contents *Page*

Classes

D.1 Classes

D.1.1 CLASS **ProximityIntentReceiver**

Listens for proximity events and triggers nudges.

DECLARATION

public class ProximityIntentReceiver **extends** BroadcastReceiver

CONSTRUCTORS

- *ProximityIntentReceiver*
  `public` **ProximityIntentReceiver**`( )`

METHODS

- *onReceive*
  `public void` **onReceive**`( Context` **context,** `Intent` **intent** `)`
- *soundAlert*
  `public static void` **soundAlert**`( Context` **context,** `int` **nudgeSound** `)`

  - **Usage**
    - Sounds an alert.
  - **Parameters**
    - `context` - Context to use when playing sounds.
    - `nudgeSound` - Nudgesound to use.

METHODS INHERITED FROM CLASS `BroadcastReceiver`

D.1.2 CLASS **ubimap**

Main class for UbiMap. UbiMap is a simple location aware application. It has a zoomable/pannable map view, and can load hotspots from a GEORSS-source, which in turn triggers actions when the user enters one.

DECLARATION

public class ubimap **extends** Activity

CONSTRUCTORS

- *ubimap*
  `public` **ubimap**`( )`

METHODS

- *onCreate*
  `protected void` **onCreate**`( Bundle` **savedInstanceState** `)`
  - **Usage**
    - Invoked when the Activity is created.
  - **Parameters**
    - `savedInstanceState` - a Bundle containing state saved from a previous execution, or null if this is a new execution

- *onCreateOptionsMenu*
  `public boolean` **onCreateOptionsMenu**`( Menu` **menu** `)`
  - **Usage**
    - Invoked during init to give the Activity a chance to set up its Menu.
  - **Parameters**

    * `menu` - The Menu to which entries may be added.
  - **Returns** - true

- *onOptionsItemSelected*
  `public boolean` **onOptionsItemSelected** `( MenuItem` **item** `)`
  - **Usage**
    * Invoked when the user selects an item from the Menu.
  - **Parameters**
    * `item` - the Menu entry which was selected
  - **Returns** - true if the Menu item was legit (and we consumed it), false otherwise

---

- *onPause*
  `protected void` **onPause** `( )`
  - **Usage**
    * Invoked when the Activity loses user focus.

- *onSaveInstanceState*
  `protected void` **onSaveInstanceState** `( Bundle` **outState** `)`
  - **Usage**
    * Notification that something is about to happen, to give the Activity a chance to save state.
  - **Parameters**
    * `outState` - a Bundle into which this Activity should save its state

METHODS INHERITED FROM CLASS `Activity`

D.1.3 CLASS **UbiMapView**

View that represents a map. This view is can be used to pan and zoom the map, as well as viewing hotspots.

DECLARATION

public class UbiMapView
extends SurfaceView

CONSTRUCTORS

- *UbiMapView*
  `public` **UbiMapView**`( Context` **context,** `AttributeSet` **attrs** `)`
  - **Usage**
    - Constructs a new UbiMapView.
  - **Parameters**
    - `context` - Context to use.
    - `attrs` - Attributes to use.

METHODS

- *addHotspot*
  `public void` **addHotspot**`( nu.hkj.ubimap.POI.Hotspot` **h** `)`
  - **Usage**
    - Add a hotspot to the view.
  - **Parameters**
    - `h` - Hotspot to add.

- *getThread*
  `public UbiMapView.UbimapThread` **getThread**`( )`
  - **Usage**
    - Returns the animation thread corresponding to this view.
  - **Returns** - The animation thread.

- *onKeyDown*
  `public boolean` **onKeyDown**`( int` **keyCode**`, KeyEvent` **msg** `)`
  - **Usage**
    * Standard override to get key-press events.

- *onKeyUp*
  `public boolean` **onKeyUp**`( int` **keyCode**`, KeyEvent` **msg** `)`
  - **Usage**
    * Standard override for key-up.

- *onLocationChanged*
  `public void` **onLocationChanged**`( Location` **arg0** `)`
  - **Usage**
    * Callback invoked when location changes.

- *onProviderDisabled*
  `public void` **onProviderDisabled**`( java.lang.String` **provider** `)`
  - **Usage**
    * Callback invoked when a locationprovider is disabled.

- *onProviderEnabled*
  `public void` **onProviderEnabled**`( java.lang.String` **provider** `)`
  - **Usage**
    * Callback invoked when a locationprovider is enabled.

- *onStatusChanged*
  `public void` **onStatusChanged**`( java.lang.String` **provider**`, int` **status**`, Bundle` **extras** `)`
  - **Usage**

    * Callback invoked when status changes.

- *onTouchEvent*
  `public boolean` **onTouchEvent**`( MotionEvent` **event** `)`
  - **Usage**
    * Standard override for touch events.

- *onWindowFocusChanged*
  `public void` **onWindowFocusChanged**`( boolean` **hasWindowFocus** `)`
  - **Usage**
    * Standard window-focus override. Notice focus lost so we can pause on focus lost. e.g. user switches to take a call.

- *removeHotspots*
  `public void` **removeHotspots**`( )`
  - **Usage**
    * removes all hotspots from the view.

- *setTextView*
  `public void` **setTextView**`( TextView` **textView** `)`
  - **Usage**
    * Installs a pointer to the text view used for messages.

- *surfaceChanged*
  `public void` **surfaceChanged**`( SurfaceHolder` **holder,** `int` **format,** `int` **width,** `int` **height** `)`
  - **Usage**
    * Callback invoked when the surface dimensions change.

- *surfaceCreated*
  `public void` **surfaceCreated**`( SurfaceHolder` **holder** `)`

- **Usage**
  - Callback invoked when the Surface has been created and is ready to be used.

- *surfaceDestroyed*
  `public void` **surfaceDestroyed** `( SurfaceHolder` **holder** `)`
  - **Usage**
    - Callback invoked when the Surface has been destroyed and must no longer be touched.

METHODS INHERITED FROM CLASS `SurfaceView`

D.1.4 CLASS UbiMapView.UbimapThread

Map thread used for updating the map view.

DECLARATION

public class UbiMapView.UbimapThread **extends** java.lang.Thread

CONSTRUCTORS

- *UbiMapView.UbimapThread*
  `public` **UbiMapView.UbimapThread** `( SurfaceHolder` **surfaceHolder,** `Context` **context** `)`
  - **Usage**
    - Constructs a new UbimapThread.
  - **Parameters**
    - `surfaceHolder` - Surface holder.

* `context` - Context to use.
* `handler` -

METHODS

- *addHotspot*
  `public void` **addHotspot**`( nu.hkj.ubimap.POI.Hotspot` **h** `)`
  - **Usage**
    * Adds a hotspot to the map.
  - **Parameters**
    * `h` - Hotspot to add.

- *centerOnCurrentPosition*
  `public void` **centerOnCurrentPosition**`( )`
  - **Usage**
    * Center the map on the current location. This may be either obtained from cell or wifi-positioning, or (if available) from an on-board GPS unit.

- *doKeyDown*
  `public boolean` **doKeyDown**`( int` **keyCode**`, KeyEvent` **msg** `)`
  - **Usage**
    * Handles a key-down event.
  - **Parameters**
    * `keyCode` - the key that was pressed.
    * `msg` - the original event object.
  - **Returns** - true

- *doKeyUp*
  `public boolean` **doKeyUp**`( int` **keyCode**`, KeyEvent` **msg** `)`
  - **Usage**
    * Handles a key-up event.

  - **Parameters**
    - `keyCode` - the key that was pressed.
    - `msg` - the original event object.
  - **Returns** - true if the key was handled and consumed, or else false.

- *doTouchEvent*
  `public boolean` **doTouchEvent** `( MotionEvent` **event** `)`
  - **Usage**
    - Handles a touch event.
  - **Parameters**
    - `event` -
  - **Returns** -

- *getHandsetPOI*
  `public POI` **getHandsetPOI** `( )`
  - **Usage**
    - Returns the POI representation of the handset. The handset POI holds the last known position of the handset.
  - **Returns** -

- *gotoPostition*
  `public void` **gotoPostition** `( double` **x,** `double` **y,** `int` **zoom** `)`
  - **Usage**
    - Tells the viewthread to initiate at a given position and xoomlevel.
  - **Parameters**
    - `x` - East coordinate.
    - `y` - North coordinate.
    - `zoom` - Zoomlevel to use.

- *pause*
  `public void` **pause** `( )`

  - **Usage**
    - Pauses the view thread.

- *removeHotspots*
  `public void` **removeHotspots**`( )`
  - **Usage**
    - Removes all hotspots from the map.

- *run*
  `public void` **run**`( )`
  - **Usage**
    - Starts the view thread.

- *saveState*
  `public Bundle` **saveState**`( Bundle` **map** `)`
  - **Usage**
    - Saves the map state.
  - **Returns** - Bundle with this view's state.

- *setRunning*
  `public void` **setRunning**`( boolean` **b** `)`
  - **Usage**
    - Signals the view thread, and tells it whether it should run or shut down.
  - **Parameters**
    - `b` - true to run, false to shut down.

- *setStyle*
  `public void` **setStyle**`( int` **style** `)`
  - **Usage**
    - Sets the map style.

  - **Parameters**
    - `style` - May be either 0=vector, or 1=ortho.

- *setSurfaceSize*
  `public void` **setSurfaceSize**`( int` **width**`, int` **height** `)`
  - **Usage**
    - Callback invoked when the surface dimensions change.

- *zoomIn*
  `public void` **zoomIn**`( )`
  - **Usage**
    - Zoom the map one level in.

- *zoomOut*
  `public void` **zoomOut**`( )`
  - **Usage**
    - Zoom the map one level out.

METHODS INHERITED FROM CLASS `java.lang.Thread`

- *activeCount*
  `public static int` **activeCount**`( )`

- *checkAccess*
  `public final void` **checkAccess**`( )`

- *countStackFrames*
  `public native int` **countStackFrames**`( )`

- *currentThread*
  `public static native Thread` **currentThread**`( )`

- *destroy*
  `public void` **destroy**`( )`

- *dumpStack*
  `public static void` **dumpStack** `( )`
- *enumerate*
  `public static int` **enumerate** `( java.lang.Thread [] arg0 )`
- *getAllStackTraces*
  `public static Map` **getAllStackTraces** `( )`
- *getContextClassLoader*
  `public ClassLoader` **getContextClassLoader** `( )`
- *getDefaultUncaughtExceptionHandler*
  `public static Thread.UncaughtExceptionHandler`
  getDefaultUncaughtExceptionHandler `( )`
- *getId*
  `public long` **getId** `( )`
- *getName*
  `public final String` **getName** `( )`
- *getPriority*
  `public final int` **getPriority** `( )`
- *getStackTrace*
  `public StackTraceElement` **getStackTrace** `( )`
- *getState*
  `public Thread.State` **getState** `( )`
- *getThreadGroup*
  `public final ThreadGroup` **getThreadGroup** `( )`
- *getUncaughtExceptionHandler*
  `public Thread.UncaughtExceptionHandler` **getUncaughtExceptionHandler** `( )`
- *holdsLock*
  `public static native boolean` **holdsLock** `( java.lang.Object arg0 )`
- *interrupt*
  `public void` **interrupt** `( )`
- *interrupted*
  `public static boolean` **interrupted** `( )`

- *isAlive*
  `public final native boolean` **isAlive** `( )`
- *isDaemon*
  `public final boolean` **isDaemon** `( )`
- *isInterrupted*
  `public boolean` **isInterrupted** `( )`
- *join*
  `public final void` **join** `( )`
- *join*
  `public final synchronized void` **join** `( long` **arg0** `)`
- *join*
  `public final synchronized void` **join** `( long` **arg0**`, int` **arg1** `)`
- *resume*
  `public final void` **resume** `( )`
- *run*
  `public void` **run** `( )`
- *setContextClassLoader*
  `public void` **setContextClassLoader** `( java.lang.ClassLoader` **arg0** `)`
- *setDaemon*
  `public final void` **setDaemon** `( boolean` **arg0** `)`
- *setDefaultUncaughtExceptionHandler*
  `public static void` **setDefaultUncaughtExceptionHandler** `(`
  `java.lang.Thread.UncaughtExceptionHandler` **arg0** `)`
- *setName*
  `public final void` **setName** `( java.lang.String` **arg0** `)`
- *setPriority*
  `public final void` **setPriority** `( int` **arg0** `)`
- *setUncaughtExceptionHandler*
  `public void` **setUncaughtExceptionHandler** `(`
  `java.lang.Thread.UncaughtExceptionHandler` **arg0** `)`
- *sleep*
  `public static native void` **sleep** `( long` **arg0** `)`

- *sleep*
  `public static void` **sleep** `( long` **arg0**`, int` **arg1** `)`
- *start*
  `public synchronized void` **start** `( )`
- *stop*
  `public final void` **stop** `( )`
- *stop*
  `public final synchronized void` **stop** `( java.lang.Throwable` **arg0** `)`
- *suspend*
  `public final void` **suspend** `( )`
- *toString*
  `public String` **toString** `( )`
- *yield*
  `public static native void` **yield** `( )`

Appendix E

Package nu.hkj.ubimap.DB

Package Contents *Page*

Classes

E.1 Classes

E.1.1 CLASS feedReader

Reads a RSS-based feed, and creates new Hotspots.

DECLARATION

public class feedReader **extends**

CONSTRUCTORS

- *feedReader*
  `public` **feedReader** `( nu.hkj.ubimap.UbiMapView` **view** `)`

METHODS

- *doInBackground*
  `protected Object` **doInBackground** `( java.lang.Object [] arg0 )`
  - **Usage**
    - Thread method for adding hotspots.

- *getCharacterDataFromElement*
  `public static String` **getCharacterDataFromElement** `( org.w3c.dom.Node` **e** `)`
  - **Usage**
    - Retrieve character data from a node.
  - **Parameters**
    - `e` - Node to read from.

  - **Returns** - Node contents.

- *onPostExecute*
  `protected void` **onPostExecute** `( nu.hkj.ubimap.POI.Hotspot [] hots )`

- *parseFeed*
  `public Hotspot` **parseFeed** `( )`

  - **Usage**
    - Parses an ubimap RSS-feed and returns the hotspots.
  - **Returns** - Prsed hotspots.

METHODS INHERITED FROM CLASS

E.1.2 CLASS **HotspotDB**

Used for storing map data in a database.

DECLARATION

---

public class HotspotDB **extends** ContentProvider

CONSTRUCTORS

- *HotspotDB*
  `public` **HotspotDB** `( )`

METHODS

- *delete*
  `public int` **delete**`( Uri` **uri**`, java.lang.String` **where**`, java.lang.String [] whereArgs )`
  - **Usage**
    - Deletes from the database.

- *getType*
  `public String` **getType**`( Uri` **uri**`)`
  - **Usage**
    - Returns type of uri.

- *insert*
  `public Uri` **insert**`( Uri` **uri**`, ContentValues` **initialValues**`)`
  - **Usage**
    - Inserts into the database.

- *onCreate*
  `public boolean` **onCreate**`( )`
  - **Usage**
    - Callback invoked when a new database is created.

- *query*
  `public Cursor` **query**`( Uri` **uri**`, java.lang.String [] projection, java.lang.String` **selection**`, java.lang.String [] selectionArgs, java.lang.String` **sortOrder**`)`
  - **Usage**
    - Returns a Cursor from the database.

- *update*
  `public int` **update**`( Uri` **uri**`, ContentValues` **values**`,`
  `java.lang.String` **where**`, java.lang.String [] ` **whereArgs** `)`
  - **Usage**
    - Update a field in the database.

METHODS INHERITED FROM CLASS `ContentProvider`

Appendix F

Package nu.hkj.ubimap.POI

Package Contents *Page*

Classes

F.1 Classes

F.1.1 CLASS **Hotspot**

Represents a geographical hotspot. Can be used to associate nudges to specific locations, or areas.

DECLARATION

public class Hotspot **extends** nu.hkj.ubimap.POI.POI

FIELDS

- public static final int NUDGE_SOUND
  - Sound nudge
- public static final int NUDGE_VIBRO
  - Vibrate nudge
- public static final int NUDGE_VIBROSOUND
  - Vibrate and sound nudge

CONSTRUCTORS

- *Hotspot*
  `public` **Hotspot**`( double` **x,** `double` **y,** `double` **r** `)`
  - **Usage**
    - Creates a new instance of Hotspot.
  - **Parameters**
    - `x` - Easting.
    - `y` - Northing.

    * `r` - Radius of the hotspot.

- *Hotspot*
  `public` **Hotspot**`( double` **x**`, double` **y**`, double` **r**`, int` **color** `)`
  - **Usage**
    * Creates a new instance of Hotspot.
  - **Parameters**
    * `x` - Easting.
    * `y` - Northing.
    * `r` - Radius.
    * `color` - Color to be used when drawing the hotspot.

- *Hotspot*
  `public` **Hotspot**`( java.lang.String` **name**`, double` **x**`, double` **y**`, double` **r**`, int` **nudgeType**`, int` **soundNumber** `)`
  - **Usage**
    * Creates a new instance of Hotspot.
  - **Parameters**
    * `name` - Hotspot name.
    * `x` - Easting.
    * `y` - Northing.
    * `r` - Radius.
    * `nudgeType` - Type of nudge to be used.
    * `soundNumber` - Sound number if if sound nudge is used.

METHODS

- *getColor*
  `public int` **getColor**`( )`
  - **Usage**
    * Gets the hotspot color.
  - **Returns** - The hotspot color.

- *getName*
  `public String` **getName**`( )`
  - **Usage**
    - Gets for hotspot name.
  - **Returns** - Hotspot name.

- *getNudge*
  `public int` **getNudge**`( )`
  - **Usage**
    - Gets the hotspot nudge type.
  - **Returns** - Hotspot nudge type.

- *getRadius*
  `public double` **getRadius**`( )`
  - **Usage**
    - Gets the hotspot radius.
  - **Returns** - Hotspot radius.

- *getSoundNumber*
  `public int` **getSoundNumber**`( )`
  - **Usage**
    - Gets the hotspot sound number.
  - **Returns** - Hotspot sound number.

- *setColor*
  `public void` **setColor**`( int` **color** `)`
  - **Usage**
    - Sets the hotspot color.
  - **Parameters**

* `color` - The hotspot color.

- *setNudge*
  `public void` **setNudge**`( int` **nudge** `)`
  - **Usage**
    * Sets the hotspot nudge type.
  - **Parameters**
    * `nudge` - The nudge type to set.

METHODS INHERITED FROM CLASS `nu.hkj.ubimap.POI.POI`

(in F.1.2, page 142)

- *getX*
  `public double` **getX**`( )`
  - **Usage**
    * Gets the POI easting.
  - **Returns** - Easting.

- *getY*
  `public double` **getY**`( )`
  - **Usage**
    * Gets the POI Northing.
  - **Returns** - Northing.

- *setX*
  `public void` **setX**`( double` **x** `)`
  - **Usage**
    * Sets the POI Easting.
  - **Parameters**
    * `x` - Easting.

- *setY*
  `public void` **setY**`( double` **y** `)`
  - **Usage**

* Sets the POI Northing.

– **Parameters**

* `y` - Northing.

F.1.2 CLASS **POI**

Represents a single Point Of Interest.

DECLARATION

public class POI **extends** java.lang.Object

CONSTRUCTORS

- *POI*
  `public` **POI**`( )`

  – **Usage**

  * Creates a new instance of POI.

- *POI*
  `public` **POI**`( double` **x,** `double` **y** `)`

  – **Usage**

  * Creates a new instance of POI.

  – **Parameters**

  * `x` - Easting.
  * `y` - Northing.

METHODS

- *getX*
  `public double` **getX**`( )`
  - **Usage**
    - Gets the POI easting.
  - **Returns** - Easting.

- *getY*
  `public double` **getY**`( )`
  - **Usage**
    - Gets the POI Northing.
  - **Returns** - Northing.

- *setX*
  `public void` **setX**`( double` **x** `)`
  - **Usage**
    - Sets the POI Easting.
  - **Parameters**
    - `x` - Easting.

- *setY*
  `public void` **setY**`( double` **y** `)`
  - **Usage**
    - Sets the POI Northing.
  - **Parameters**
    - `y` - Northing.

Appendix G

Package nu.hkj.ubimap.config

Package Contents *Page*

Classes

G.1 Classes

G.1.1 CLASS **Settings**

Setting for the Ubimap application.

DECLARATION

public class Settings **extends** java.lang.Object

FIELDS

- public static final String HOST
  - Host to use for tilefetching.
- public static final int TILEWIDTH
  - Tile width in pixels.
- public static final int TILEHEIGHT
  - Tile height in pixels.
- public static final int ZOOMLEVELS
  - Zoomlevels defines in meters per pixel.

CONSTRUCTORS

- *Settings*
  `public` **Settings**`( )`
  - **Usage**
    - Creates a new instance of Settings

Appendix H

Translated User Questionaires

H.1 Sprreskjemaer til brukertesten

H.1.1 Pre-test skjema

Before the tests are executed, every participant fills out the pre-test questionaire. This has questions regarding how the participant uses sound in everyday navigation.

1. Har du et synshandicap?

a. Ja.

b. Nei.

2. Om du har et synshandicap; i hvilken grad er synet nedsatt?

a. Blind.

b. Delvis blind.

c. Grad av syn i prosent:

3. Hvilken erfaringsgrad har du med mobiltelefoner?

a. Erfaren/ekspertbruker.

b. Daglig bruk, men bruker ikke avanserte funksjoner.

c. Nybegynner/enkel bruk.

4. Har mobiltelefonen din støtte for navigasjon?

a. Ja.

b. Nei.

5. Bruker du mobiltelefonen din til navigasjon?

a. Ofte (3-4 ganger i uken).

b. Noen ganger (1-2 ganger i måneden).

c. Sjelden (1-2 ganger i halvåret).

d. Aldri.

6. Hvilken del av navigasjonsprogrammet bruker du mest?

a. Et visuelt kart.

b. Navigasjon ved hjelp av talesyntese.

c. Taktil respons (vibrasjon).

d. Ingen av delene.

7. Bruker du andre hjelpemidler for å navigere?

a. Blindestokk.

b. Frerhund.

c. Andre - oppgi hjelpemiddel.

d. Ingen av delene.

8. Hvordan vil du beskrive ditt daglige navigasjonsmiljø?

a. Stille, med lite trafikk.

b. Moderat støy, med noe trafikk.

c. Mye støy, med forskjellig trafikk (biler, busser, trikk osv.).

9. Hvor viktig er synes ditt når du beveger deg utendørs?

a. Det er min viktigste sans.

b. Jeg bruker det som et supplement til andre sanser.

c. Jeg er ikke avhengig av synet mitt når jeg beveger meg utendørs.

10. Hvor viktig er hørselen din når du beveger deg utendørs?

a. Det er min viktigste sans.

b. Jeg bruker den som et supplement til andre sanser.

c. Jeg er ikke avhengig av hørselen min når jeg beveger meg utendrs.

11. Hvor viktig er berringssansen din når du beveger deg utendørs?

a. Det er min viktigste sans.

b. Jeg bruker den som et supplement til andre sanser.

c. Jeg er ikke avhengig av berringssansen min når jeg beveger meg utendrs.

12. Hender det at du hører du på musikk eller bruker hodetelefoner når du beveger deg utendørs?

a. Ja.

b. Nei.

H.1.2 Post-test Skjema

1. I hvilket type miljø brukte du prototypeapplikasjonen?

a. Stille, med lite trafikk.

b. Moderat støy, med noe trafikk.

c. Mye støy, med forskjellig trafikk (biler, busser, trikk osv.).

2. Synes du det var lett å skille lydene fra applikasjonen fra annen bakgrunnsstøy?

a. Ja.

b. Nei.

3. Synes du at den genererte lyden var forstyrrende?

a. Ja.

b. Nei.

4. Ville du foretrukket tale, fremfor lydsignaler?

a. Ja.

b. Nei.

5. Brukte du de vibrotaktile signalene fra applikasjonen?

a. Ja.

b. Nei.

6. Synes du de vibrotaktile signalene var forstyrrende?

a. Ja.

b. Nei.

7. Brukte du det visuelle kartet?

a. Ja.

b. Nei.

8. Ville du foretrukket et visuelt kart?

a. Ja.

b. Nei.

9. Kunne du ha brukt en lignende applikasjon som daglig navigasjonshjelpemiddel?

a. Ja.

b. Nei.

c. Kanskje.

10. På hvilken måte synes du at appliaksjonen kunne vært forbedret? Beskriv gjerne hvordan du kunne tenke deg å forandre den.

Beskrivelse: