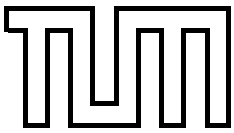
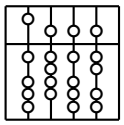


Technische Universität München
Fakultät für Informatik

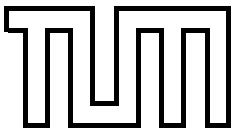


Diplomarbeit

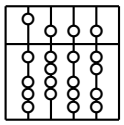
Design eines für Blinde und Sehbehinderte geeigneten Navigationssystems mit taktiler Ausgabe

NAVI
Navigation **A**id for **V**isually **I**mpaired

Günther Harrasser



Technische Universität München
Fakultät für Informatik



Diplomarbeit

Design eines für Blinde und Sehbehinderte geeigneten Navigationssystems mit taktiler Ausgabe

NAVI
Navigation **A**id for **V**isually **I**mpaired

Günther Harrasser

Aufgabensteller:	Prof. Gudrun Klinker, Ph.D.
Betreuer:	Dipl.–Inf. Martin Bauer Dipl.–Inf. Martin Wagner
Abgabedatum:	15. Dezember 2003

Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Dezember 2003 Günther Harrasser

Inhaltsverzeichnis

1 Einführung

Laut dem DBSV [1] gibt es in Deutschland 155.000 blinde Menschen und rund 500.000 Menschen mit Sehbehinderung. Hilfsmittel wie der weiße Langstock und der Blindenhund haben es ermöglicht, dass sie sich in ihnen bekannter Umgebung frei bewegen können. Diese Autonomie endet aber, wenn ihr Weg sie in unbekanntes Gebiet führt. Ohne fremde Hilfe können Sehbehinderte ihr gewünschtes Ziel nur mit großer Anstrengung erreichen. Wenn kein hilfreiches Auge ihnen Informationen über die Umgebung gibt, wird das Suchen einer Adresse zum Spießrutenlauf.

Mit dem Projekt NAVI haben wir, Florian Reitmeir und ich, versucht, eine sprechende Straßenkarte zu entwickeln, die dem blinden Benutzer auf dem Weg zu seinem Ziel eine Hilfe darstellen soll. Mit diesem System soll er bei seiner Navigation nicht mehr auf fremde Hilfe angewiesen sein und selbständig sein Ziel erreichen.

Diese Diplomarbeit beschäftigt sich hauptsächlich mit dem Navigationssystem und dessen Ein- und Ausgaben an den Benutzer. Damit ein Navigationssystem von blinden Menschen bedient werden kann, müssen die Eingaben an das System so gestaltet sein, dass diese auch ohne visuelle Rückmeldung funktionieren. Dadurch, dass Teile der Ausgabe durch taktile [2] Vibrationen ersetzt werden, kann in bestimmten Situationen, wo bei existierenden Systemen bisher akustische Ausgabe benutzt wurde, das Gehör entlastet und weiterhin auf die Wahrnehmung der Umgebung gerichtet werden. Bei der Ausgabe von Positionsinformationen wird versucht, durch gezielte und an Blinde angepasste Formulierungen der Informationen ein optimales Verständnis zu erreichen.

Die gewonnenen Erkenntnisse und daraus getroffenen Entscheidungen wurden in Form eines Prototypen implementiert.

Die zweite Diplomarbeit im Rahmen des NAVI–Projektes beschäftigt sich mit der Filterung der vom System ausgegebenen Umgebungsdaten, damit nur für den Benutzer interessante Informationen mitgeteilt werden. Mehr dazu ist in [44] nachzulesen.

1.1 Aufbau des Dokumentes

Ich werde im folgenden versuchen, die Bereiche Navigationssystem für Blinde, Eingabe und Ausgabe zu erarbeiten. Im **zweiten Kapitel** gehe ich deshalb auf die Problemstellung an sich ein, um die Anforderungen des Gesamtsystems zu analysieren.

In anderen Universitäten und mehreren Firmen wurden bereits Systeme für das Thema "Navigation für Blinde" entwickelt und erarbeitet, worauf ich in **Kapitel drei** eingehen werde.

[1] Deutscher Blinden- und Sehbehindertenverband e.V.
[2] fühlbar

In **Kapitel vier** werden einige Entscheidungen für Problemlösungen, die im Verlauf des Projektes getroffen wurden, erleutert und erklärt.

Kapitel fünf und sechs behandeln die Ein- und Ausgabe, auf welche bei der Entwicklung eines für Blinde zugänglichen Systems hauptsächlich geachtet werden muss.

Das **Kapitel sieben** beschreibt den im Rahmen dieser Diplomarbeit entwickelten Prototypen. Teile dieses Kapitels, die das Gesamtsystem NAVI beschreiben, wurden in Zusammenarbeit mit Florian Reitmeir [44] geschrieben.

Im letzten Kapitel, **Kapitel acht**, führe ich die erreichten Ziele und einen Ausblick in die Zukunft auf, was noch alles möglich ist bzw. was es noch zu tun gibt und welche Weiterentwicklungen in den nächsten Jahren zu erwarten sind.

2 Problemstellung, Anforderungen

Will man ein "Navigationssystem für Blinde" konstruieren, so gibt es zwei große Themengebiete, die man erarbeiten muss. Zum ersten muss ein funktionierendes **Navigationssystem** vorhanden sein und zum zweiten muss dieses System **für Blinde** benutzbar gemacht werden.

Die Kombination aus diesen beiden weitreichenden Gebieten, stellt eine Problemstellung für sich dar. Ausgereifte Navigationssysteme gibt es für viele Anwendungsgebiete, wobei die Benutzung dieser vor allem auf sehende Personen ausgerichtet ist. Ebenso gibt es viele Systeme für Blinde, die aber auf die Nutzung von Computern bzw. Handhelds ausgelegt sind. Die Zusammenführung dieser Bereiche fordert vor allem die Anpassung des Navigationssystems an die Bedürfnisse blinder Personen.

In unterschiedlichsten Studien und Arbeiten wurde versucht, die Anforderungen an ein Navigationssystem für Blinde zu sammeln. Bei einer Studie in England wurden insgesamt 24 Personen für das MOBIC Projekt [46] befragt. Unter diesen waren betroffene Männer und Frauen im Alter zw. 26 und 75 Jahren, die sich im täglichen Leben mit Hilfe des weißen Stocks bzw. eines Blindenhundes fortbewegten. Außerdem wurden Mobilitätstrainer und Personen, die mit diesem Gebiet etwas zu tun hatten, in die Studie miteinbezogen.

Im Rahmen eines Systementwicklungsprojektes an der TU München wurden von mir, zusammen mit einem sehbehinderten Mitstudenten, Rahmenbedingungen für ein Navigationssystem für Blinde [32] erarbeitet. Außerdem hatte ich im Rahmen eines anderen Projektes die Möglichkeit mit mehreren blinden Personen über die Thematik zu sprechen und ihre Meinungen bzw. Vorschläge einzuholen.

Aufgrund dieser unterschiedlichen Quellen ergaben sich folgende Anforderungen:

Berechnung der Route zum gewünschten Ziel: Das Navigationssystem soll dem Benutzer helfen an sein gewünschtes Ziel zu gelangen. Dies beinhaltet als ersten Schritt die Auswahl desselben. Der Weg zu diesem Ziel wird dann vom System berechnet. Bei Blinden Personen ist es dabei wichtiger, dass der Weg zum Ziel wenig Hindernisse und Gefahren beinhaltet, als dass der Weg der kürzeste aller möglichen ist. In die Berechnung sollen bei Bedarf auch öffentliche Verkehrsmittel wie Bus oder U–Bahn miteinbezogen werden können.

Lokalisierung des Benutzers: Um eine Navigation zu ermöglichen, muss das System erkennen, wo sich der Benutzer gerade befindet. Dabei sollen Faktoren wie Wetter, Zeit und Ort die Navigation nicht beeinträchtigen.

Information über die aktuellen Position: Der Benutzer muss bei Bedarf über seine aktuelle Position informiert werden. Dabei sollen Informationen darüber, wo er sich gerade befindet, ausgegeben werden.

Informationen über die Umgebung: Während ein Blinder eine Straße entlang geht, nimmt er seine Umgebung vor allem akustisch war. Er kann nicht sehen was um ihn herum ist. Deshalb muss das Navigationssystem ihm mitteilen, an welchen Objekten er gerade vorbei geht. Dabei können Geschäfte, Imbisse, Restaurants usw. für ihn interessant sein. Bei Banken und damit verbunden Geldautomaten möchte er vielleicht Geld abheben.

Signalisierung von Objekten: Bei gewissen, vorher definierten Objekten der Umgebung wie Briefkästen oder Geldautomaten soll der Benutzer auf sie hingewiesen werden.

Überprüfung der Funktionalität von Ein- und Ausgabekomponenten Beim Starten des Systems muss die Funktionalität aller Komponenten sichergestellt werden, damit eventuelle Fehler wie Wackelkontakte oder fehlende Verbindungen erkannt werden.

Status des Navigationssystems: Es muss zu jedem Zeitpunkt klargestellt werden, ob das System gerade verwendbar ist oder nicht. Dabei ist es vor allem wichtig den aktuellen Status der Positionsbestimmung erfahren zu können. Funktioniert diese nicht vollständig, kann sich der Benutzer nicht auf das Navigationssystem verlassen.

Hinweis auf Gefahren: Auf dem Weg zum Zielpunkt können sich Gefahrenstellen befinden. Zu diesen gehört auch eine Kreuzung. An diesen Stellen muss dem Benutzer eine Warnung ausgegeben werden, damit dieser vorsichtiger weitergeht. Der Weg wird zwar durch den weißen Stock abgetastet bzw. der Blindenhund erkennt gewisse Hindernisse, doch es gibt Stellen wo der Blinde mit dem Stock das Hindernis nicht erkennt bzw. der Blindenhund dieses nicht als solches wahrnimmt.

Weiterer Verlauf bei Kreuzungen: Kreuzungen sind wichtige Entscheidungspunkte für die Navigation. Der Benutzer muss eventuell abbiegen, um zum Ziel zu gelangen. Treffen an einer Kreuzung mehrere Straßen zusammen, so ist es schwierig zu erkennen, auf welcher Straße nun weitergegangen werden soll.

Minimale Belastung des Gehörs: Die Umgebung außerhalb der Reichweite des weißen Stocks wird ausschließlich durch das Gehör wahrgenommen. Die Aufmerksamkeit von Blinden ist sehr stark auf die umgebende Geräuschkulisse gerichtet. Deshalb soll das Gehör so minimal wie möglich durch das Navigationssystem in Anspruch genommen werden.

Feedback ohne visuelle Ausgabe: Bei Eingaben an das System braucht der Benutzer ein Feedback auf seine Aktionen. Er muss wissen, ob die Taste, die er soeben gedrückt hat, auch die gewünschte Aktion ausgelöst hat. Ändert sich der Status des Systems, muss er wissen was gerade passiert ist bzw. welche Eingabe von ihm erwartet wird. Aufgrund des degenierten bzw. fehlenden Sehsinns kann keine visuelle Ausgabe erkannt werden. Deshalb muss eine Alternative dafür verfügbar sein.

Ausgabe von Daten: Daten die vom System an den Benutzer ausgegeben werden kann dieser nicht von einem Display ablesen. Deshalb muss eine andere Form der Ausgabe gewählt werden.

Benutzbarkeit: Das System soll so einfach wie möglich benutzbar sein. Komplexe Aktionen müssen so modelliert werden, dass der Benutzer nicht überfordert ist. Die Bedienung

muss einfach und leicht verständlich sein, damit auch technisch nicht versierte Personen das System bedienen können.

Ergonomische Handhabung: Die Eingabe erfolgt entweder im Ruhezustand oder während der Navigation. Vor allem komplexere Eingabesequenzen werden nicht während des Gehens gemacht, weshalb dabei beide Hände benutzt werden können. Während der Navigation wird eine Hand meist für den weißen Stock bzw. den Blindenhund benötigt weshalb erforderliche Eingaben nur mit einer Hand getätigt werden können.

Planung der Navigation: Es soll möglich sein, die Navigation am heimischen PC zu planen und Informationen zu hinterlegen. Die Tagesrouten sollen erstellt werden können um eine individuelle Planung zu ermöglichen.

Größe des Systems: Das System soll so unauffällig wie möglich sein. Außerdem darf das Gewicht und die Größe des Systems keine Behinderung während der Navigation darstellen.

Hardwarekomponenten: Um den Preis der Hardware minimal zu halten, sollen soweit wie möglich Serienprodukte verwendet werden. Kostspielige Sonderentwicklungen sollen wenn möglich vermieden werden.

Das NAVI–System wurde auf diese Probleme hin entwickelt und für den Großteil davon wurde ein Lösungsvorschlag erarbeitet und implementiert.

Ein Navigationssystemen für Blinde soll NUR als zusätzliche Hilfe zu schon bestehenden Blindenhilfsmitteln wie dem weißen Stock oder dem Blindenführhund dienen. Eventuelle Ungenauigkeiten während der Positionsbestimmung müssen vom System miteinberechnet werden. Damit die Umgebung in die Navigation miteingebracht werden kann, benötigt es sehr genaue Karten, in denen Geschäfte, Restaurants usw. aber auch Details wie Hindernisse für blinde Personen, Parkbänke, Aufzüge, Treppen, Unterführungen, Gehsteige usw. abgespeichert sind.

2.1 Szenario

Folgendes Szenario stellt einen möglichen Ablauf für eine Navigation dar.

Stevie sitzt daheim vor dem Rechner und sucht sich für den nächsten Tag eine Straßenkarte für das Gebiet aus, in dem er sich bewegen wird. Er braucht für morgen noch etwas Geld von der Bank, weshalb er dies vermerkt. Zum Schluss überträgt er die Daten auf das mobile Navigationssystem.

Am nächsten Tag schaltet er das Navigationssystem ein, welches sofort mit der Positionsbestimmung startet. Wurde der aktuelle Aufenthaltsort gefunden, so meldet das System dies. Auch alle anderen Komponenten melden ihre Funktionalität.

Nun muss das gewünschte Ziel dem Navigationssystem mitgeteilt werden, worauf dieses die optimale Route dorthin berechnet und Informationen über den zurückzulegenden Weg ausgibt. Anschließend geht Stevie los, wobei ihm das Navigationssystem die empfohlene Startrichtung mitteilt.

Auf seinem Weg kommt er an einer Bank vorbei und bekommt dies vom System auch mitgeteilt. Er kann sich nun Geld abheben und geht dann weiter.

Wenn er an eine Kreuzung kommt, gibt das Navigationssystem einen Warnhinweis über die taktile [1] Ausgabe aus, wodurch Stevie weiß, dass er vorsichtiger sein muss. Wenn an der Kreuzung abgebogen werden soll, dann wird dies ebenfalls über die taktile Ausgabe mitgeteilt.

Um die aktuelle Position zu erfahren, kann Stevie den Namen der aktuellen Straße bzw. Orientierungsinformationen jederzeit ausgeben lassen. Außerdem bekommt er während des Gehens Informationen über umliegende Geschäfte, Imbisse usw. Er kann diese bewerten, ob sie ihn interessieren oder nicht, wodurch das Navigationssystem ein Benutzerprofil von ihm erstellt. Im besten Fall gibt das System dadurch nur noch solche Daten aus, die ihn auch wirklich interessieren. Alles Uninteressante wird nicht mehr erwähnt.

Biegt Stevie in eine falsche Straße einbiegen, so wird die Route über diesen Umweg neu berechnet. Würde die Straße dabei nicht ans Ziel führen, so würde das Navigationssystem ihm mitteilen, dass er umdrehen muss.

Ist Stevie am Ziel angekommen, wird ihm dies signalisiert und das System kann bis zur nächsten Verwendung abgeschalten werden.

[1] fühlbar

3 Existierende Navigationssysteme

Die menschliche Navigation besteht aus zwei äußerst unterschiedlichen Funktionen: Zum einen müssen lokal gegebene Hindernisse umgangen, zum anderen muss zu einem entfernten Ziel hingesteuert werden, welches außerhalb der aktuellen wahrnehmbaren Umgebung liegt.

Navigation beinhaltet die abwechselnde Aktualisierung der momentanen Position und die Orientierung auf ein gewünschtes Ziel hin, meistens entlang einer festgelegten Route.

GPS ist die Abkürzung für "Global Positioning System". Das System besteht aus 24 nichtstationären Satelliten, die vom US–Militär betrieben werden [1]. Von jedem Punkt der Erde sind jederzeit mindestens 12 Satelliten am Himmel sichtbar und funken ständig ihre aktuelle exakte Position sowie die genaue Uhrzeit zum Boden. Hat der GPS–Empfänger freie Sicht zu den Satelliten, so können diese Signale ausgewertet und dadurch die mehr oder weniger genaue Position bestimmt werden. Zur Berechnung des Standortes werde mindestens drei Satelliten benötigt, bei vier ist zusätzlich die Berechnung der Höhe möglich.

Seitdem die Empfänger für GPS–Signale kleiner und billiger geworden sind, ist der Markt für Navigationssysteme rapide gewachsen. GPS–Navigationssysteme werden vor allem in Autos, Booten und Flugzeugen eingebaut. Wegen der eventuell schlechten Genauigkeit [2] von GPS wird das System eher seltener für Fußgänger verwendet. Vor allem in der Nähe von hohen Gebäuden und unter dicht belaubten Bäumen werden die Signale verfälscht und können teilweise überhaupt nicht mehr empfangen werden. Bei Autos kann man durch Sensoren an den Rädern Entfernungen und Richtungsänderungen erkennen und dadurch Ungenauigkeiten korrigieren, was bei Fußgängern leider nicht möglich ist. In Zukunft soll das Europäische Projekt GALILEO [6] das GPS–System ergänzen bzw. sogar ersetzen.

3.1 Navigationssysteme Allgemein

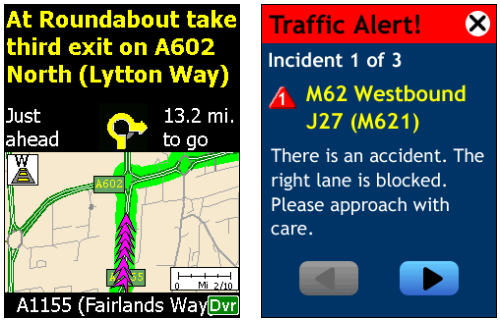
Die meisten Handhelds der jetzigen Generation können entweder durch ein Zusatzmodul erweitert werden oder haben schon einen Empfänger für GPS integriert. Es gibt viele konkurrierende Systeme, die sich teilweise nur in Details unterscheiden.

Die meisten Navigationssysteme für PDAs [3] sind vorwiegend für den Einsatz in Autos gedacht. Das Gerät wird in einen Clip, der an der Seitenkonsole im Auto angebracht ist, gesteckt und kann über einen eingebauten oder einen außerhalb angebrachten GPS–Empfänger die aktuelle Position empfangen.

[1] Ein anderer Begriff für das GPS System ist NAVSTAR

[2] GPS kann bis zu 15 Meter von der genauen Position abweichen [4].

[3] Personal Digital Assistant



(a) iPAQ Navigation System während der Fahrt

(b) LiveWire Traffic mit aktueller Verkehrsmeldung

Abbildung 3.1: PDA Navigationssystem für Autos mit Verkehrsmeldung

Die Software *iPAQ Navigation System* [13] ist nur ein Beispiel für so ein Navigationssystem. Auf dem PDA–Display wird auf einer Karte die aktuelle Position angezeigt. Falls der Fahrer abbiegen soll, so wird dies auf dem Bildschirm angezeigt (Abb. 3.1(a), S. 13) und über Sprachausgabe zusätzlich ausgegeben. Mit der Erweiterung *LiveWire Traffic* [15] können über GPRS aktuelle Verkehrsinformationen angezeigt werden (Abb. 3.1(b), S. 13).

Die Navigationssysteme für PDAs können zwar auch von Fußgängern benutzt werden, doch dies ist nur eine Zusatzfunktion.

3.2 Navigationssysteme für Blinde

Die erste Navigationshilfe war in den 40er Jahren der "weiße Stock" (Abb. 3.2, S. 13). Mit diesem 90 - 145 cm langen Stock wird während des Gehens der Bereich vor den Füßen abgetastet. Durch technologische Fortschritte wird der Stock immer mehr durch elektronische Hilfen, sog. ETAs [4], ersetzt bzw. mit Navigation und Hindernisüberwindung erweitert.

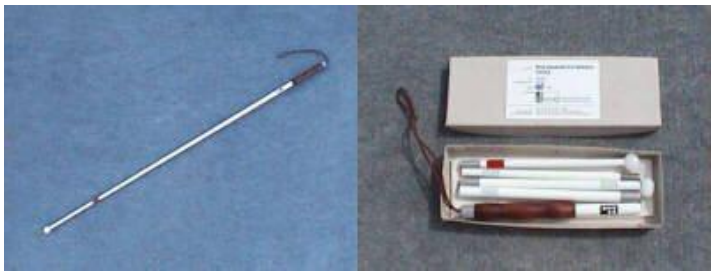


Abbildung 3.2: Weißer Stock der zusammengeklappt werden kann

[4] Electronic Travel Aid

Elektronische Geräte wie der Laser–Stock oder durch Ultraschall gesteuerte Hilfen warnen vor Hindernissen und helfen einen sicheren Ausweichweg zu finden. Durch diese Hilfsmittel haben ihre Benutzer einen gewissen Grad von autonomer Fortbewegung erlangt, doch effiziente Navigation basiert auf Informationen, die außerhalb der limitierten Reichweite dieser Geräte liegen.

In den letzten Jahrzehnten wurden ETAs vermehrt auf die Navigationsfähigkeit hinentwickelt. Ein Ansatz war die Ausstattung der Umgebung von mit Positionsmarken/Symbolen. Tastbare Markierungen sind nicht effektiv, da die Person erst durch Abtasten der Umgebung von deren Existenz erfährt. Stattdessen wurden Marken entwickelt, die mit entsprechenden Geräten aus einer gewissen Entfernung registriert werden können.

Ein anderer Ansatz ist die Nutzung von GPS, wobei die globalen Positionsinformationen für die Lokalisierung des Benutzers genutzt werden.

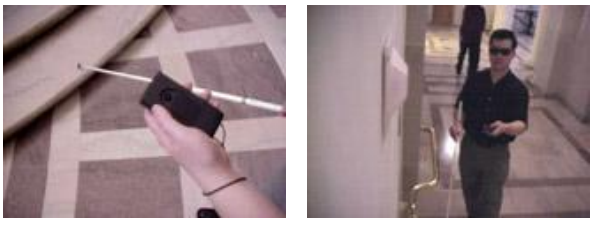
NAVI ist nicht der erste Versuch für ein Navigationssystem für Blinde. An unterschiedlichen Universitäten und in der Wirtschaft wurden mehrere Systeme entwickelt, die teilweise einen ähnlichen, teilweise einen völlig unterschiedlichen Lösungsweg beschreiten.

3.2.1 Talking Signs, Verbal Landmark

Dieses [25] in den USA entwickelte System besteht aus 2 Teilen:

1. Einem fest installierten **Sender**
2. Einem tragbaren **Empfänger**

Die Sender (Abb. 3.3(a), S. 14) sind an Türen, Schildern, Ampeln usw. angebracht und senden kontinuierlich durch unsichtbares Infrarot ein kurzes Audiosignal aus. Kommt der Benutzer mit dem Empfänger (Abb. 3.3(b), S. 14) in den Bereich des Senders (ca. 20 Meter), so werden die gesendeten Daten entweder über einen kleinen Lautsprecher oder über einen Kopfhörer ausgegeben. Das System funktioniert im Außen und Innenbereich und kann als Navigationshilfe und Umgebungsinformationssystem benutzt werden. Bei der Verwendung zeigt der Benutzer mit dem Empfänger in die Richtung, von der er das stärkste Signal bekommt und weiß dadurch, wo sich der Sender bzw. das beschriebene Objekt genau befindet. Steht er z.B. gerade in einer Hotelhalle, so hört er von der linken Seite “Zu den Aufzügen”,



(a) Sender (b) Empfänger

Abbildung 3.3: Talking Signs im Einsatz

von der rechten Seite “Rezeption” und von vorne “Speisesaal”. Die Nachrichten sind kurz und direkt gehalten und werden kontinuierlich wiederholt. Mittlerweile wird das System in Bahnhöfen, Hotels, Flughäfen, Krankenhäusern, Öffentlichen Einrichtungen, Geschäften uvm. genutzt. Bei einigen Bushaltestellen werden z.B. die Routen des Busses, vor dem sich der Benutzer gerade befindet, ausgegeben. Als weiteres Funktionsgebiet werden bei Ampeln die Rot–Grün Phasen ausgegeben.

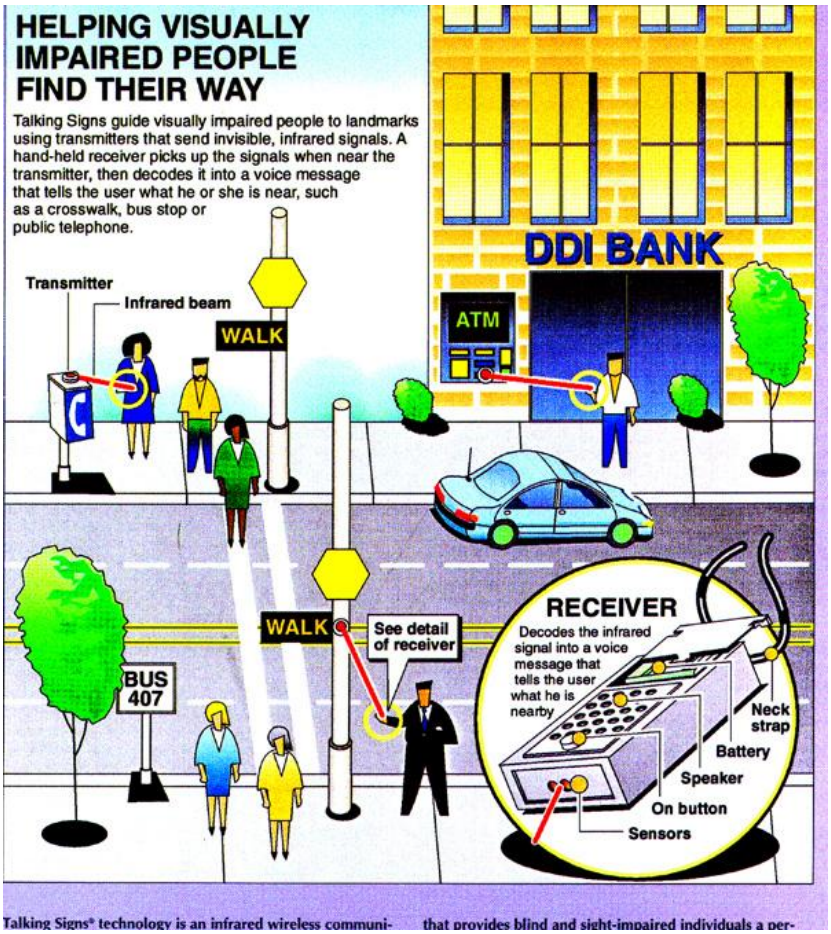


Abbildung 3.4: Talking Signs System

“Verbal Landmark” ist ein ähnliches System wie Talking Signs, mit dem Unterschied, dass die Signalreichweite kleiner ist und die Sender in alle Richtungen senden. Außerdem ist dieses System weniger verbreitet.

Kommentar: *Ein Nachteil dieser Systeme liegt vor allem darin, dass die Umgebung, in der Talking Signs bzw. Verbal Landmark funktionieren sollen, zuallererst mit den Sendern ausgestattet werden müssen. Sind diese angebracht, sprechen Faktoren wie Genauigkeit für das System. Die Navigation in nicht bebauten Gegenden stellt sich als schwieriger heraus. Zur Zeit sind einige größere Einrichtungen in den USA mit TS ausgestattet, in Norwegen, Finland, Italien, Japan und Schottland gibt es Installationen. Österreich und die Niederlande scheinen großes Interesse an dem System zu haben.*

3.2.2 Wearable Haptic Navigation Guidance System

Das "Wearable Haptic Navigation Guidance System"[34] wurde am Massachusetts Institute of Technology (MIT) in Cambridge entwickelt. In einer Weste sind auf dem Rücken 4×4 Vibrationsmotoren angebracht die taktile Navigation ermöglichen sollen. Durch Infrarotsignale wird der Standort des Benutzers ermittelt, ein tragbarer Computer übernimmt die Routenplanung. Das System ist für den Indoor–Bereich konstruiert und wurde in den Gebäuden der Universität getestet, wobei ein Plan der Räume auf dem PC, mit dem die Navigation geschieht, gespeichert sind.

Die Infrarot–Emitter senden eine eindeutige Kennung aus, wodurch der Empfänger die aktuelle Position berechnen kann. Die Anweisungen "vorwärts", "rückwärts", "links", "rechts" und "stop" leiten den Benutzer. Die Weste mit den Vibrationsmotoren ist in der Größe veränderbar. Auch die Position der Motoren kann verändert werden, damit das optimale taktile Ergebnis erreicht wird. Wird z.B. das Signal "rechts" ausgegeben, dann vibriert zuerst die rechteste Seite der Motoren drei mal, danach die zweitrechteste drei mal usw., "stop" wird durch eine rotierende Vibration ausgegeben. Die Signalreihenfolgen werden dabei mehrmals wiederholt.

Kommentar: *Dieses System ist das einzige mir bekannte, welches als Richtungsausgabe nicht akustische Informationen sondern Vibrationen benutzt. Wie die durchgeführten Studien zeigen, scheint dieser Ansatz vielversprechend zu sein. Das System wurde nur im Indoor–Bereich getestet, doch die Adaptierung auf den Outdoor–Einsatz scheint sich nur auf die Teile für die Positionsbestimmung bzw. Navigation zu beziehen. Ob die Methode mit den Motoren in der Weste die beste ist, oder ob die Anbringung der Vibrationsemitter an den Oberarmen, wie es von uns gedacht ist, besser ist, müsste durch weitere Untersuchungen geklärt werden. Wichtig ist auf jeden Fall, dass die Vibrationen auch bei Wind und anderen Bedingungen noch gut zu spüren sind.*

3.2.3 Drishti - An Integrated Navigation System for Visually Impaired and Disabled

Drishti[5] [38] wurde 2001 an der University of Florida entwickelt. Das System (Abb. 3.5, S. 17) benutzt GPS mit DGPS Korrektur für die Positionsbestimmung. Über wLAN wird auf eine GIS[6] Datenbank zugegriffen, welche aktuelle Informationen über das Gelände, Umgebungsinformationen und Navigationsdaten bereitstellt. Kommandos wie "Bring mich zu dem Gebäude XYZ" werden mittels Spracherkennung[7] eingegeben und Daten werden über eine Sprachsynthese ausgegeben. Für Personen mit einem Sehrest werden außerdem visuelle Informationen über einen kleinen, vor dem rechten Auge angebrachten Monitor dargestellt (Abb. 3.6, S. 18).

Über DGPS wird die aktuelle Position des Benutzers ermittelt und mit den Daten aus dem GIS–System werden Hindernisse auf dem geplanten Weg miteinberechnet. Hindernisse sind Baustellen, gerade stattfindende Umzüge, Staus und gefährliche Stellen wie z.B. abhängende Baumäste nach einem Gewitter.

[5] Drishti bedeutet in der antiken indischen Sprache Anskrit "sehen"
[6] Geografisches Informations System
[7] IBM Viavoice

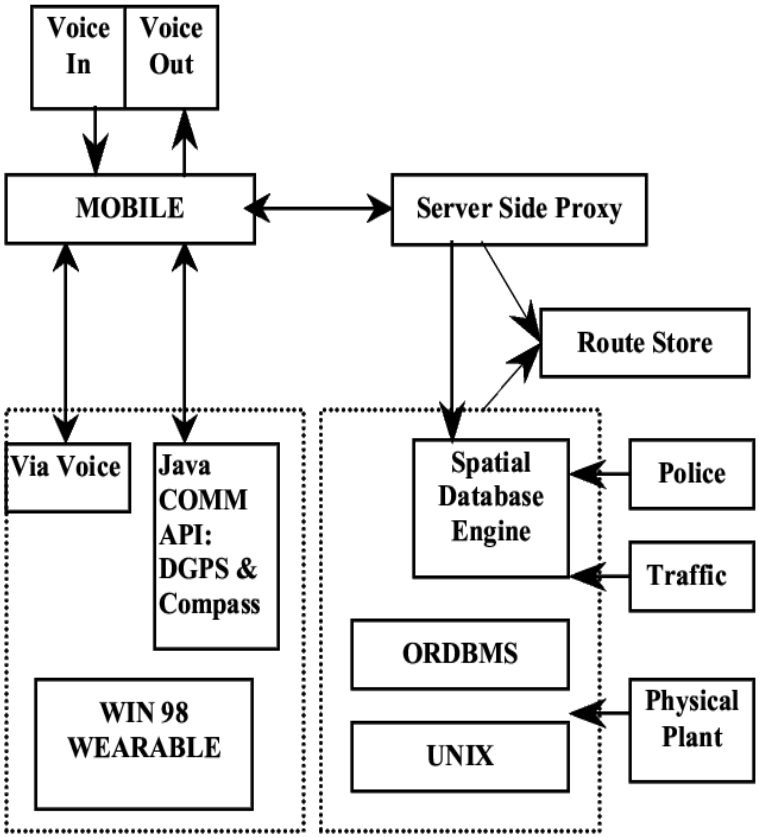


Abbildung 3.5: Drishti Systemarchitektur

Die Entwickler wollen mit Hilfe von Ultraschall Positionsmarken das System für den Indoor–Bereich erweitern. Außerdem ist die Portierung auf ein High–End Smartphone geplant.

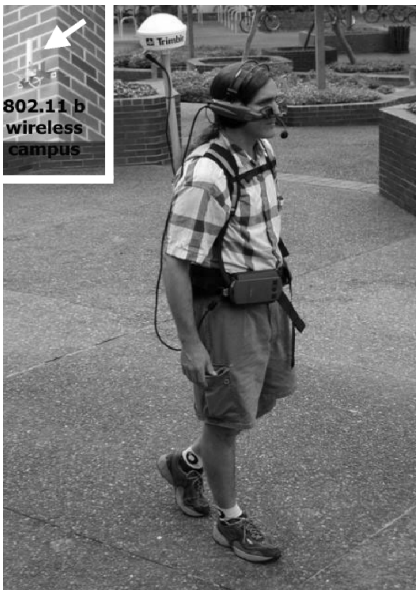


Abbildung 3.6: Drishti bei einer Vorführung

Kommentar: *Die in diesem System verwirklichten Ideen sind ein interessanter Ansatz zur Lösung des "Navigation für Blinde"–Problems. Vor allem die Einbindung der Geländeinformationen erscheint sinnvoll, da gerade bei blinden Personen der kürzeste Weg nicht immer der einfachste und deshalb der beste ist. Dabei kann man sicherlich noch mehr Informationen bei der Berechnung von Routen und Hindernissen einbringen als nur Umzüge und Baustellen. Die Bedienung des Systems ist nur durch Sprache möglich, was sich bei Listenauswahlen und sich häufig wiederholenden Kommandos als schwierig und nervend herausstellen dürfte. In diesem Bereich könnte das System durch andere Eingabegeräte noch sinnvoller genutzt werden. Durch die Verbindung zu einem zentralen Server wird die Aktualität der GIS–Daten sichergestellt. Das Navi–System gibt über taktile Ausgabe Richtungsinformationen und Warnhinweise aus um das Gehör des Benutzers zu entlasten. Bei Drishti wird dies dem Fußgänger durch Sprachausgabe mitgeteilt, was gerade bei vollblinden Personen einen Nachteil darstellt.*

3.2.4 Personal Guidance System

An der Universität von Kalifornien Santa Barbara und der Carnegie Mellon University wurde das sog. Personal Guidance System [21] entwickelt. Das Ziel war es, wie bei den anderen Projekten auch, ein System zu schaffen, das einer sehgeschädigten Person die Navigation und Orientierung in bekannter und aber auch unbekannter Umgebung ohne zusätzliche Hilfe ermöglicht. Das Grundkonzept beruht auf der Idee eines "virtual acoustic display". Dieses künstlich erzeugte Audiosignal vermittelt mit Hilfe von Kopfhörern dem Benutzer den Eindruck, dass die ihn umgebenden Gebäude mit ihm kommunizieren. Wenn er sich

fortbewegt, hört er z.B. die Namen der Gebäude aus der Richtung, wo das entsprechende Gebäude steht. Dadurch sollte auch ein Nicht–Sehender bei seiner "Reise" einen Eindruck von seiner Umgebung bekommen und sich besser orientieren können. Zur Vermeidung von Hindernissen werden weiterhin der Blindenstock und dessen Weiterentwicklungen eingesetzt.

Das System besteht aus mehreren Teilen (Abb. 3.7, S. 19):

- Zur Positionsbestimmung wird ein 12–Band DGPS Empfänger benutzt, der von einer 20km entfernten DGPS–Station das Korrektursignal erhält.
- Auf dem Kopfhörerbügel ist ein Kompass angebracht, mit dessen Hilfe man die Sehrichtung der Person ermitteln kann.
- In einer Tragevorrichtung befindet sich ein Notebook als Rechenzentrale. Auf diesem Rechner läuft neben der Systemsoftware und der Datenbank für die Umgebungsdaten zusätzlich noch ein GIS–System.
- Das Benutzerinterface besteht aus einem Mikrophon, durch welches man Befehle eingeben kann.

Abbildung 3.7: Der Prototyp des PGS

Kommentar: *Das PGS ist das älteste Navigationssystem für Blinde, das ich bei meiner Recherche gefunden habe. Es hat die Grundfunktionalitäten für ein GPS–basiertes Navigationssystem. In den Bereichen der Ein– und Ausgabe sind noch viele Möglichkeiten offen.*

Der interessanteste Teil dieses Systems ist das akustische Display. Es ist sehr schwierig jemandem, der nicht sehen kann, eine Richtung mitzuteilen weshalb das akustische Display sicherlich eine gute Lösung darstellt. Ich vermute, dass die Ausgabe nur für allgemeine Informationen über die Umgebung ausreicht, sozusagen als Übersicht. Die Genauigkeit ist vermutlich für gezielte Richtungen zu gering.

3.2.5 Visuaide - TREKKER

TREKKER [28] ist ein GPS–Navigationssystem der Kanadischen Firma Visuaide und wurde am 21. März 2003 eingeführt. Es besteht aus einem GPS–Empfänger und einem Compaq iPAQ 3950 (Abb. 3.8, S. 20). Der Benutzer kann die Straßenkarten käuflich erwerben und entweder per Download oder von CD auf den heimischen PC spielen. Von dort wird die Karte auf den PDA übertragen.

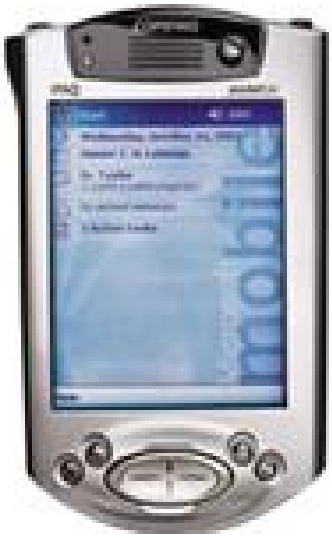


Abbildung 3.8: Der PDA von TREKKER

Einige der wichtigsten Funktionen von Trekker (Abb. 3.9, S. 21) sind:

- Auf den Straßenkarten können so genannte “Points Of Interest - P.O.I.” gesetzt werden, die während der Navigation angesteuert bzw. ausgegeben werden können. Außerdem können Routen geplant und gespeichert werden.
- Zu den P.O.I. können Sprachaufzeichnungen hinterlegt werden.
- Das Gerät besitzt eine Sprachausgabe.
- Im Navigationsmodus bekommt der Benutzer, wenn er weniger als 10 Meter von einer Kreuzung entfernt ist, eine Warnung dass er sich einem Fußgängerübergang nähert. Außerdem wird über eine Sprachausgabe die Navigationsrichtung ausgegeben.
- Der Benutzer kann sich zu jeder Zeit die Straße mit Hausnummer ausgeben lassen, in der er sich gerade befindet.

- Das Touchpad ist in mehrere Bereiche eingeteilt und kann als Brailletastatur oder Telefontastatur für die Eingabe verwendet werden.
- Die GPS–Status Informationen können ausgegeben werden.

Abbildung 3.9: TREKKER im Einsatz

Die Eingabe bei diesem Gerät erfolgt über 5 Knöpfe und einem Steuerkreuz bzw. über das Touchpad. Des weiteren hat der iPAQ noch einen An/Aus Knopf und einen Navigationsknopf zum Durchblättern von Listen. Man kann Sprachaufzeichnungen abspeichern und Texte schreiben. Die Benutzung erfolgt über unterschiedliche Modi: Es gibt einen Navigationsmodus, einen Offlinemodus und einen Lesemodus.

In den unterschiedlichen Modi hat der Esc–Knopf überall die Semantik von Nein/Stop/Abbrechen, der Enter–Knopf bedeutet immer Ja/Weiter. Alle anderen Knöpfe haben eine dem Modus entsprechende Funktion.

Über das Touchpad kann Text eingegeben werden, wobei durch die Sprachausgabe entsprechendes Feedback gegeben wird. Der Sensorbereich ist dabei wahlweise als ein Braillekeyboard (Tab. 3.1) oder als eine Telefontastatur (Tab. 3.2) benutzbar. Bei der Braillemethode wird eine Tabelle bestehend aus 3 Spalten und 4 Zeilen benutzt, wobei die äußeren Spalten eine Braillezelle bilden. Es wird empfohlen, das Gerät in beide Hände zu nehmen und mit beiden Daumen zu tippen.

B1	Annehmen	B4
B2	Leerzeichen	B5
B3	Löschen	B6
B7		B8

Tabelle 3.1: TREKKER Braille Tastatur

Alternativ zum Braillekeyboard gibt es für Benutzer, die sich damit schwer tun oder keine Übung haben, die Telefontastatur (Tab. 3.2).

1	2,A,B,C	3,D,E,F
4,G,H,I	5,J,K,L	6,M,N,O
7,P,Q,R	8,S,T,U	9,V,W,X
*	0,Y,Z	#

Tabelle 3.2: TREKKER Telefon Tastatur

Prinzipiell, und darauf wird auch in der Gebrauchsanweisung [27] wiederholt hingewiesen, soll das Gerät nicht den weißen Stock oder den Blindenhund ersetzen sondern nur ergänzen. Wegen der Ungenauigkeit von GPS kann nicht festgestellt werden, auf welcher Straßenseite sich der Benutzer befindet.

Der Preis dieses Gerätes liegt bei 1500.– US$.

Kommentar: *TREKKER ist zur Zeit das vielversprechendste, kommerziell erhältliche Navigationssystem für Blinde. Durch die Möglichkeit, sich Statusinformationen über GPS ausgeben zu lassen, weiß der Benutzer ob das System korrekt funktioniert oder ob es zur Zeit unzuverlässlich ist. Da das System erst seit März 2003 auf dem Markt ist, werden in nächster Zeit sicherlich viele Verbesserungen und Neuerungen erscheinen. Die Handhabung des iPAQs scheint durch die wählbare Braille–Telefon Eingabe und die 5 Knöpfe gut zu funktionieren, doch ich kann mir vorstellen, dass sich während des Gehens die Benutzung als schwierig herausstellt. Ein Gerät, das man mit einer Hand halten und gleichzeitig bedienen kann, wäre gerade während der Navigation sicherlich praktischer. Der Signalton bei jeder Kreuzung und die sprachliche Richtungsausgabe lenken den Benutzer jedoch von seiner Aufmerksamkeit auf die Umgebung ab und vermutlich sind diese bei erhöhtem Straßenlärm bzw. lauter Umgebung schwer zu verstehen oder werden sogar überhört.*

3.2.6 NOPPA

NOPPA [17] ist ein 3 jähriges Pilotprojekt des Finnischen “Ministry of Transport and Communications Finland’s Passenger Information Programme” (HEILI) mit einem Budget von 500.000.– Euro. Das Projekt startete im Juni 2002 und endet im Dezember 2004.

Das System soll eine Hilfe für Sehbehinderte sein und Informationen über Verkehrsmittel bzw. Navigationshilfen für Fußgänger bereitstellen. Über das Internet werden aktuelle Informationen von öffentliche Datenbanken bezogen, damit der Wissensstand so aktuell wie möglich ist.

Der aktuelle Stand des Systems umfasst minimale Routenplanung und bietet Navigationsdienste in Helsinki, Espoo, Vantaa, Kauniainen and Tampere.

Die Hauptfunktionen von Noppa (Abb. 3.10, S. 24) sind:

- Informationen über öffentliche Verkehrsmittel
  - Fahrpläne, Identifikation von Bussen
  - Routenplanung
  - Echtzeitinformationen für Passagiere
- Navigation
  - Bestimmung der Innen– und Außenposition
  - Personalisierte Routenplanung und Führung
  - Richtungsinformationen
  - Warnungen bei Straßenarbeiten
- Kommunikation
  - PDA oder 3G Telefon
  - Spracherkennung
  - GPRS Serververbindung
  - SMS Dienst
  - News Dienste
- Lokale Informationen
  - Kurzstreckenkommunikation durch Bluetooth
  - POI[8] und AOI[9] Informationen
- Zusatzausrüstung
  - Gerät für Kollisionswarnung
  - Videokamera

Noppa wird in Zusammenarbeit von Finnischen Verkehrsbetrieben, dem Finnischen Ministerium für Transport und Kommunikation, der Stadt Espo, der Finnischen Blindenorganisation (FFVI), Timehouse, Microsoft uvm. entwickelt.

Kommentar: *Wenn NOPPA fertig gestellt ist und alle angekündigten Funktionen auch wirklich vorhanden sind, scheint das System eine Menge an Wünschen von blinden Fußgängern zu erfüllen. Durch die ständig aktualisierten Daten von öffentlichen Verkehrsmitteln usw. kann die Navigation auf einem optimalen Stand gehalten werden. Auch bei diesem System wurde die Ausgabe allein auf Sprache begrenzt. Das endgültige Ergebnis des Projektes wird man 2004 dann sehen.*

[8] Points Of Interest
[9] Areas Of Interest

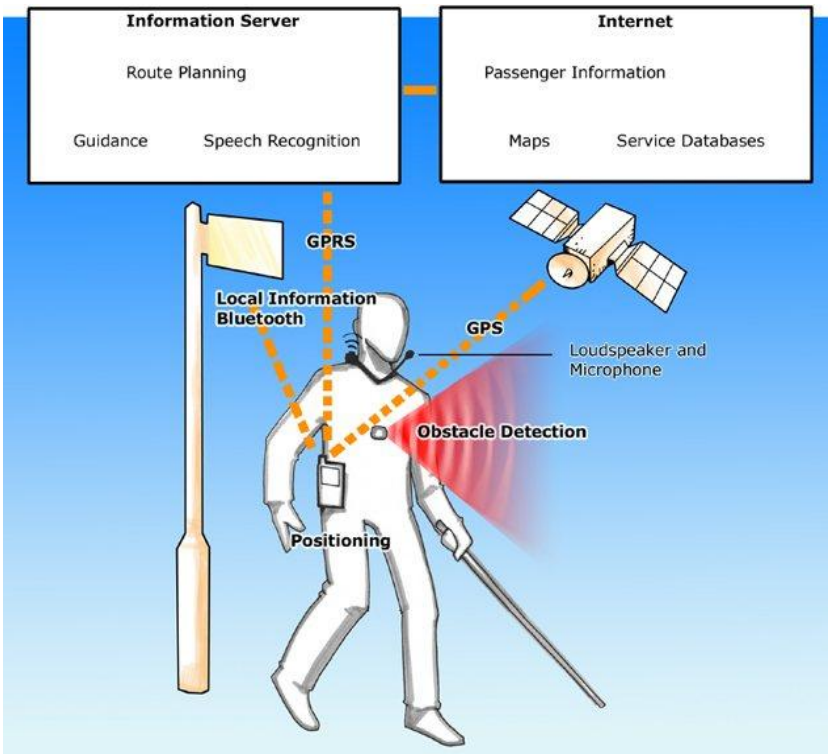


Abbildung 3.10: Die Komponenten von NOPPA

4 Lösungsansätze

An vielen Punkten der Entwicklungsphase gab es für die Lösung eines Problems mehrere Möglichkeiten, aus denen eine ausgesucht werden musste. Welche Lösung dabei genommen wurde, hing im wesentlichen von der Sinnhaftigkeit, der Machbarkeit und dem geschätzten Aufwand ab. Außerdem sollte die Lösung für den Prototyp einen Sinn ergeben.

Eingabe An verschiedensten Punkten der Benutzung müssen Eingaben in das System getätigt werden. Ich wollte mich auf die Problematik von **alternativen Eingaben** (siehe Kap. 5.1) konzentrieren, die zwar eher eine nebensächliche Rolle spielen, aber trotzdem benötigt werden. Ich habe **keine Spracheingabe** verwendet, da es zur Zeit keine freie[1] Bibliothek dafür gibt. Außerdem sollte das System portierbar bleiben und für PDAs gibt es noch keine Spracherkennung, die den Anforderungen entsprechen würde. Die Zieleingabe würde einen idealen Anwendungsfall für Spracheingabe darstellen, doch dies kann in einer Weiterentwicklung von NAVI eingebunden werden.

Anfangs war für die Eingabe von Text die Entwicklung einer **Braille–Tastatur** (siehe Kap. 5.1.3) geplant. Die Tatsache, dass ein ungeübter Benutzer diese nicht bedienen kann, war der ausschlaggebende Grund dies als weiterführende Entwicklung zu vermerken. Für den Prototypen wäre diese Funktionalität wahrscheinlich ein Feature, das zwar vorhanden aber von niemandem benutzt würde. Um mit dem Braille–Alphabet sinnvoll arbeiten zu können ist eine sehr große Einarbeitung nötig. Der relativ große programmtechnische Aufwand hätte sich deshalb kaum gelohnt.

Das Design der Eingabe wurde dahingehend gestaltet, dass es unabhängig von der konkreten Implementierung ist. Somit herrscht eine größtmögliche **Flexibilität** und die Eingabemodalitäten können bei Bedarf ausgetauscht und erweitert werden.

Joypad Der erste Vorschlag für das Eingabegerät war die Benutzung der am Laptop vorhandenen **Tastatur**. Da aber während der Navigation Eingaben benötigt werden und eine Tastatur in dieser Situation nicht zu gebrauchen ist, habe ich ein Joypad als Eingabegerät gewählt. Die Handhabung kommt einem eventuellen Seriengerät am nächsten. Die Vorteile sind vor allem, dass alle Knöpfe gut und klar erreichbar sind und der Druckpunkt bei den Tasten gut fühlbar ist. Es können mehrere Knöpfe ohne große Verrenkungen gleichzeitig betätigt und gewisse Funktionen können auch mit nur einer Hand erledigt werden.

Das von uns benutzte Joypad hat 8 Knöpfe und ein Steuerkreuz. Dadurch haben wir genug Tasten um verschiedenste Eingaben zu modellieren.

Ausgabe Bei der Navigation müssen Positionsinformationen, Umgebungsdaten, Feedback zu Benutzereingaben und Richtungsinformationen ausgeben werden. Da die Benutzer

[1] frei im Sinne von freier Software mit einer freien Lizenz wie GPL, BSD-License

hauptsächlich Blinde sind, fiel die Entscheidung, den Großteil der Informationen über **Sprachausgabe** auszugeben.

Das Gehör soll vor allem während der Navigation nicht zu sehr belastet werden. Deshalb werden Richtungsinformationen für das Abbiegen an Kreuzungen über **taktile Vibratoren** ausgegeben. Anfangs wollten wir mit Hilfe von elektronischen Impulsen, wie es bei den elektrischen Apparaten für den Muskelaufbau passiert, ein künstliches "nach hinten Ziehgefühl"[2] erzeugen, doch nach einigen selbst durchgeführten Tests war klar, dass Elektrostöße keinem Fußgänger über längere Zeit hinweg zumutbar sind.

Als Alternative bot sich die taktile Richtungsausgabe über Vibratoren an. Wie und wann man die Ausgaben am geeignetsten macht, wurde dann versucht zu erörtern (siehe Kap. 6).

Neben der Richtungsausgabe werden ununterbrochen Informationen über die Umgebung ausgegeben. Damit die Flut an Informationen nicht zu groß wird und dadurch ebenfalls das Gehör belastet wird, wurde ein Informationsfilter eingebaut. Dieser Bereich ist in der zweiten Diplomarbeit des Naviprojektes beschrieben [44].

Navigationssystem Um die Anforderungen an die Navigation zu erfüllen muss ein auf **Vektoren basierter Routenplaner** [3] benutzt werden. Da zur Zeit keine freien Navigationssysteme mit dieser Funktionalität existieren, war eine Eigenimplementierung notwendig. Ich hatte schon einige Erfahrungen mit der Boost–Bibliothek [1] gemacht und für diese Problematik schien dies das beste und einfachste Hilfsmittel zu sein. Um bei der Programmierung und auch bei der späteren Vorführung den aktuellen Status der Navigation und eventuelle Fehler zu erkennen, habe ich ein **Debug Fenster** für die Programmkontrolle eingebaut.

Die Implementierung des Navigationssystems wurde aber von Anfang an so unabhängig wie möglich von der Debug Ausgabe gestaltet, damit später einmal dieses von Blinden nicht gebrauchte Fenster ohne größere Probleme entfernt werden kann.

Karten In den Karten müssen neben Straßen und Kreuzungen noch Umgebungsdaten abgespeichert werden, die nicht nur Gebäude sondern auch detailliertere Objekte enthalten müssen. Es gibt zur Zeit nur kommerzielle Kartenanbieter, die die Daten in einem proprietären Format abgespeichert haben. Für den Prototypen sollte es reichen, eine einfache Versuchsstrecke selber auszumessen (siehe A.3). Dafür haben wir uns einen geeigneten Weg, an dem alle Funktionen getestet und vorgeführte werden können, um das Gebäude der TU–München herum gesucht. In diese Karte wurde dann die Umgebungsinformationen zum Testen eingefügt.

Positionsbestimmung Es gibt weltweit mehrere Systeme für die Bestimmung der Position. GPS ist das einzige flächendeckende System, das ohne zusätzliche Ausstattung der Umgebung auf jedem Punkt der Erde verfügbar ist. Außerdem ist es das zur Zeit genaueste System. Es gibt eine riesige Anzahl von Empfängern in allen Preisklassen, die

[2] Ist ein Elektro–Tab am Oberarm angebracht und wird eingeschalten, so fühlt sich dies durch die Kontraktion des Muskels ähnlich an, als ob jemand den Oberarm leicht nach hinten ziehen würde.

[3] Die Straßen müssen als richtige Linien dargestellt werden, wobei sich Endpunkte mehrerer Linien in einer Kreuzung treffen. Dadurch können Richtungsänderungen beim Abbiegen genau berechnet werden.

großteils alle für die Verbindung mit PCs und PDAs ausgerüstet sind. Diese Argumente lassen für die Positionsbestimmung keine Alternative zu GPS zu.

5 Benutzereingabe

Damit ein Navigationssystem den Benutzer auch wirklich an das gewünschte Ziel lotsen kann, muss der Benutzer dem System erstmal sagen, wohin er genau will. Außerdem gibt es viele Optionen, die, teilweise nur selten, angegeben werden müssen. Bei blinden bzw. sehbehinderten Benutzern treten bei Informationseingaben besondere Probleme auf.

5.1 Eingabe von blinden Benutzern

Soll der Sinn und Zweck eines minimalen Navigationssystems mit einem Satz erklärt werden, so würde dieser sicher

"Bring mich an mein gewünschtes Ziel."

lauten. Da die heutige Technik noch nicht so weit ist, um die Gedanken und Wünsche des Menschen lesen zu können, muss der Benutzer eine Eingabe in das System machen um seine Wünsche zu übertragen. Im Fall eines Navigationssystems muss der Benutzer also sein gewünschtes, anzusteuerndes Ziel dem System eingeben.
Bei einem vollwertigen Navigationssystem reicht diese Eingabe allein nicht mehr. Für viele Funktionen müssen Parameter, Einstellungen usw. angegeben werden können.

Es gibt mehrere Arten von Eingaben, welche sich mehr oder weniger stark unterscheiden. Ich habe mich für ein Joypad als Eingabegerät entschieden (siehe Kap. 4). Mit diesem können alle geforderten Eingaben einfach und schnell getätigt werden.

Im folgenden werde ich einige Formen der Benutzereingabe genauer betrachten und analysieren:

5.1.1 Eingabe von Ja - Nein Werten

Bei unserem Prototyp ist es der Fall, dass der Benutzer auf eine Umgebungsinformation mit "ja" bzw. "nein" antworten muss, ob die gerade eben ausgegebene Information für ihn interessant bzw. uninteressant ist. Wenn solche Eingaben häufig zu tätigen sind, so sollte der Weg zu dieser Funktionalität so kurz wie möglich sein.

Wird **eine Taste** zur Eingabe von ja - nein Werten benutzt, so kann man davon ausgehen, dass der Normalzustand der Taste den Wert 0 hat. Wird die Taste gedrückt, so bekommt sie den Wert 1[1]. Die Zeit zwischen einer Eingabe und der nächsten kann man als Zeitfenster sehen. Wenn der Benutzer vergisst in diesem Intervall die Taste zu drücken, so nimmt das System automatisch den Standardwert, in diesem Fall eben 0, an. Wenn bei einer Aktion

[1] 0 und 1 sind in diesem Zusammenhang einfach nur zwei unterschiedliche Werte

ein Timeout mit gewünschtem Standardwert bei Nichteingabe benötigt wird, so ist diese Version der ja - nein Eingabe sicherlich passend, doch im Fall einer zwingenden Eingabe hat das System keine Möglichkeit zu erkennen, ob das Nichtdrücken der Taste gewollt oder nur ein Versäumnis war. Ebenfalls kann beim Wert 1 festgestellt werden wie lange es gedauert hat, bis der Benutzer die Taste gedrückt hat, für den Wert 0 ist das nicht möglich.

Bei **zwei Tasten** wird beim Drücken der einen der Wert 0 und bei der anderen der Wert 1 eingegeben. Dies hat den Vorteil, dass das System immer genau weiß, ob und wann der Benutzer "ja" oder "nein" gesagt und gemeint hat. Der Nachteil hierbei ist aber, dass zwei Tasten an der Hardware angebracht werden müssen.

Im Falle unseres Prototypen habe ich die Methode mit den zwei Tasten benutzt und dafür zwei gesonderte Knöpfe am Joypad reserviert. Dadurch kann der Benutzer im System navigieren, also Menüs auswählen bzw. Eingaben machen, und inzwischen unabhängig davon für auftretende Ereignisse Feedback geben. Hätte ich z.B. die Esc– bzw. Enter–Funktion des Steuerkreuzes benutzt, so hätte der Benutzer immer wieder zwischen den zwei Systemzuständen "Navigation" und "Benutzer–System–Eingabe" hin und her wechseln müssen. Für diesen Statuswechsel wären mehrere Eingabefolgen nötig gewesen, was ich verhindern wollte. Intuitiv gesehen ist die Bewertung einer Information bezüglich ihrer Interessantheit eine spontane Entscheidung. Deshalb sollten die Schritte für so eine Bewertungseingabe so kurz wie möglich sein, in unserem Fall also direkt.

5.1.2 Listenauswahl

An mehreren Stellen bei der Benutzung eines Navigationssystems muss der Benutzer einen Wert aus einer Liste auswählen. Beispiele dafür sind z.B. Ausgabeart von Richtungsinformationen, Lautstärkeeinstellungen, Sprachwiedergabegeschwindigkeit usw.

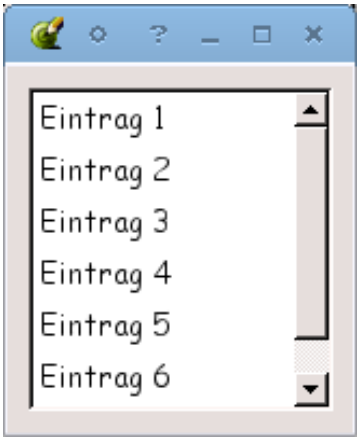


Abbildung 5.1: Ein Beispiel für eine Liste

Wie in 6.1 beschrieben, braucht der Benutzer auf jede Aktion ein Feedback, also eine Rückmeldung. Ist die Anzahl der Listeneinträge beschränkt, so kann die komplette Liste der Einträge auf einmal ausgegeben werden. Der Benutzer wählt in diesem Fall einen Eintrag nach dem anderen aus, macht bei Finden des gesuchten Listenelements halt und bestätigt dieses.

Bei normal sehenden Personen genügt meist ein kurzer Blick um die Übersicht über die Listeneinträge zu erhalten. Für Blinde und sehbehinderte Personen hingegen ist diese Orientierung schwieriger. Deshalb muss bei den Einträgen in der Liste wenn möglich eine Sortierung bzw. eine erkennbare Abfolge bzw. Abgrenzung innerhalb der Listenelemente existieren. Damit kann der Benutzer schon bevor er das gewünschte Element erreicht abschätzen, ob bzw. wann dieses auftritt. Man muss prinzipiell davon ausgehen, dass ein sehbehinderter Benutzer, der auf ein anderes Feedback als visuelle Ausgabe angewiesen ist, beim Durchsuchen der Liste langsamer ist als ein sehender Benutzer, der sich auf einen "Blick" eine Orientierung verschaffen bzw. den aktuellen Eintrag erfassen kann.

Wenn die Anzahl der Elemente in einer Liste eine gewisse Größe erreicht hat, so wird es ziemlich langwierig aus dieser Flut an Möglichkeiten den richtigen Eintrag auszuwählen. Man kann sich leicht vorstellen, dass z.B. eine Liste aller Straßennamen inkl. Hausnummern von München weit mehr als 1 Million Einträge haben würde. In diesem Fall kommt man um eine Unterteilung der Liste nicht herum: Die Liste wird in mehrere Unterbereiche unterteilt, wobei diese Unterbereiche wieder aus anderen Unterbereichen zusammengesetzt sein können.

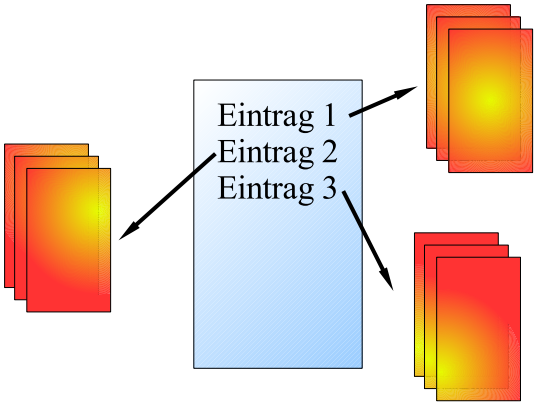


Abbildung 5.2: Unterteilung einer Liste in Unterlisten

Die Unterbereiche können dabei eine Gruppierung von Eigenschaften bzw. Gemeinsamkeiten der Elemente in ihnen sein, wie z.B. Anfangsbuchstabe, Bezirk von Straßennamen usw. (Abb. 5.2, S. 30).

Ein Beispiel: Jemand will aus einer Liste von möglichen Navigationszielen in München genau die Arcisstraße 21 heraussuchen. Die Liste kann dabei in (*Stadtbezirk: Max Vorstadt, Schwabing, ...*) → (*Anfangsbuchstaben der Straßen in dem Bezirk: A, B, G ...*) → (*Ersten zwei Zeichen der Straßen mit dem Anfangsbuchstaben in dem Bezirk: Ar, Au, ...*) → ... → (*gewolltes Ziel: Arcisstraße*) unterteilt werden.

5.1.3 Texteingabe

Die Eingabe von z.B. einer Notiz kann mit einer Liste nicht mehr effizient gemacht werden. Man könnte zwar jeden Buchstaben einzeln aus dem Alphabet auswählen, doch für größere

Texte wird dies zu einer langwierigen und anstrengenden Arbeit.

Bei normalen Computer–Systemen werden Texte mittels einer Tastatur eingegeben. Eine für Blinde handhabbare[2] Tastatur könnte zwar von der Funktionalität her den Anforderungen entsprechen, scheidet aber wegen Unbenutzbarkeit im mobilen Betrieb und wegen der Größe aus. Es sei dem Leser überlassen im Selbstversuch stehend mit verbundenen Augen, die Tastatur in der Luft haltend, einen Text einzugeben.

Mit der sog. Braille–Tastatur (Abb. 5.3, S. 31) können heutzutage blinde Benutzer an vielen Geräten schnell und einfach Text eingeben. Die Tastatur besteht aus 8 Tasten. Durch deren Kombination werden einzelne Buchstaben oder auch, ähnlich wie bei der Stenographie, einzelne Silben eingegeben. Im Rahmen eines anderen Projektes konnte ich eine kleine Versuchsreihe mit blinden Stenotypisten durchführen. Im Vergleich zu Sehenden, die Texte über normale Tastaturen eingeben, konnte ich keinen wesentlichen Geschwindigkeitsunterschied in der Eingabe feststellen. Dies zeigt, dass die Braille–Tastatur mit entsprechender Übung gut für Texteingabe geeignet ist.

Abbildung 5.3: Braille Tastatur: Die Hände sind auf den Eingabetasten, darüber befindet sich die Brailleausgabe

Neben den herkömmlichen Eingabesystemen - sei es für normal sehende, sei es für blinde Benutzer - tendiert die Entwicklung immer mehr zur Spracherkennung und –eingabe. Die heutige Technik ist leider noch nicht 100%ig funktionstüchtig (v.a. unter nicht optimalen Bedingungen). Wenn diese Eingabemethodik jedoch ausgereift ist, dann werden sicherlich viele alternative Eingabemethoden verdrängt und viele Probleme gelöst werden. Bis zum Enterprise[3] Dialog

Mensch: *ich will zum nächstgelegenen Imbiss, aber kein Inder*

[2] handhabbar deswegen, weil die Tasten auf so einer Tastatur nicht zu klein sein dürfen, damit auch ein Blinder sie ertasten und drücken kann

[3] Raumschiff Enterprise ist eine US-Fernsehserie über die Abenteuer einer Weltraumbesatzung in der Zukunft. Der Bordcomputer der Enterprise kommuniziert über eine Frauenstimme mit der Besatzung. Dabei nimmt er alle möglichen Formen von Befehlen entgegen und führt immer genau das aus, was der Befehlende gemeint hat.

Computer: *der nächste Chinese ist in 160 Metern, gehen Sie erstmal gerade aus*
wird es noch etwas dauern, doch unmöglich gilt es schon lange nicht mehr.

6 Ausgabe an den Benutzer

Bei einem Navigationssystem für Blinde müssen unterschiedliche Daten ausgegeben werden:

- Auf Eingaben muss der Benutzer ein **Feedback** bekommen, ob die Eingabe erkannt wurde.
- Während der Navigation muss das System dem Benutzer durch **Richtungsausgaben** mitteilen, wohin er gehen muss.
- **Umgebungsdaten** und **Informationen** wie Zeit müssen ausgegeben werden.
- Die aktuelle **Position** hilft dem Benutzer bei seiner Orientierung.

Die unterschiedlichen Ausgaben stellen für sich gesonderte Probleme dar. Feedback muss schnell ausgegeben werden, da meistens mehrere Aktionen hintereinander auszuführen sind. Richtungsdaten sind von der Größe her begrenzt, können aber unerwartet auftreten. Umgebungsdaten hingegen sind größer, weil mehr Informationen übermittelt werden müssen.

6.1 Feedback auf Eingabe

Wurde eine Eingabe (siehe Kap. 5.1) getätigt, so weiß der Benutzer nicht, ob sein Vorhaben geglückt ist oder nicht. Bei Benutzerinterfaces für normal sehende Personen wird eine getätigte Aktion meist visuell dargestellt bzw. ausgegeben. Wenn z.B. ein Element in einer Liste ausgewählt wird, so wird in der am Bildschirm dargestellten Liste dieses durch eine Farbe untermalen. Bei der Visualisierung einer Dialogbox erscheint ein neues Fenster bzw. eine neue Fläche und durch einen Titel bzw. durch eine Überschrift erkennt der Benutzer, dass die von ihm gewollte Aktion auch ausgeführt worden ist und dass das System nun auf eine weitere Eingabe wartet.

Bei einem Navigationssystem für blinde Benutzer muss solches Feedback in einer anderen Art und Weise generiert werden, da wenn überhaupt ein Bildschirm vorhanden ist, dieser nicht erkannt bzw. nicht genutzt werden kann.

Lösungsansätze für diese Problematik gibt es zur Zeit mehrere - einige haben sich schon über mehrere Jahre bewährt, andere sind noch in der Anfangsphase.

Aktuelle Hilfsmittel

Zur Zeit gibt es eine Vielzahl an Hilfsmitteln, die blinden Personen die Benutzung von Computern ermöglichen sollen. Viele dieser Systeme sind schon seit geraumer Zeit in Verwendung und haben sich als besonders nützlich und brauchbar erweisen, weshalb sie auch für ein Navigationssystem eine sinnvolle und verwendbare Methodik liefern.

- **Braillezeile** Eines der ersten Hilfsmittel für Blinde und Sehbehinderte war die Braille-

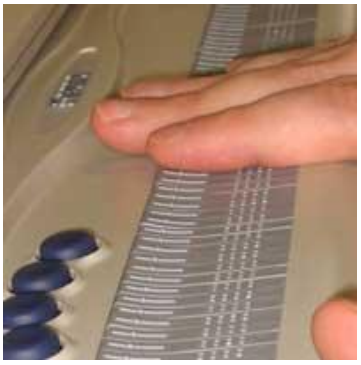


Abbildung 6.1: Braillezeile für die Ausgabe von Text

zeile. Bei diesem Gerät werden Bildschirmausgaben wie z.B. Texte auf einer langen Blindenschriftzeile (Abb. 6.1, S. 34) dargestellt. Die Blindenschrift besteht aus tastbaren Erhebungen wobei in 6 Punkten die Buchstaben kodiert werden (Abb. 6.2, S. 34). Auf der Brailleausgabe wird Zeile für Zeile der Text ausgegeben und der Benutzer liest durch Darübertasten die Worte. Diese Art der Ausgabe ist weit verbreitet und hat sich v.a. beim Lesen von Texten sehr bewährt.

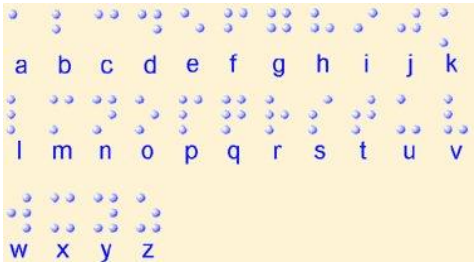


Abbildung 6.2: Braille Alphabet

Für ein Navigationssystem scheint diese Ausgabeform nicht sonderlich geeignet, da v.a. im mobilen Betrieb das Lesen der Braillezeile sich als schwierig erweist. Die Finger müssen kleinste Unterschiede ertasten und wenn keine feste Auflage für das Gerät gegeben ist, führen Wackeln und Zittern zu Ungenauigkeiten. Außerdem muss für jegliche Datenausgabe das Gerät in der Hand gehalten werden und die lesende Hand sich auf der Braillezeile befinden.

- **Bildschirm Lupe, Vergrößerungssoftware** Wenn bei sehbehinderten Personen noch ein gewisser Sehrest vorhanden ist, dann können Bildschirmausgaben vergrößert bzw. durch Bildbearbeitungsalgorithmen gefiltert werden um die Lesbarkeit zu verbessern oder aber erst zu ermöglichen. Vergrößerung wird entweder durch eine optische Lupe, die vor dem Monitor angebracht ist, bzw. durch Software erreicht. Bei zweitem besteht auch noch die Möglichkeit mit Hilfe von Bildverarbeitung den Kontrast bzw. die Helligkeit zu verbessern.
  Diese Arten von Hilfsmittel scheinen für ein mobiles Navigationssystem nicht sonderlich geeignet. Das Navigationsgerät soll so klein und handlich wie möglich gehalten werden, weshalb die Größe für eine eventuelle Bildschirmausgabe sehr klein ausfällt, ca. maximal einem Handhelddisplay gleich. Auf so einer Fläche vergrößerten Inhalt darzustellen ist nicht sonderlich sinnvoll.

- **Bildschirm Lesesoftware** In den letzten Jahren wurden im Bereich der automatischen, computergestützten Sprachsynthese große Fortschritte gemacht. Mittlerweile können mit annehmbarer Rechenleistung Texte akustisch ausgegeben werden. Die dabei erzielte Qualität ist durchaus akzeptabel und durch die Möglichkeit, die Geschwindigkeit des gesprochenen Textes zu beschleunigen bzw. zu verlangsamen, kann eine erhebliche Arbeitsgeschwindigkeitssteigerung erzielt werden.
  Die akustische Ausgabe von Informationen ist die zielführendste Art für Feedback für ein Navigationssystem für Blinde. Aktuelle Menüeinträge, Fensternamen, Auswahlen können vorgelesen und ausgeführte Aktionen können akustisch bzw. textual beschrieben werden.

- **PAC Mate BNS**

Abbildung 6.3: PAC Mate BNS

Dieses Gerät der Firma Freedom Scientific [20] ist ein speziell für Blinde ausgelegter Pocket PC. Die Eingabe erfolgt durch ein Braille System bestehend aus neun Tasten. Über ein Steuerkreuz kann in Menüs navigiert werden.

Als Ausgabe für Feedback und Daten dient die Sprachsynthesesoftware JAWS.
Die Größe, Ein– und Ausgabemethoden und die Funktionalität dieses Gerätes zeigen, wie die Hardware eines Navigationssystems für Blinde aussehen bzw. zu handhaben sein könnte.

6.1.1 Besondere Entwicklungen

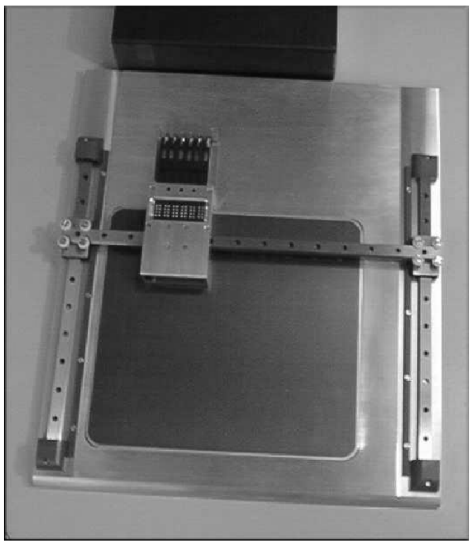


Abbildung 6.4: Anzeige von Bildern durch ein taktiles Display

An der Uni Heidelberg wurde das **Taktile Display für das Anzeigen von Bildern** entwickelt [41]. Auf einem umgebauten IR–Maussensor ist eine Braillezelle montiert (Abb. 6.4, S. 36). Diese Zelle hat 2 Bewegungsfreiheiten (nach oben und nach unten) womit eine rechteckige Fläche abgefahren werden kann. Über mehrere Verfahren wird das darzustellende Bild so umgerechnet, dass es mit Hilfe der Braillezelle "ertastet" werden kann.
Dieses Gerät könnte bei einem Navigationssystem dem Benutzer z.B. den noch zu verfolgenden Weg anzeigen. Eine andere Möglichkeit wäre, dass die aktuelle Umgebung dargestellt wird und der Benutzer das ihn umgebende Areal abtastet.

6.2 Richtungsausgabe durch vibrierende Tabs/Sticks

Blinde bzw. sehbehinderte Menschen orientieren sich im Leben und v.a. auch im Straßenverkehr großteils mit Hilfe des Gehörs. Da der Sehsinn degeneriert bzw. gar nicht vorhanden ist, müssen Sie ihre Umgebung anderweitig wahrnehmen. Der Tastsinn dient vor allem der genaueren Erkennung kleiner, direkt berührbarer Objekte. In der Straßennavigation dient der weiße Langstock als verlängerte Tasthilfe, wodurch der zurückzulegende Weg Schritt für Schritt abgetastet wird. Die weiterreichende Umgebung jedoch kann und wird nur mit dem Gehör wahrgenommen. So muss sich der Blinde auf seinen Hörsinn verlassen, wenn er versucht zu erkennen, ob an der Kreuzung, an der er gerade steht und die er überqueren will, die Fußgängerampeln gerade für ihn auf grün oder auf rot stehen. Mit einem geschulten Ohr kann der Blinde den Verkehr, der gerade in Gehrichtung fließt bzw. auf grün wartende Autos erkennen und somit abschätzen, ob für ihn gerade grün oder rot ist. Auch auf dem Weg zu Kreuzungen wird mit Hilfe des Gehörs wahrgenommen, ob man sich gerade einer solchen Straßenüberquerung nähert oder ob der Weg einfach nur geradeaus geht.

Bei einem Navigationssystem für Blinde ist es gerade bei den oben beschriebenen Situationen sehr wichtig, dass das Gehör so wenig wie möglich belastet wird. Trotzdem sollte aber ein Navigationssystem auch mitteilen, dass man sich gerade einer Kreuzung nähert bzw. dass man abbiegen soll. Auch wenn die Straße eine Biegung macht sollte das Navigationssystem den Benutzer vorwarnen. Mehrere Betroffene haben mir von öfteren Erlebnissen erzählt, wobei sie mit dem Langstock irgendwo dagegen rennen und sich den Handgriff in den Bauchbereich stoßen. Im ersten Moment wissen sie nicht ob sie nach links oder rechts ausweichen sollen um dem weiteren Wegverlauf zu folgen. Auch das Stolpern über einen Bordstein ist normaler Bestandteil des täglichen Lebens.

Die einzig sinnvolle Ergänzung zum Hörsinn bei Richtungsausgaben ist der Tastsinn.

Abbildung 6.5: Taktile Ausgabe über Vibrationssticks bei NAVI

Mit Hilfe von kleinen Vibratoren, im weiteren Text Tabs genannt[1], die entweder am Oberarm, am Unterarm, an der Hose oder wo immer man links und rechts unterscheiden kann, angebracht sind, können Richtungsangaben bzw. Warnsignale ausgegeben werden.

Folgende Signalvarianten sind dabei z.B. möglich:

- Vibration LINKS:
  Das bedeutet, dass links abgebogen werden muss
- Vibration RECHTS:
  Es muss rechts abgebogen werden
- Vibration RECHTS + LINKS:
  Achtung! Eine Gefahrenstelle nähert sich, höhere Aufmerksamkeit ist erforderlich.

Neben den Primitivrichtungen LINKS, RECHTS, LINKS + RECHTS können die Tabs noch durch Länge und Reihenfolge der Vibrationen Informationen ausgeben. So kann man z.B. einen falschen Weg durch abwechselnde Vibrationen von LINKS, RECHTS, LINKS, RECHTS signalisieren. Wenn der Benutzer nur auf eine leichte Krümmung in der Straße zugeht, dann können die Tabs z.B. nur kurz vibrieren, bei einer größeren Abbiegung hingegen länger.

Bei der Benutzung des NAVI–Prototypen (siehe Kap. 7) sind wir auf mehrere Probleme bezüglich der Richtungsausgabe mit Vibration gestoßen. Arbeitet die Positionsbestimmung

[1] Der Begriff Tabs hat sich während der Diplomarbeit eingeprägt. Als erste Idee wollten wir Vibratoren in Tab–Form verwenden, doch dann haben sich die Handysticks ergeben. Der Name Tab ist aber geblieben.

nur ungenau, so ist der Zeitpunkt für die Vibrationsausgabe schwer zu ermitteln. Es kann passieren, dass die gemessenen Position des Benutzers bereits auf der Kreuzung liegt, obwohl dieser noch einige Meter davon entfernt ist. In diesem Fall würde die Richtungsausgabe viel zu früh geschehen. Andererseits, und das ist schlimmer, könnte er schon längst an diesem Punkt vorbeigegangen sein und die Vibrationsausgabe kommt viel zu spät. Bei der NAVI–Version haben wir das Vibrationssignal innerhalb eines Radius (ca. 4 Meter) um den Kreuzungspunkt ausgegeben. Betritt der Benutzer diesen Kreis, so vibriert der entsprechende Tab. Durch diese Methode werden Positionsmessungen, die dem Benutzer vorausgehen, zwar noch früher ausgegeben, doch wenn die Messungen dem Benutzer nachfolgen bekommt dieser zumindestens früh genug eine Ausgabe. Die Vibration ist dadurch nicht mehr als "sofort abbiegen" zu interpretieren, sondern bedeutet "in den nächsten Metern bzw. wann möglich abbiegen".

Neben dem Zeitpunkt ist für die taktile Ausgabe vor allem die Sequenz wichtig. Eine kurze, unerwartete Vibration kann eventuell nicht bemerkt werden. Deshalb ist, wie es auch in Kap. 3.2.2 gemacht wird, eine mehrfache Ausgabe nötig. So könnte z.B. bei jeder Richtungsausgabe als erstes Warnsignal eine starke Vibration darauf hinweisen, dass nun eine Richtungsangabe folgt und diese kann dann aus kurzen bzw. langen Vibrationen zusammengesetzt sein.

6.3 Ausgabe von Daten

Ein Navigationssystem, das nur Richtungen ausgeben kann, ist wenig sinnvoll. Vor allem wenn Umgebungsinformationen, Navigationsinformationen, Statusinformationen usw. ausgegeben werden sollen, reichen Vibrationstabs und akustische Signale nicht mehr aus. Dadurch, dass blinde Personen keine Bildschirmausgabe lesen können, bleibt als einziges funktionierendes Mittel die Sprachausgabe übrig.

In der PC Welt arbeiten die meisten Bildschirmausgabesysteme für Blinde mit Sprachsyntheseprogrammen. Dabei wird der am Monitor dargestellte Text vom Computer in Sprache umgewandelt und so ausgegeben. Bei solchen Programmen ist die Lesegeschwindigkeit, die Sprachhöhe und Sprachform parametrisiert, wodurch v.a. bei erfahrenen Benutzern die Ausgabe schlussendlich wenig mit normalem Sprechen zu tun hat: Durch ausreichend Übung und Erfahrung kann die Lesegeschwindigkeit gegenüber dem Normalsprechen um ein vielfaches gesteigert werden, wodurch der Informationsfluss erheblich beschleunigt wird und so auch längere Texte effektiv gelesen werden können.

Bei einem Navigationssystem für Blinde - und so auch bei unserem Prototypen NAVI - ist die Sprachausgabe die einzige verwendbare Ausgabe für Texte (siehe Kap. 6.1).

6.3.1 Ausgabe von absoluten Positionsinformationen

Neben den allgemeinen Umgebungsdaten (z.B. *"Sie befinden sich auf der Theresienwiese"*) und z.B. Zeitansagen (*"Es ist 13 Uhr und 45 Minuten"*) sind bei einem Navigationssystem Positionsausgaben nötig.

Im Gespräch mit potenziellen Benutzern hat sich herauskristallisiert, dass viele es als sehr sinnvoll und hilfreich erachten würden, wenn sie die Möglichkeit hätten, sich auf Wunsch

ihre aktuelle Position in Bezug auf einen größeren Orientierungspunkt ausgeben lassen zu können. In unbekannten Städten nützt der Name der Straße, in der man sich gerade befindet, nicht viel. In vielen Situationen weiß ein Blinder v.a. nach einigen Abbiegungen und Kreuzungen nicht mehr, wo er sich jetzt eigentlich genau befindet bzw. wohin er gerade geht. Die komplette Orientierung findet nämlich im Kopf statt - er muss sich merken wo und wie er abgebogen ist um seinen bisherigen Weg nachvollziehen zu können.

Aus diesem Grund sollte ein Navigationssystem für Blinde auch die Fähigkeit besitzen, die aktuelle Position in Bezug auf ein größeres Gebäude bzw. einen Platz auszugeben. Es ist erwiesen, dass vor allem blinde Personen ein sehr gutes und ausgeprägtes räumliches Vorstellungsvermögen haben.

Beispiel: *Der Benutzer befindet sich am Eingang des Stammgeländes der TU München: "Sie befinden sich 150 Meter nördlich des Königsplatzes"*

In dieser Ausgabe sind folgende Informationen enthalten, die dem Benutzer helfen sich eine Orientierung zu verschaffen:

- **Königsplatz** - Orientierungspunkt
- **Nördlich** - Auf der virtuellen Straßenkarte im Kopf im Norden
- **150 Meter** - Die Entfernung - wird mit Gehzeit assoziiert

6.3.2 Ausgabe von relativen Richtungsinformationen

Geht der Benutzer eines Navigationssystems in einer Stadt von Punkt A nach Punkt B, so kommt er mehr oder weniger oft an Geschäften, Imbissen, Ämtern usw. vorbei. Da diese Umgebung nicht gesehen werden kann, muss das Navigationssystem dies dem Benutzer mitteilen.

Neben der Beschreibung ist es auch nötig die Position der Lokalitäten auszugeben. Eine Erfahrung die Blinde immer wieder machen und leicht schmunzelnd erzählen, ist wenn sie auf der Straße nach dem Weg bzw. nach einem Gebäude fragen. Die Antwort: "Da vorne rechts", wobei mit der Hand die Richtung gezeigt wird, ist häufig der Fall. Diese Information von zwar hilfsbereiten Passanten ist wenig nützlich. Erstens sieht ein Blinder nicht, wo der Helfer gerade hinzeigt und zweitens sind Angaben wie "vorne" und "rechts" zu ungenau.

Bei üblichen Navigationssystemen werden solche Informationen mit "in 10 Metern rechts" oder "vorne links" ausgegeben, was für Blinde aber relativ wenig aussagt. Im Gespräch mit Blinden hat sich herausgestellt, dass Entfernungsangaben in Metern relativ gut abgeschätzt werden können - Richtungsangaben jedoch schwierig zu interpretieren sind.

Die Idee Richtungsangaben in der "Ziffernblattmethode" anzugeben, also statt "vorne rechts" lieber "auf 2 Uhr" zu sagen könnte da genauer und dadurch hilfreicher sein.

Dadurch dass bei einem Navigationssystem die Gehgeschwindigkeit bekannt ist, könnte man außerdem noch überlegen, statt der Entfernung in Metern die Entfernung in Zeit anzugeben, also die noch zu navigierende Zeit, bis man an dem besagten Punkt angekommen ist.

7 NAVI Prototyp

Abbildung 7.1: Ich mit dem NAVI Prototypen auf einer Trage auf dem Rücken

Um die vielen Überlegungen, die während der Einarbeitung in die Thematik angestellt und Problemlösungen die erarbeitet wurden, in etwas Angreifbares und Vorzeigbares zu packen, haben wir einen Prototypen gebaut. Dieses Navigationssystem für Blinde sollte als "proof of concept" dienen und die Möglichkeiten bzw. Funktionalitäten aufzeigen, die für die Navigation von blinden und sehbehinderten Menschen bereitstehen. Der Schwerpunkt wurde dabei vor allem auf die Neuerungen wie die Vibrationsausgabe und der Filterung der Umgebungsinformationen gelegt - Bestandteile, die bis zu dem Zeitpunkt laut meiner Nachforschungen erst wenig bzw. gar nicht beachtet wurden.

Der NAVI - Prototyp (Abb. 7.1, S. 40) besteht aus einem Laptop, auf dem DWARF läuft, einem GPS–Empfänger der die Positionsinformationen liefert, einem USB–Joypad (Abb. 7.2, S. 41) über das die Benutzereingaben getätigt werden und einer selbst gebauten Schaltung an der parallelen Schnittstelle, über die die Richtungsinformationen über 2 vibrierende Tabs (Abb. 7.3, S. 42) ausgegeben werden.

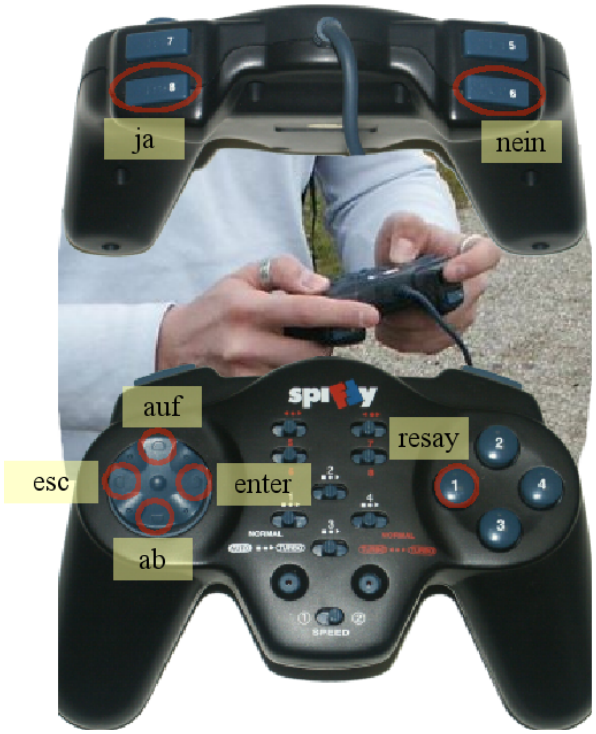


Abbildung 7.2: Das Joypad des NAVI Prototypen mit Tastenbelegung

7.1 DWARF

DWARF steht für Distributed Wearable Augmented Reality Framework und wurde an der TU–München als grundlegendes Framework, um Augmented Reality einfacher umzusetzen, entwickelt. DWARF stellt dem Anwender eine auf CORBA [2] basierende Middleware zur Verfügung.

Zum Konzept von DWARF gehört es, unabhängige Programmteile (Services) zu erstellen, welche mit einer möglichst allgemeinen aber einfachen Schnittstelle ausgestattet sind. Daten werden über ein auf Nachrichten (Events) basiertes Kommunikationssystem vermittelt. Dienste können Nachrichten erstellen, welche an einen oder mehrere andere Services adressiert sind. DWARF übernimmt die korrekte und effiziente Zustellung dieser.

Das Verschalten, oder anders formuliert das Auffinden eines geeigneten Empfängers für eine Nachricht übernimmt ebenfalls DWARF zur Laufzeit. Services kommunizieren über IP basierte Dienste untereinander, und müssen weder auf dem gleichen Rechner ausgeführt werden noch auf dem gleichen Betriebssystem aufsetzen.

Abbildung 7.3: Rechter Vibrationstab an der Hose angebracht

Ziel des Frameworks ist es für möglichst viele allgemeine Nachrichtentypen Services zu entwickeln, die dank DWARF und CORBA mit verschiedenen Programmiersprachen erstellt und wiederverwendbar sind.

Servicemanager

Pro beteiligtem Rechner [1] wird ein **Servicemanager** benötigt. Dieser ist zentraler Bestandteil von DWARF . Wenn ein Service gestartet wird, meldet sich dieser am Servicemanager an. Dadurch registriert der Servicemanager den Servicenamen, seine **Needs** und seine **Abilities**. Andererseits kann der Servicemanager Services auch bei Bedarf automatisch starten (falls der Service dafür entsprechend konfiguriert wurde).

Services

Services sind eigenständige Programme, welche über ihre **Needs** und **Abilities** mit den anderen DWARF –Komponenten kommunizieren.

Abilities: Abilities sind Datenschnittstellen, welche vom Service zur Verfügung gestellt werden. Abilities besitzen einen

Typ, welcher einen IDL-Datentypen referenziert, ein oder mehrere

Attribute, welche die Semantik des Datentyps für die aktuelle Anwendung genauer spezifizieren und einen

[1] DWARF setzt in der aktuellen Implementierung auf den slpd [19] auf.

Namen, welcher eine möglichst aussagekräftige Beschreibung des Interfaces darstellen sollte.

Needs: Ein **Need** ist eine eingehende Schnittstelle. Die Daten für diese Schnittstelle werden von anderen Services über deren Abilities zur Verfügung gestellt. Needs bestehen aus einem

Typ, welcher einen IDL–Datentypen referenziert, einem

Prädikat, welches die Semantik des angeforderten Datentyps genauer spezifizieren und einem

Namen, welcher eine möglichst aussagekräftige Beschreibung des Interfaces darstellen sollte.

DWARF beherrscht mehrere Kommunikationsprotokolle und Verschaltungstypen, in NAVI setzen wir auf rein eventbasierte Kommunikation (Sender–Verbraucher Prinzip).

Der Servicemanager versucht nach dem Start der Services jedem **Need** eine **Ability** zuzuordnen. Eine Verknüpfung (Verschaltung) zwischen Need und Ability zweier Services findet statt, falls deren Typ, Attribute und Prädikate einander entsprechen. Die Verschaltung kann sich zur Laufzeit dynamisch ändern/anpassen, da neue Services gestartet bzw. laufende Services gestoppt werden können.

7.2 Die Komponenten von NAVI

Am Ende der Realisierungsphase des Projektes NAVI entstand folgender Prototyp. Um das Projektziel auch nur annähernd innerhalb von 6 Monaten zu erreichen haben wir eine Vielzahl von freien Bibliotheken eingesetzt.

Die Zeit reichte nicht aus um das System auf einen Handheld–Computer zu portieren, es wurde aber mit dem Gedanken erschaffen, später portierbar zu sein.

Derzeitige Hardwareanforderungen [2]:

- GPS–Empfänger, der als Ausgabe NMEA–0183 [16] beherrscht.
- Tabs für Taktile Ausgabe
- USB Joypad
- Notebook 256 MB RAM, Pentium 4 2.4 GHz, mit folgenden Schnittstellen:
  - Serielle Schnittstelle R232 für GPS Empfänger
  - Parallele Schnittstelle (IEEE 1284) für Taktile–Ausgabe
  - USB 1.0/2.x für Joypad–Eingabe
  - Soundkarte für Sprachausgabe

[2] Wir haben mit dieser Hardware das System entwickelt und alle Vorführungen im Rahmen des Projektes mit dieser durchgezogen. Die Bibliotheken sollten portierbar sein und die Prozessor- und Speicheranforderungen um ein Vielfaches niedriger als hier von uns angegeben.

Als Software bringen wir GNU/Linux [8] zum Einsatz. Benötigt wird mindestens die Kernelversion 2.4.15. Wir verwendeten die Distribution **Debian Linux unstable** vom 18.11.2003 mit einem 2.6.0–test9 Kernel.

Bei der Wahl der Bibliotheken haben wir uns entschlossen, nur reine OpenSource Produkte zu wählen. Dadurch gewährleisten wir, dass unser Projekt NAVI später im universitären Umfeld usw. weiterentwickelt werden kann.

Die Linux Distribution Debian hat eine sehr umfangreiche Softwaresammlung und eine konservative Einstellung zu freien Softwarelizenzen [3]. Dies stellt sicher, dass keine Unsicherheiten bezüglich der Lizenzen in Zukunft zu erwarten sind.

Vorausgesetzte Software Pakete die zum Betrieb erforderlich sind:

Grafische Oberfläche: Als Debug–Plattform für sehende Entwickler haben wir uns für das Kommon Desktop Environment (KDE) [14] in der Version 3.1.4 entschieden. Die Oberfläche ist einfach zu verwenden und durch Entwicklerprogramme wie den QT–Designer [22] können Programmoberflächen schnell und einfach generiert werden. KDE beruht auf der Bibliothek QT [23] 3.2.1 der Firma Trolltech.

Die Softwareteile von NAVI , in denen eine grafische Oberfläche Verwendung findet, sind funktional in eigene Bestandteile ausgegliedert. Dadurch können diese Debug–Ausgaben später ohne große Änderungen entfernt werden [3].

Joypad API: Als Eingabegerät haben wir uns für ein Joypad entschieden, da diese unter Linux einfach anzusprechen sind, wenig kosten und viele Tasten besitzen. Standardschnittstelle ist die libjsw [29] Version 1.5.0.

Unsere Entscheidung fiel auf diese Bibliothek, da sie uns einen einfachen Zugriff auf die Benutzereingaben ermöglicht. In dem dafür zuständigen Programmteil muss nur an einer Stelle abgefragt werden, ob ein Joypad–Event aufgetreten ist. Wenn das der Fall ist, werden alle gedrückten Knöpfe ausgelesen und somit kann der aktuelle Status der Eingabe verarbeitet werden. Falls kein Joypad verfügbar ist, kann mittels **readline** [9] die Eingabe auch über eine Konsole erfolgen.

Datenbanksystem: Als Datenbanksystem haben wir uns für die speicherschonende Variante SQLite [24] Version 2.8.6 entschieden.

Bei SQLite wird kein externes DBMS [4] benötigt. Die Daten werden in einer einzigen Datei abgelegt, auf die während der Benutzung mit normalen SQL92-Anfragen zugegriffen werden kann. Sie ist für viele kleine Anwendungen durch die ressourcenschonende Implementierung ideal. Bezüglich der Geschwindigkeit kann sie mit anderen DBMS verglichen werden, ist aber eine der wenigen Datenbanken die auch auf Handheld–PCs läuft.

Algorithmen: Zur Implementierung der Wegesuch–Algorithmen setzten wir auf die Boost [1] STL Headers Version 1.30.2. Diese implementieren alle gängigen Wegesuchverfahren äußerst effizient.

[3] In unserer Grundüberlegung gehen wir davon aus, dass Blinde und Sehbehinderte Menschen keinen Nutzen aus einer visuellen Oberfläche ziehen, sehr wohl aber sehende Entwickler

[4] Datenbank Management System

Bei NAVI werden die aus der Datenbank ausgelesenen Daten in einen Boost-Graph konvertiert. Dieser Graph wird während der Programmausführung zum Berechnen der Navigation im Speicher gehalten. Die Wegesuchalgorithmen bestimmen, auf dessen Datenstruktur aufbauend, den aktuell kürzesten Pfad zum Ziel.

SVM [5] Lernverfahren: Die Bibliothek Torch3 [26] liefert alles was gebraucht wird um Lernalgorithmen anzuwenden. Zum Einsatz kam die Version 0.0-2.

Lernverfahren beruhen auf vielen Algorithmen, welche untereinander Daten austauschen. Torch3 liefert viele Algorithmen mit einer gemeinsamen Datenbasis und ermöglicht so das schnelle Austauschen von Subalgorithmen um so das Lernverhalten optimal auf die Aufgabenstellung abzustimmen.

GPS: Die GPS Daten werden vom GPS–Dämon [10] Version 1.14 aufbereitet, und dem NAVI System zur Verfügung gestellt. Der GPS–Dämon wird als einzelne Software nicht mehr weiterentwickelt, ist jedoch als Teilprojekt des Programms GPSDrive [11] verfügbar.

GPS Daten können auf eine Vielzahl von Arten empfangen und aufbereitet werden. Der gpsd liefert uns eine einheitliche Schnittstelle zur Abfrage dieser Daten. Des weiteren ermöglicht er das Generieren von Testläufen indem er gespeicherte GPS–Daten zur Verfügung [6] stellen kann.

Sprachsynthese: Die kleine Version des Paketes Festival, Festival–Lite [5] ist speziell für den Einsatz mit wenig Ressourcen gedacht und wurde bei uns in der Version 1.2 verwendet.

FLite ist nach unserem Wissensstand derzeit die einzige freie Sprachsynthesebibliothek [7], welche auch auf einem Handheld lauffähig ist (compilierte Größe 175954 Byte).

NAVI besteht aus mehreren Services. Jeder Service besteht aus einem oder auch mehreren Objekten. Zur Kommunikation nutzen wir das DWARF –Framework, welches wiederum selbst auf CORBA basiert. Der Datenaustausch erfolgt mit Hilfe von Events.

Aufgrund der Problemstellung ergaben sich folgende NAVI–Komponenten (Services):

GPSPosition verschickt die aktuelle Position des Benutzers.

StreetMap enthält die Datenbasis und erledigt die Routenplanung.

Filter erstellt Benutzerprofil bezüglich der Umgebungsdaten.

JoyPadInput nimmt die Benutzereingabe entgegen.

TactileOutput gibt Richtungsänderungen an den Benutzer mittels Vibration aus.

SpeechOut gibt Informationen mittels Sprachsynthese an den Benutzer weiter.

[6] Die Entwicklung GPS basierter Software ist recht mühsam, wenn man für jeden Test einen Feldversuch starten muss.

[7] Einschränkung bei der Bibliothek ist leider derzeit die Qualität. Des weiteren ist die Sprachausgabe aktuell nur auf englische Sprache abgestimmt.

Serviceübersicht

Die in 7.2 aufgelisteten Services sind über das DWARF–Framework miteinander verschalten (siehe Abb. 7.4). Jeder Service hat dabei seine spezielle Teilaufgabe zu erledigen und wird rein anhand seiner Needs und Abilities ausgewählt, dadurch ist es möglich nicht nur Services durch andere Komponenten auszutauschen, sondern auch Needs und Abilities in andere Komponenten aufzuteilen bzw. umzuverteilen (siehe 7.1).

Wie in Abbildung 7.4 dargestellt ist, benutzt NAVI eine Reihe von Datenstrukturen zur Kommunikation:

string stellt eine Abfolge von ASCII-Zeichen dar. Die Semantik, in der der Datenstrom zu sehen ist, wird über das Semantik–Attribut angegeben:

- **Speech:** Der String enthält Daten in englischer Sprache [8], welche für Sprachausgabe geeignet sind.
- **Vibration:** Die Daten stellen eine einfache Meta-Sprache dar, die in Vibrationsausgabe umgesetzt werden kann (siehe Kap. 7.3.4).

InputDataString wird bei NAVI für die Übermittlung von Benutzereingaben über das Joypad verwendet. Der in dieser Datenstruktur enthaltene String entspricht dabei den gedrückten Knöpfen auf dem Eingabegerät.

GlobalPosition besteht aus Breiten und Längengraden der aktuellen Position, welche mit Hilfe von GPS ausgewertet wurden. Des weiteren werden die aktuelle Richtungsangabe, Bewegungsgeschwindigkeit, Höhe und das Kartendatum mit übertragen.

NaviMeta ist eine NAVI-spezifische Datenstruktur, welche GPS-Daten und Umgebungsinformationen in sich vereint.

Ablauf

Das zeitliche Verhalten der einzelnen Komponenten, welche untereinander Events austauschen, wird hier auf zwei Abbildungen verteilt dargestellt.

1. Abbildung 7.5(a) zeigt den zeitlichen Verlauf bei Verwendung des Menüs, welches vom StreetMap–Service dem Benutzer als Interaktionsmöglichkeit zur Verfügung gestellt wird.

   Jeder Zyklus beginnt damit, dass der Benutzer eine Eingabe über das Steuerkreuz am Joypad tätigt. Der StreetMap–Service wertet die Eingabe aus und verschickt ein entsprechendes Event als Feedback über den SpeechOut–Service (Sprachausgabe) an den Benutzer. Dieser kann sich anhand der Rückgabe im Benutzermenü orientieren.

[8] Diese Einschränkung besteht derzeit hauptsächlich, weil die Bibliothek FLite [5] nur englische Sprachausgabe unterstützt.

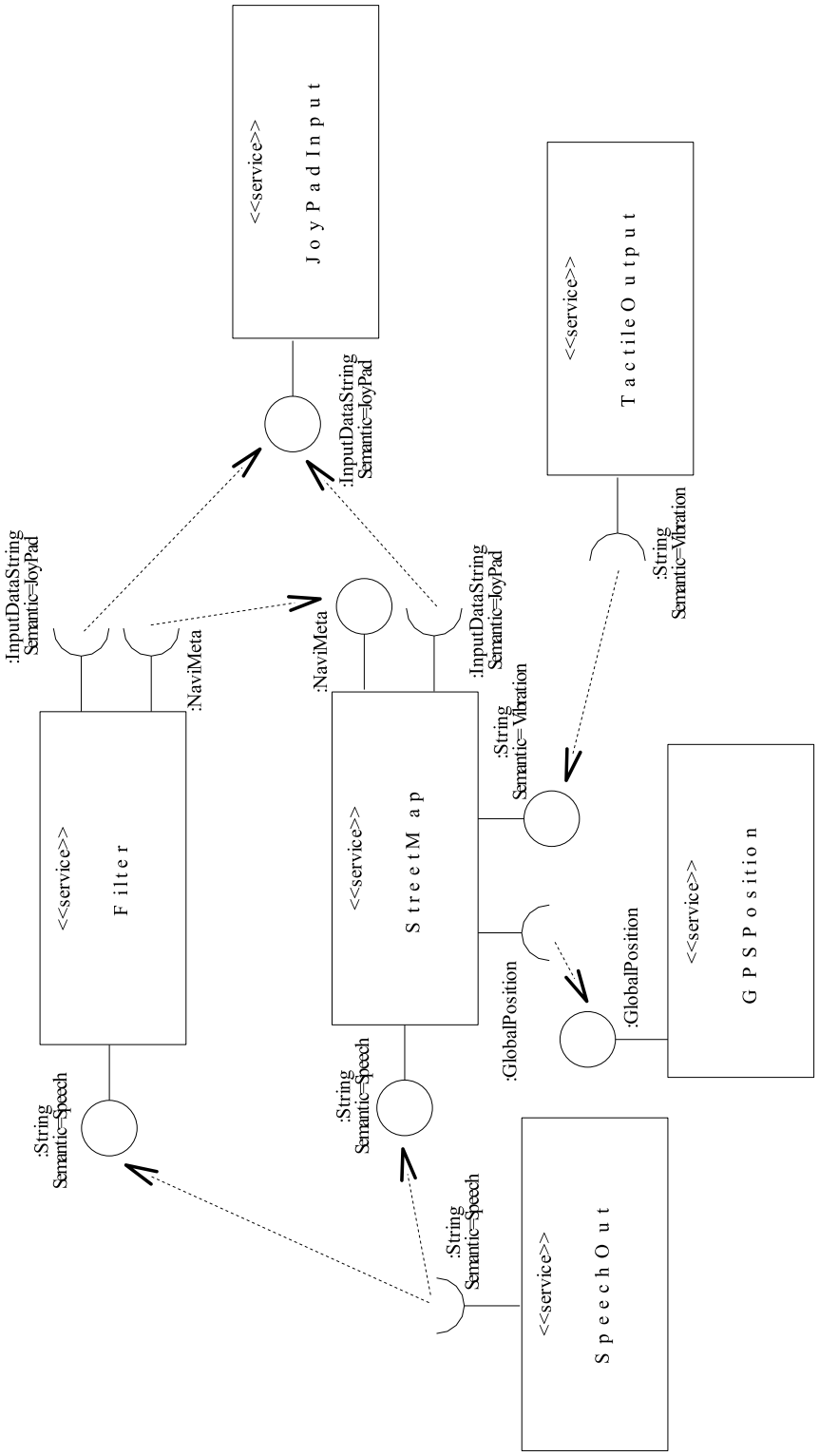
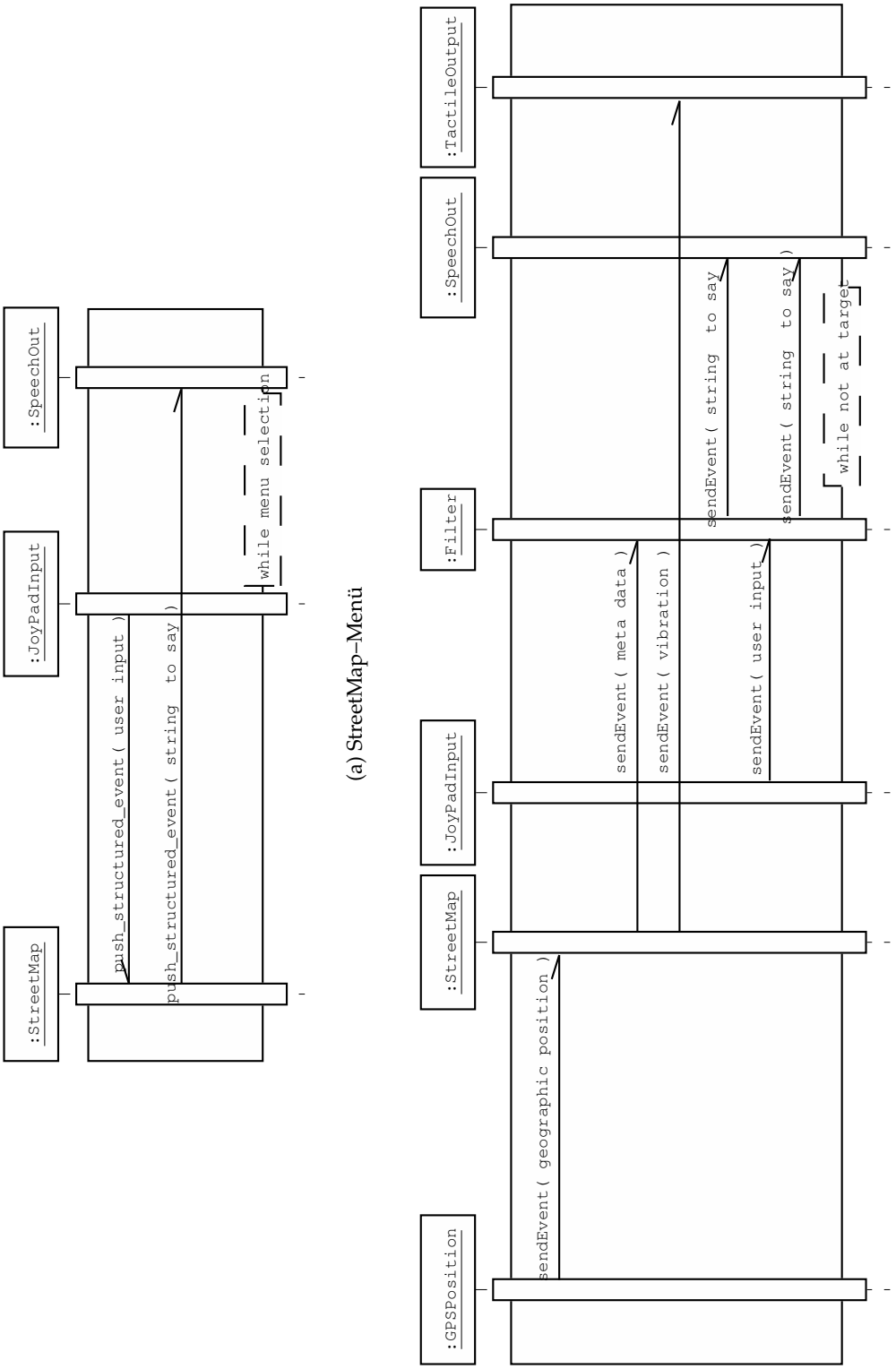


Abbildung 7.4: UML Serviceübersicht



(a) StreetMap–Menü

(b) Navigation

Abbildung 7.5: UML Ablaufdiagramme

2. Abbildung 7.5(b) spiegelt den Ablauf während der Navigation wieder. Hier beginnt der Ablauf mit einer Positionsmeldung des GPSPosition–Service, welche den aktuellen Standort und die Bewegungsrichtung des Benutzers beinhaltet. Der StreetMap–Service ergänzt bei Bedarf diese Information um die Umgebungsdaten (NaviMeta Struktur) und verschickt diese an den Filter–Service. Wenn eine Richtungsänderung dem Benutzer bekannt gegeben werden soll, wird ein entsprechendes Event an den TactileOutput–Service geschickt. Der Filter leitet anhand seines internen Entscheidungsprozesses Umgebungsinformation an den SpeechOut–Service weiter und erwartet Feedback vom Benutzer. Falls der Benutzer sich entschließt entsprechende Rückmeldung an den Filter–Service zu geben, wird die erfolgreiche Verarbeitung dem Benutzer über den SpeechOut–Service bekannt gegeben.

7.3 Implementierung der Services

Das NAVI–System besteht aus mehreren DWARF–Services (Abb. 7.6, S. 49), die untereinander über CORBA Events kommunizieren.

Die Services wurden ausschließlich in C++ geschrieben. Dadurch ist keine Laufzeitumgebung (wie z.B. bei JAVA) nötig, das bei der Ausführung, v.a. auf kleinen, leistungsschwachen Geräten, einen Vorteil bringt.

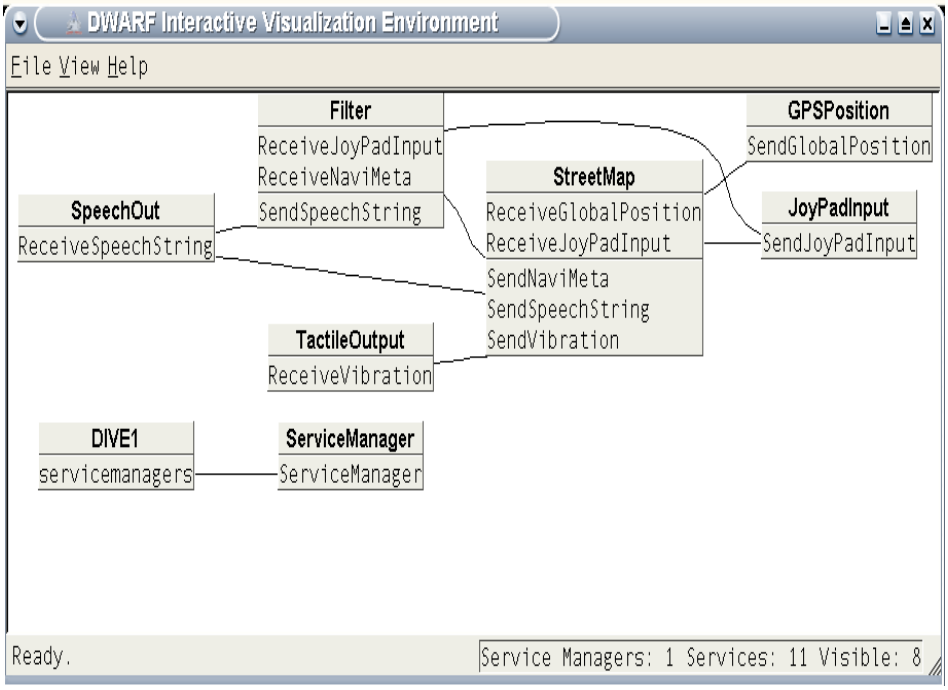


Abbildung 7.6: NAVI Services mit Verschaltung

7.3.1 StreetMap

Parameter für Kommandozeile:

> **`-Dmapfile`** Gibt die Datenbank für die Straßenkarte (siehe A.1) an.
> *default:* zwingend

Der StreetMap Service ist der zentrale Kern von NAVI. Wird der Service gestartet, so muss eine Straßenkarte angegeben werden in der navigiert wird.

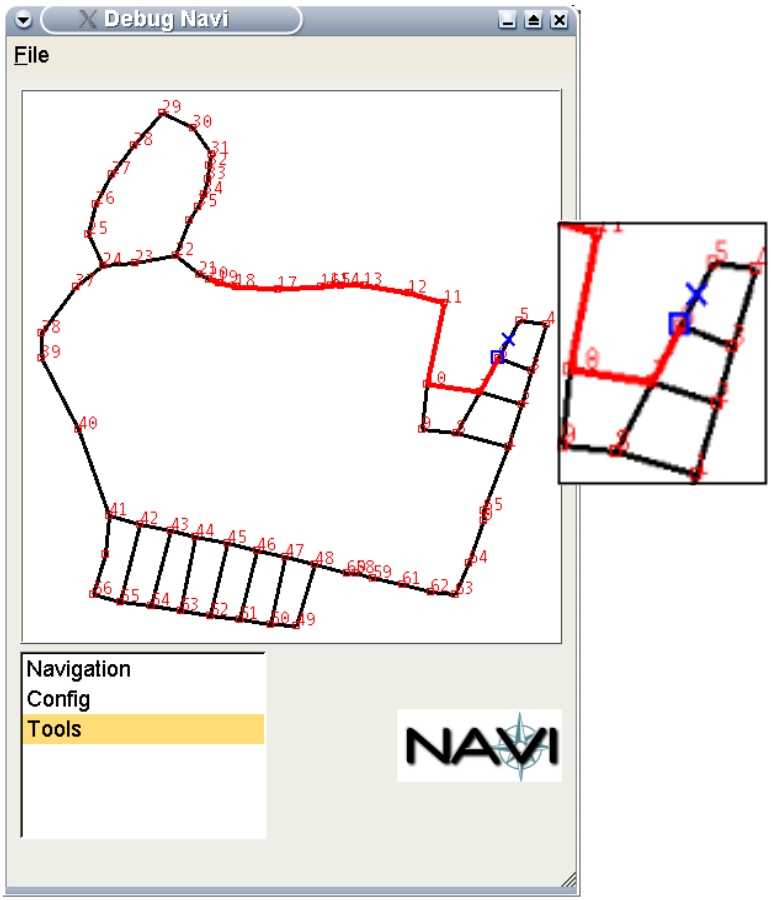


Abbildung 7.7: NAVI Debug Fenster

Über den GPSPosition Service (siehe Kap. 7.3.2) erhält der StreetMap Service die aktuellen GPS–Daten (siehe Kap. 7.3.2). Als Debug–Ausgabe wird nach dem Start ein Fenster (Abb. 7.7, S. 50) angezeigt, in dem man die geladene Karte sehen kann und durch ein blaues "x" die aktuelle Position dargestellt wird. Durch ein blaues Quadrat wird der dem Benutzer nahste Kreuzungspunkt dargestellt.

Unter der Karte befindet sich eine Liste. Über das Joypad (Abb. 7.2, S. 41) kann auf dem Steuerkreuz durch "auf" und "ab" ein Eintrag ausgewählt werden und durch "rechts" wird

der Eintrag bestätigt. Wenn unter dem Listenelement eine weitere Liste liegt, so wird diese geladen, ansonsten wird die dem Eintrag zugewiesene Funktion ausgeführt. Wird dem Joypad–Steuerkreuz "links" gedrückt, so kommt man in das vorherige Menü.

- **Navigation:** Unter diesem Punkt liegt das Untermenü für die Navigation.
  - **set target:** Dieser Eintrag führt zur Zielauswahl.
  - **orientate:** Hier werden Orientierungsinformationen in der Form "you are *200* meter southwest from *ort*" ausgegeben.
  - **get distance:** Die Distanz zum ausgewählten Ziel wird ausgegeben.
- **Config:** Dieser Eintrag führt zum Untermenü für die Konfiguration.
  - **listmode full:** Wählt man diesen Modus aus, so werden alle verfügbaren Ziele in der Zielauswahlliste dargestellt.
  - **listmode partial:** Wird dieser Modus ausgewählt, so entspricht die Liste für die Zielauswahl einer partiellen Liste (siehe Kap. 5.1.2). Die Einträge bestehen beim ersten Durchlauf aus allen Anfangsbuchstaben der Ziele, danach aus den ersten zwei Buchstaben usw. bis das Ziel identifiziert ist.
- **Tools:** Dieser Menüpunkt führt zur Liste der Tools.
  - **time:** Hier wird die aktuelle Zeit wird ausgegeben.
  - **saying:** Aus einer Liste von Sprichwörtern wird eines ausgegeben.

Wird im StreetMap aus der Zielauswahl ein neues Ziel ausgewählt, so wird die kürzeste Route dahin mit Hilfe des Dijkstra–Algorithmus [33] berechnet und in der Debugausgabe (Abb. 7.7, S. 50) wird diese durch eine rote Linie eingezeichnet.

Klassendesign

Der StreetMap Service ist von den Klassen Navigate und TemplateService abgeleitet (Abb. 7.8, S. 52). Die **Navigate** Klasse übernimmt die Routenplanung und die Verarbeitung der GPS–Daten. Außerdem wird der aktuelle Stand der Routenplanung in dieser Klasse gespeichert. Im TemplateService, der mittlerweile Bestandteil von DWARF (siehe Kap. 7.1) ist, wird die CORBA Verarbeitung erledigt. Die StreetMap Klasse implementiert vor allem die Funktionen "createAbilityObject" zum Erzeugen der Abilities, "createNeedObject" zum Erzeugen der Needs und "run", welche die eigentliche Funktionalität darstellt.

Die run–Methode besteht dabei aus folgendem Ablauf:

1. Die aktuelle Position wird gesetzt
2. Bei Bedarf werden Taktile Signale ausgegeben
3. NaviMeta Daten (siehe A.1.1) werden, wenn vorhanden, geschickt
4. Die Debug Ausgabe wird aktualisiert

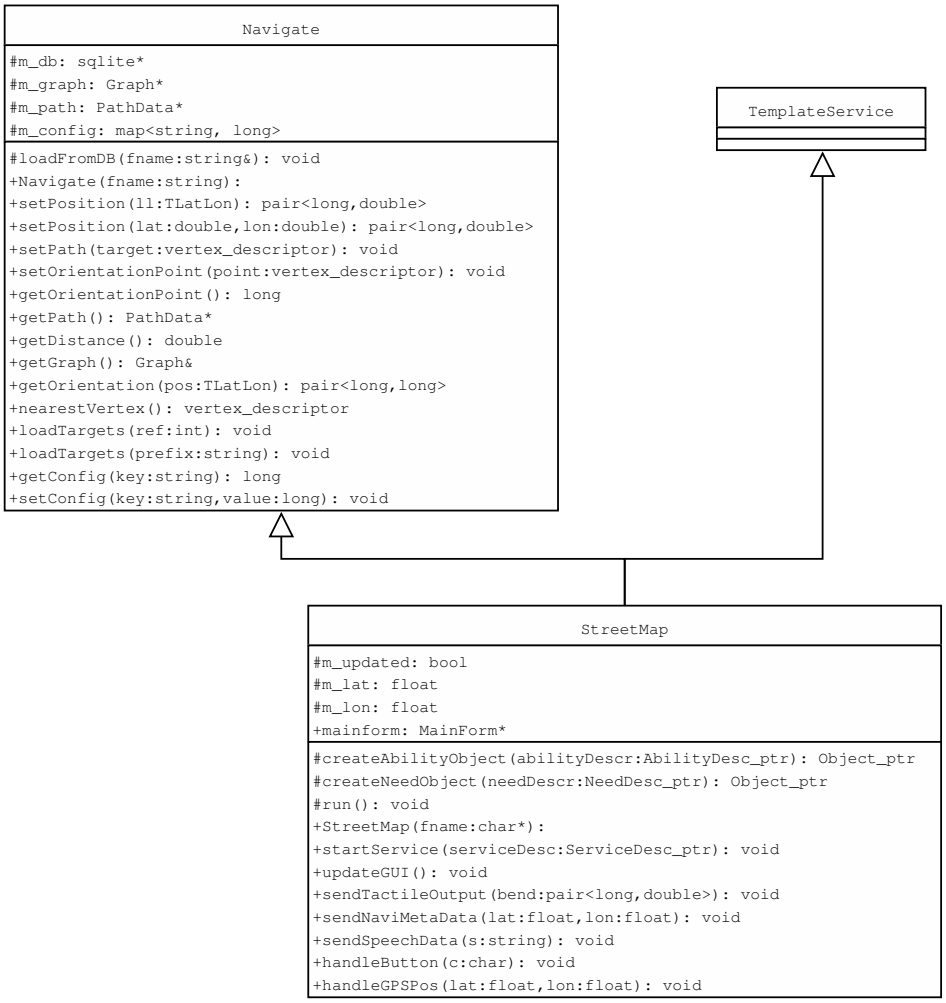


Abbildung 7.8: StreetMap Klassenhierarchie

Needs und Abilities

Der StreetMap Service hat mehrere Needs und Abilities (Abb. 7.9, S. 53):

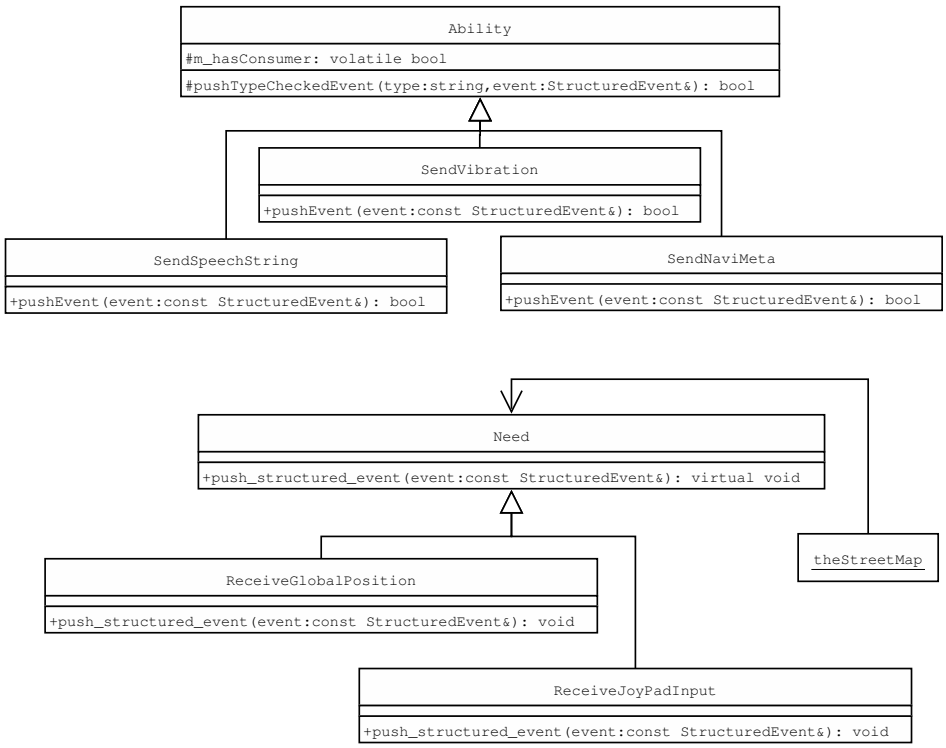


Abbildung 7.9: StreetMap Needs und Abilities

- ReceiveJoyPadInput gibt die Events vom Joypad weiter an den StreetMap.
- ReceiveGlobalPosition setzt die aktuelle Position.
- SendNaviMeta schickt die Daten für die Umgebung an den Filter.
- SendSpeechString schickt Text für die akustische Ausgabe.
- SendVibration schickt Richtungsinformationen an die Tabs.

7.3.2 GPSPosition

Parameter für Kommandozeile:

`-Dhost` Gibt die IP–Adresse des Rechners an, auf dem der gpsd läuft.
default: 127.0.0.1 (der lokale Rechner)

`-Dport` Gibt den Port an, auf dem der gpsd läuft.
default: 666786

Der GPSPosition Service verbindet sich über eine TCP/IP Verbindung zu dem gpsd. Über diese Verbindung wird fortlaufend die aktuelle Position des Benutzers ausgelesen und als Event verschickt. Bei Navi ist der StreetMap Service (siehe Kap. 7.3.1) der alleinige Empfänger der GPS–Position.

Klassendesign

Der GPSPosition Service ist von den Klassen GPSData und TemplateService abgeleitet (Abb. 7.10, S. 54). In der **GPSData** Klasse wird die Verbindung zum gpsd verwaltet und die Kommunikation mit diesem erledigt. Die GPSPosition Klasse implementiert die "createAbilityObject" Funktion, die die Ability zum Versenden der GPS–Position erstellt.

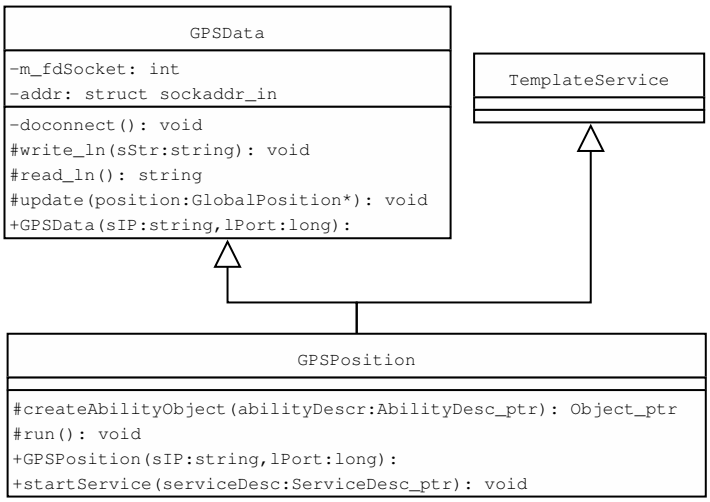


Abbildung 7.10: GPSPosition Klassenhierarchie

Needs und Abilities

Der GPSPosition Service macht nichts anderes, als die aktuelle Position vom gpsd einzulesen und diese dann als Event weiterzuschicken. Dieses Verschicken wird in der Ability SendGlobalPosition (Abb. 7.11, S. 54) ausgeführt.

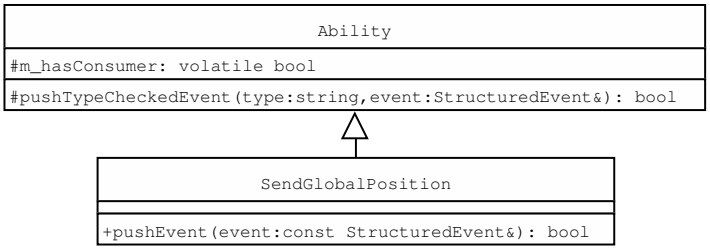


Abbildung 7.11: GPSPosition Ability

gpsd

gpsd [10] steht für GPS–Dämon. Dieser Dienst liest wahlweise von der COM–Schnittstelle, wahlweise aus einer Datei NMEA–0183 Daten ein. Diese Daten werden zwischengespeichert. Über ein Socket kann man sich zum gpsd hinverbinden und einen Anfragestring (Tab. 7.1) abschicken. Als Antwort erhält man dann eine durch Komma getrennte Liste von Werten, die der Anfrage entsprechen. Der gpsd könnte über eine bestehende Internetverbindung zusätzlich noch DGPS–Korrektursignale verarbeiten.

Zur ursprünglichen Funktionalität des gpsd habe ich bei Navi noch die Funktion "h"[9], für die Magnetische Richtung, dazu programmiert. Dadurch bestand der Anfrage–String aus den Zeichen "pamsvh".

Zeichen	Bedeutung	Einheit	Datentyp
P	Lat/Lon Koordinaten	Grad	2 Double Werte
A	Höhe	Meter	Double
M	Mode	0=kein GPS, 1=GPS, 2=DGPS	Integer
S	Status	Status des GPS–Empfängers	Integer
V	Geschwindigkeit	Knoten	Double
H	Richtung	Grad	Double

Tabelle 7.1: Befehlssatz des von uns erweiterten gpsd

GPSPosition Datenstruktur

Die GPS–Daten vom gpsd werden im GPSPosition Service in eine CORBA–Event Datenstruktur gepackt. Diese Datenstruktur setzt sich aus mehreren Bestandteilen zusammen (Tab. 7.2). Einer der Bestandteile ist das Kartendatum [10]. Bei uns war dieses fix auf "WGS84" gesetzt.

Feld	Datentyp	Bedeutung
lat	Double	Latitudinal - Koordinate.
lon	Double	Longitudinal - Koordinate.
altitude	Double	Meereshöhe in Metern.
speed	Double	Navigationsgeschwindigkeit in Metern.
status	GPSstatus	Der Status des GPS Empfängers.
mode	GPSmode	Der Modus: kein GPS, GPS, DGPS.
heading	Double	Die Navigationsrichtung.
date	string	Das verwendete Kartendatum.

Tabelle 7.2: Datenstruktur des CORBA–Events

[9] h für heading

[10] Das Kartendatum beschreibt die Transformation durch die die 3–dimensionale Erde in eine 2-dimensionalen Karte abgebildet wird [30].

7.3.3 JoyPadInput

Parameter für Kommandozeile:

`-Ddevice` Gibt das Device an, an dem das Joypad verbunden ist.
default: /dev/js0

`-Dconfig` Gibt die Konfigurationsdatei des Joypads an (siehe A.4)
default: ./jswconfig

`-DuseKB` Wenn "true", dann wird die Benutzereingabe von der Kommandozeile eingelesen.
default: false

Dieser Service nimmt die Benutzereingabe entgegen und verschickt diese. Dabei wird jedesmal der gesamte Eingabe–Status verschickt, d.h. sind mehrere Tasten gleichzeitig gedrückt, stehen alle diese in dem Eingabe–String. Durch einen Algorithmus (siehe Kap. 7.3.3) wird erkannt, ob mehrere Tasten gedrückt sind wodurch verhindert wird, dass nur Teile davon versendet werden. Ohne diesen Algorithmus könnte es passieren, dass z.B. beim Drücken der Tasten "1", "3", "4" zuerst der String "13" und anschließend "134" verschickt wird. Ebenso würden beim Loslassen der Tasten Probleme auftauchen.

Bei unserem Prototypen ist über die USB–Schnittstelle ein Joypad mit 8 Tasten und einem Steuerkreuz angeschlossen (Abb. 7.2, S. 41). Zur Zeit ist der NAVI JoyPadInput Service auf dieses Joypad ausgelegt und kann deshalb die Richtungen "oben", "unten", "links", "rechts" und die 8 Knöpfe unterscheiden.

Klassendesign

Der JoyPadInput Service ist von der Klasse TemplateService abgeleitet (Abb. 7.12, S. 56) und implementiert dabei hauptsächlich die Funktionen "createAbilityObject" und "run". In letzterer Methode wird beim Drücken einer oder mehrerer Joypad–Tasten das entsprechende Event verschickt.

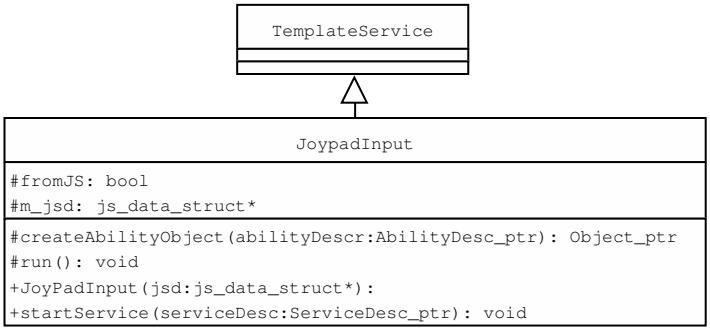


Abbildung 7.12: JoyPadInput Klassenhierarchie

Needs und Abilities

Der JoyPadInput Service verschickt die Benutzereingaben als Event. Dieses Verschicken wird in der SendJoyPadInput Ability (Abb. 7.13, S. 57) erledigt. Das dabei generierte Event ist vom Typ InputDataString, ein in DWARF (siehe Kap. 7.1) enthaltener Datentyp und besteht aus:

- eventType: InputDataChange
- stringValue: Die versendete Joypad Eingabe

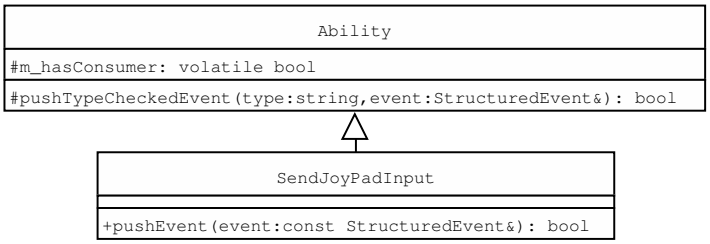


Abbildung 7.13: JoyPadInput Ability

Algorithmus für Erkennung von Tastenkombinationen

Wenn mehrere Tasten gleichzeitig gedrückt werden passiert es meistens, dass der Druckpunkt nicht bei allen gleichzeitig erreicht wird. Dadurch kann es z.B. passieren, dass beim Drücken der Kombination “1 2 4” zuerst “1 2”, danach erst “1 2 4”, anschließend beim Loslassen “2 4” und zum Schluss noch “4” erkannt wird. Um dies zu verhindern wurde folgender Algorithmus (hier in Pseudocode) benutzt:

```
while( Service läuft ) {
        sendebereit = false
        while( Tastenänderung ) {
                sendebereit = true
                tastencode = "";
                if( Kreuz gedrückt )
                        tastencode = tastencode + Kreuztaste
                if( Knopf 1 gedrückt )
                        tastencode = tastencode + '1'
                if( Knopf 2 gedrückt )
                        tastencode = tastencode + '2'
                ...
                if( Knopf 8 gedrückt )
                        tastencode = tastencode + '8'
        warte 100 ms
        }
```

```
        if( Länge von tastencode > alte Länge UND sendebereit ) {
                alte Länge = Länge von tastencode
                schicke tastencode
        }

        if( Länge von tastencode = 0 ) {
                alte Länge = 0
        }

        warte 200 ms
}
```

7.3.4 TactileOutput

Über den TactileOutput Service werden Richtungsangaben ausgegeben. Dabei wird über die Parallele Schnittstelle ein Bitmuster an die angeschlossenen Vibrations–Tabs geschickt. Solange das dem Tab entsprechende Bit gesetzt ist, vibriert dieses. Durch Ein– und Ausschalten der Bits ist somit eine Vibrationsreihe möglich.

Der Service erhält Events, die einen einfachen String enthalten. Dadurch können die Events einfach debugged und einfach generiert werden. Der erhaltene String entspricht einer Vibrationskodierung (Tab. 7.3) mit der unterschiedliche Vibrationsabfolgen erstellt werden können. So würde z.B. der Code "LR31L1R11R1" zuerst mit beiden Tabs 3 Sekunden vibrieren, danach 1 Sekunde nichts passieren, dann der Linke Tab 1 Sekunde vibrieren und zum Schluss der Rechte Tab 1 Sekunde vibrieren, 1 Sekunde nichts tun, 1 Sekunde vibrieren.

Klassendesign

Der TactileOutput Service ist von der Klasse TemplateService abgeleitet (Abb. 7.14, S. 59) und implementiert dabei hauptsächlich die Funktionen "createNeedObject", "setVibration" zum Aktivieren der Tabs und "vibrate" zum Verarbeiten des Vibrationcodes.

Needs und Abilities

Die Need–Klasse "ReceiveVibration"(Abb. 7.15, S. 59) erhält einen Vibrationscode in einem Event verpackt, extrahiert diesen und übergibt ihn an die TactileOutput Klasse.

R	Setze Status des rechten Tabs auf "ein"
L	Setze Status des linken Tabs auf "ein"
1 . . . 9	Vibriere n Sekunden

Tabelle 7.3: TactileOutput: Vibrationskodierung

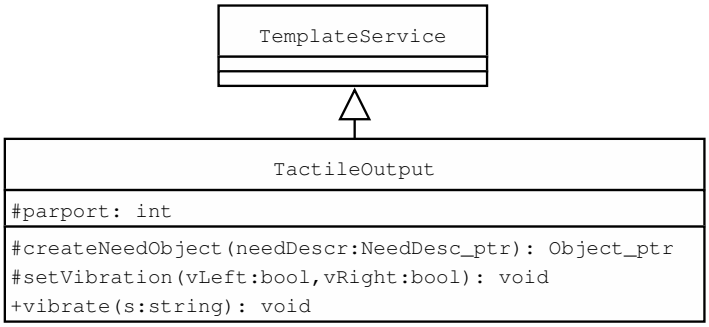


Abbildung 7.14: TactileOutput Klassenhierarchie

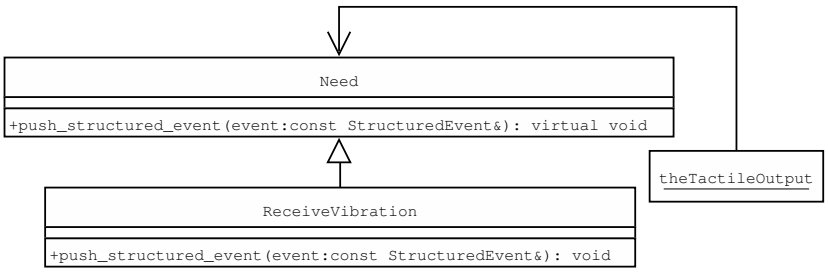


Abbildung 7.15: TactileOutput Need

7.3.5 Filter

Der Filter Service bekommt vom StreetMap Service aktuelle Umgebungsinformationen in einer NaviMeta (siehe A.1.1) Struktur übergeben. Anhand seines gelernten Benutzerprofils wird entschieden, ob die Information für den Benutzer interessant ist oder nicht. Könnte sie interessant sein, so wird sie ausgegeben. Der Benutzer kann anschließend entscheiden ob er die Nachricht hören wollte oder nicht und entsprechendes Feedback geben.

Genauere Informationen findet man in [44].

Needs und Abilities

Der Filter Service hat mehrere Needs und Abilities (Abb. 7.16, S. 60):

- ReceiveJoyPadInput gibt die Events vom Joypad weiter an den Filter.
- ReceiveNaviMeta bekommt Umgebungsinformationen.
- SendSpeechString bei Interesse werden die Daten ausgegeben.

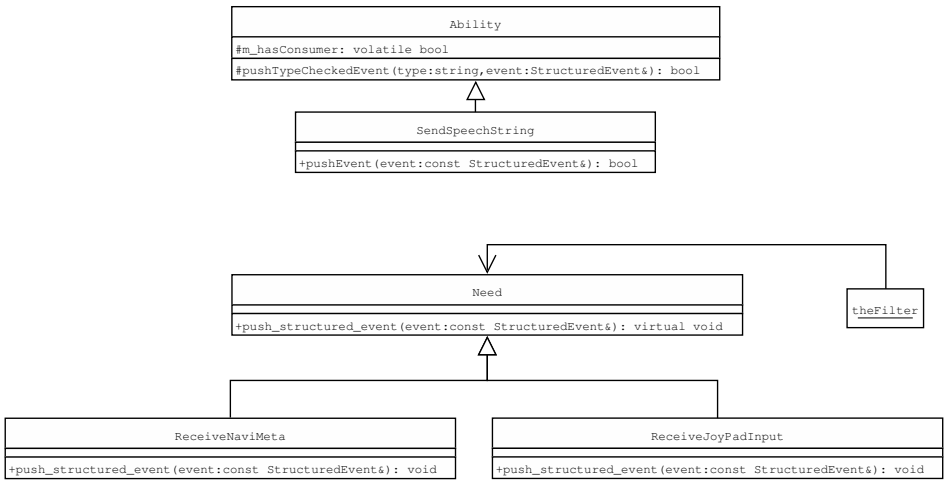


Abbildung 7.16: Filter Needs und Abilities

7.4 NAVI Use Cases

Der NAVI Prototyp erfüllt die meisten Grundfunktionalitäten eines Navigationssystems das von blinden und sehbehinderten Menschen bedient werden kann. Dabei können folgende Use Cases auftreten und bewältigt werden [11].

Szenario: **Starten des Systems**

Akteur: `Stevie:Benutzer`

Ereignisfluss:

1. Stevie trägt auf seinem Rücken einen Laptop, auf dem DWARF (siehe Kap. 7.1) mit einem gestarteten Servicemanager läuft und an dem die NAVI Ein– und Ausgabegeräte angeschlossen sind. Außerdem sind alle Services startbereit und richtig konfiguriert. In der linken Hand hält er das Joypad für die Eingabe, in seinen Hosentaschen sitzen die Vibrationstabs und er hat einen Kopfhörer auf, durch den er die Sprachausgabe hören kann. Der GPS Empfänger bekommt ausreichend Satellitensignale und hat die aktuelle Position ermittelt. In der rechten Hand hat Stevie einen weißen Stock, mit dem er den Weg vor sich abtastet um Hindernissen aus dem Weg gehen zu können.
2. Wenn Stevie die NAVI–Services startet, meldet sich die Vibrationsausgabe dass sie richtig funktioniert und das System heißt ihn mit *"welcome, enjoy your navigation"* [12] willkommen.
3. Stevie weiß, dass in der Datenbank, die er gerade geladen hat, nur eine begrenzte Anzahl an Zielpunkten vorhanden sind. Deshalb will er das System so konfigurieren, dass bei der Zielauswahl die komplette Liste der Ziele angezeigt wird. Mit dem Steuerkreuz gibt er ein:

   unten *"Navigation"* wird er später behandeln..

[11] **fett** bedeutet im folgenden eine Eingaben mit Hilfe des Joypads
[12] *"Texte in Anführungszeichen"* stellen hier die die Sprachausgabe von NAVI dar.

unten *"Config"* hat er gesucht.

rechts Nach *"entered Config"* ist er im Konfigurationsmenü.

oben Er geht die Liste von unten nach oben durch, wobei ihn der Eintrag *"listmode partial"* nicht interessiert.

oben *"listmode full"* ist für ihn interessant.

rechts Er setzt den Listenmodus um: *"set listmode full"*.

links Er hat seine Konfiguration erledigt, *"entered Main"*.

Szenario: **Zielauswahl**

Akteur: `Stevie:Benutzer`

Ereignisfluss: 1. Stevie will zu seinem gewünschten Ziel *"ZIELE"*. Deswegen führt er folgende Eingaben auf dem Joypadkreuz aus um dies dem System mitzuteilen:

unten Das Wort *"Navigation"* wird ausgesprochen und er weiß, dass er sich auf dem Navigations–Eintrag befindet.

rechts Er öffnet das Navigation–Menü und weiß durch *"entered Navigation"* dass es funktioniert hat.

unten Der Menüeintrag *"set target"* wird ausgesprochen.

rechts Er weiß durch *"entered set target"*, dass er nun eine Liste von möglichen Zielen vor sich hat.

unten Das Ziel *"ZIEL A"* interessiert ihn nicht, also sucht er weiter.

unten Das Ziel *"ZIEL B"* auch nicht

unten ...

unten Er ist am *"ZIEL E"* angekommen. Dorthin will er.

rechts *"set target ZIEL E"* sagt ihm, dass er nun sein gewünschtes Ziel markiert und aktiviert hat.

2. Stevie hat sein Ziel gesetzt (in der Debug Ausgabe (Abb. 7.7, S. 50) zeigt eine eingezeichnete rote Linie den berechneten Weg an).

Szenario: **Navigation Start**

Akteur: `Stevie:Benutzer`

Ereignisfluss: 1. Stevie geht los. Er hält in der Linken das NAVI Joypad und in der Rechten den weißen Stock, da er weiß, dass die einzige Eingabe, die er während des Gehens eventuell machen muss, das Feedback bei Umgebungsinformationen sein wird.

Szenario: **Kreuzung**

Akteur: `Stevie:Benutzer`

Ereignisfluss:
1. Stevie kommt an eine Kreuzung (in der Debug Ausgabe nähert sich das blaue Kreuz, das Stevies Position darstellt, dem blauen Kästchen, das den Kreuzungspunkt anzeigt).
2. An der Kreuzung muss Stevie rechts abbiegen und wenn er ein paar Meter von der Kreuzung entfernt ist, teilt ihm NAVI dies durch eine kurze Vibration am rechten Vibrator mit.

Szenario: **Falsche Richtung**

Akteur: `Stevie:Benutzer`

Ereignisfluss:
1. Stevie ist nicht abgebogen, weil er der Meinung war, dass es gerade aus besser ist.
2. Der Weg führt in eine Sackgasse, weshalb eine Neuberechnung über diesen Weg ihn in die andere Richtung leitet. Dazu vibrieren die Motoren auf beiden Seiten von Stevies Hosentaschen zwei mal.

Szenario: **Navigationsinformationen**

Akteur: `Stevie:Benutzer`

Ereignisfluss:
1. Stevie hat sich anscheinend geirrt. Um sich zu orientieren, wo er genau ist, fragt er Navi:

   links Er weiß nicht mehr, in welchem Menü er sich befindet. Durch *"at main"* weiß er nun, dass er sich im Hauptmenü befindet.

   rechts *"select item first"* sagt ihm, dass er noch kein Element ausgewählt hat.

   unten *"Navigation"* Menü. Das will er haben.

   rechts *"entered Navigation"*

   unten *"set target"* interessiert ihn nicht.

   unten *"orientate"* - genau das hat er gesucht.

   rechts Die Information *"you are 200 meter southwest from St. Antonius church"* sagt ihm, wo in der Umgebung er sich gerade befindet.

   unten Weil er schon dabei ist, will er auch gleich wissen, wie weit es noch ist: *"get distance"*

   rechts Die Ausgabe *"250 meters left"* teilt diese Information mit.

Szenario: **Navigation Umgebungsdaten**

Akteur: `Stevie:Benutzer`

Ereignisfluss:
1. Stevie geht nun zurück. Als er an der Kreuzung ankommt, vibriert diesmal der linke Vibrator. Gleichzeitig hört er das typische Knacken an der Fußgängerampel und weiß, dass dort gerade grün ist. Er überquert die Straße und führt seinen Weg fort.

2. Während er die Straße so entlang geht, meldet sich NAVI um ihm mitzuteilen dass er an einem H&M Geschäft vorbei geht. Stevie ist nicht interessiert und drückt deswegen den Nein–Knopf (**Joypad Taste 6**).
3. Das nächste Geschäft ist ein C&A Kaufhaus. Da Stevie aber schon bei anderen Navigationen mit NAVI dem System mitgeteilt hat, dass ihn Kleidergeschäfte nicht wirklich interessieren und das vorhin nochmal verstärkt hat, ist diesmal NAVI still.
4. Wieder einige Meter weiter kommt er an einem Sandwich–Stand vorbei. Da Stevie für Sandwiches sterben würde, gibt er auf die von Navi ausgegebenen Information diesbezüglich schnell ein Ja mit dem Ja–Knopf (**Joypad Taste 8**) ein.

Szenario: **Tools**

Akteur: `Stevie:Benutzer`

Ereignisfluss:

1. Stevie steht vor dem Sandwich Geschäft und will sich etwas kaufen. Er weiß aber, dass er auf Diät ist und gerade vor seinem Start um 12:00 Uhr schon etwas gegessen hat.
2. Nun will er sehen, wie spät es ist, weil sein Gefühl sagt ihm, dass das Essen schon einige Zeit zurück liegt. In NAVI erreicht er das durch die Steuerkreuzeingaben:

   oben Im Menüpunkt *"Tools"* gibt es diese Informationen.

   rechts Mit *"entered Tools"* ist er in dem gewünschten Menü.

   unten *"time"* hat er gesucht.

   rechts Die aktuelle Zeit: *"it's 12:35"*

   unten Stevie ist enttäuscht. Sein Mittagessen liegt noch nicht lange zurück. Er überlässt die Entscheidung einem Sprichwort: *"saying"*

   rechts *"You tell them, Stevie!"*. Was für ein Zufall, dass NAVI genau das Sprichwort ausgesucht hat.

   Stevie ist zufrieden. Er kauft sich ein Sandwich und steckt es in seine Tasche um es später, am Ziel angekommen, zu verspeisen.

Szenario: **Navigation - Zielpunkt, Wiederholung der Information**

Akteur: `Stevie:Benutzer`

Ereignisfluss:

1. Stevie ist mit neuem Elan weiter gegangen.
2. Nach 2 Kreuzungen und einer kurzen Geraden nähert er sich seinem Zielpunkt. Als er nur noch wenige Meter vom Ziel entfernt ist, gratuliert ihm NAVI: *"Congratulations you are at your target"*.
3. Stevie hatte sich gerade auf den Lärm neben ihm konzentriert und hat nicht verstanden, was NAVI ihm versucht mitzuteilen, also will er den letzten gesprochenen Satz nochmal hören. Dazu drückt er auf den Wiederholen–Knopf (**Joypad Taste 1**).

4. Er ist an seinem Ziel angekommen, und schaltet nun das System aus.

8 Ausblick

8.1 Erfolge - Misserfolge

Der Großteil der Ergebnisse sind im NAVI Prototypen vereint. Von unserer **anfänglichen Problemstellung** ausgehend (siehe Kap. 2) wurden viele Teilbereiche bearbeitet und Probleme gelöst. In der Testumgebung kann der Benutzer bei guten Bedingungen Navi sicherlich als Navigationshilfe einsetzen, darf sich aber wie bei allen bisherigen Systemen nicht nur allein auf das elektronische Navigationssystem verlassen.

Der **Routenplaner** von Navi berechnet die kürzeste Strecke zum ausgewählten Zielpunkt und führt anschließend den Benutzer dahin. Anfangs wurde auch der im GPS–Empfänger eingebaute **Kompass** miteingeplant, doch dieser erwies sich als zu ungenau und durch das Design des Routenplaners konnte ich die Richtungen beim Abbiegen auch aus der Straßenkarte berechnen. Der Nachteil dieser Methode ist allerdings, dass der Benutzer immer genau auf der Straße bleiben muss und sich nur auf in der Karte eingezeichneten Wegen bewegen darf. Existierende Navigationssysteme sind ebenfalls dafür ausgerichtet, dass sich der Benutzer auf genau vordefinierten Pfaden, in den meisten Fällen eben Verkehrsstraßen, bewegt. Ein Fußgänger kann sich hingegen frei bewegen, was komplett neue Probleme aufwirft.

Bei der Abschlussdemo ist es häufig passiert, dass einige Kreuzungspunkte nicht erkannt wurden, das System hat also nicht mitbekommen, dass der Benutzer an einem Wegpunkt angelangt bzw. vorbeigegangen ist. Dadurch haben auch die **Vibrationstabs** in diesem Augenblick kein Signal von sich gegeben. Ein Grund dafür war die **Ungenauigkeit von GPS**. In der Debug–Ausgabe konnte man sehen, wo sich der Navi–Benutzer zur Zeit befindet, wobei in einigen Abschnitten der Teststrecke die angezeigte Position von der realen um einige Meter abgewichen ist. Andererseits hat die taktile Ausgabe einige male erstaunlich gut funktioniert. Es kam teilweise vor, dass der richtige Tab kurz vor der Abbiegung vibriert hat. In diesem Fall konnte man kurz nach der Vibration abbiegen. Mit den derzeitigen Vibratoren erhält man das beste Ergebnis, wenn man sie in der Hand hält. Wenn sie in die Hosentaschen gesteckt werde und eng am Körper anliegen, kann man noch eine nutzbare Vibration spüren.

Das **Joypad** hat sich als gute Entscheidung erwiesen, da es vor allem von der Handhabung her gut bedienbar ist und bei Bedarf auch mit einer Hand gesteuert werden kann. Die Tasten sind für die Bedienung mit einer Hand etwas ungünstig belegt, weil bei der Abschlusspräsentation die Eingabe für zwei Hände optimiert war. Die Navigation in der **Liste** funktioniert schon ganz gut. Man weiß zu jedem Zeitpunkt, wo man sich gerade befindet. Vor allem der Menübaum kann nach einiger Übung schnell durchgegangen werden.

Mit Hilfe der **Sprachausgabe** werden einige Beispiele von Navigationsinformationen ausgegeben. Auch bei der Teststrecke konnte man sich dadurch einen Überblick über die aktuelle Position und den noch zurückzulegenden Weg verschaffen.

Funktionen, wie die Ausgabe der aktuellen Straße auf der man sich gerade befindet und die Erinnerung an Objekte wie Briefkästen usw., wurden einer zukünftigen Weiterentwicklung überlassen. Ebenso wird davon ausgegangen, dass die Straßenkarte bereits auf das Navigationssystem gespielt wurde. Der Client–Server Teil des heimischen PCs, der für die Administration der Karten zuständig ist, würde für sich ein eigenes Forschungsgebiet darstellen.

8.2 Grundlagen für die Zukunft

NAVI hat sicherlich einiges an Potential, dessen Ausschöpfung aber noch eine Menge Arbeit bedeutet. Die Methode mit taktilen Signalen Richtungsänderungen auszugeben wird noch bei keinem GPS–basierten Navigationsgerät benutzt. Das mag vor allem daran liegen, dass die Verlässlichkeit von GPS einfach zu gering ist. Sollte irgendwann mal ein neues Lokalisierungssystem funktionsbereit sein, könnte sich das natürlich ändern. Was aber mit der aktuellen Genauigkeit von GPS schon möglich ist, ist die Warnung vor Hindernissen. Wenn das taktile Warnsignal (in unserem Fall würden beide Vibratoren gleichzeitig vibrieren) nur als Hinweis auf mehr Vorsicht gedeutet wird, so wäre das sicherlich schon eine große Unterstützung, auch wenn die Warnung oftmals zu früh ausgegeben wird. Durch die Kapselung von GPS in einen DWARF–Service ist es NAVI relativ gleichgültig woher die Positionsinformationen kommen. Wenn ein anderer Service die gleichen Datenstrukturen liefert wie der GPSPosition Service, so kann dieser dank der Modularität von DWARF ohne Aufwand einfach durch den genaueren ersetzt werden. Kommende Technologien wie UMTS und GALILEO sind dabei sehr vielversprechend.

Die optimalen Route für den Weg zum Ziel wurde aus der Länge der Kanten berechnet. Da beim Berechnungsalgorithmus das Ergebnis aufgrund der Kantengewichtung (bei uns eben die Länge der Straßen) ermittelt wird, könnten Strecken mit Hindernissen bzw. Bus- und U–Bahnverbindungen einfach in das bestehende System einmodelliert werden (das Gewicht einer Straße wäre dabei eben um einen Hindernisfaktor höher bzw. eine U–Bahnstrecke um einen Verkehrsmittelfaktor kleiner).

Unter dieser Annahme könnte dann auch die taktile Richtungsausgabe im Freien ähnlich verlässlich wie das schon in Kapitel 3.2.2 beschriebene System für den Indoorbereich funktionieren. Wie die dortige Studie gezeigt hat, ist es prinzipiell möglich Navigationsrichtungen durch Vibration auszugeben, doch gerade im Outdoorbereich stellen sich neue Fragen wie “wann” muss bei einer Kreuzung die Ausgabe geschehen, was macht man bei komplexeren Kreuzungen, “wie” muss man die Ausgabe gestalten, damit der Benutzer auch richtig reagiert usw. Um all diese Fragen zu klären bedarf es noch einiger weitreichender Studien. Außerdem muss die Hardware noch um einiges kleiner werden. Da zukünftige PDAs sicherlich mit Bluetooth für die Funkübertragung besitzen werden, könnte die Vibrationsausgabe komplett ohne Kabel funktionieren. Wie diese Ausgabegeräte dann am besten angebracht werden, müsste in einer Ergonomiestudie erörtert werden.

Eine der größeren aber notwendigsten Weiterentwicklungen ist die Portierung auf ein kleineres Gerät, da die aktuelle Version mit einem Laptop eindeutig zu groß ist. NAVI ist so designed, dass es ohne größere Schwierigkeiten auf einen Handheld portierbar ist [1].

[1] Natürlich stellt sich der wirkliche Aufwand erst dann heraus, wenn man es wirklich macht, doch ich habe von

Die gewonnen Erkenntnisse bilden eine Grundlage für weitere Entwicklungen, sei es aufgrund positiver wie auch negativer Erfahrungen. Wenn man bedenkt, dass ein Autofahrer, der während der Fahrt seinen Blick auf die Straße richten muss, für das Navigationssystem wie ein "Blinder Benutzer" ist, so eröffnen sich plötzlich viele neue Anforderungen wie auch Möglichkeiten.

Anfang an darauf geachtet diese Möglichkeit immer im Hinterkopf zu haben

Abbildungsverzeichnis

A APPENDIX

A.1 NAVI SQLite Datenbank

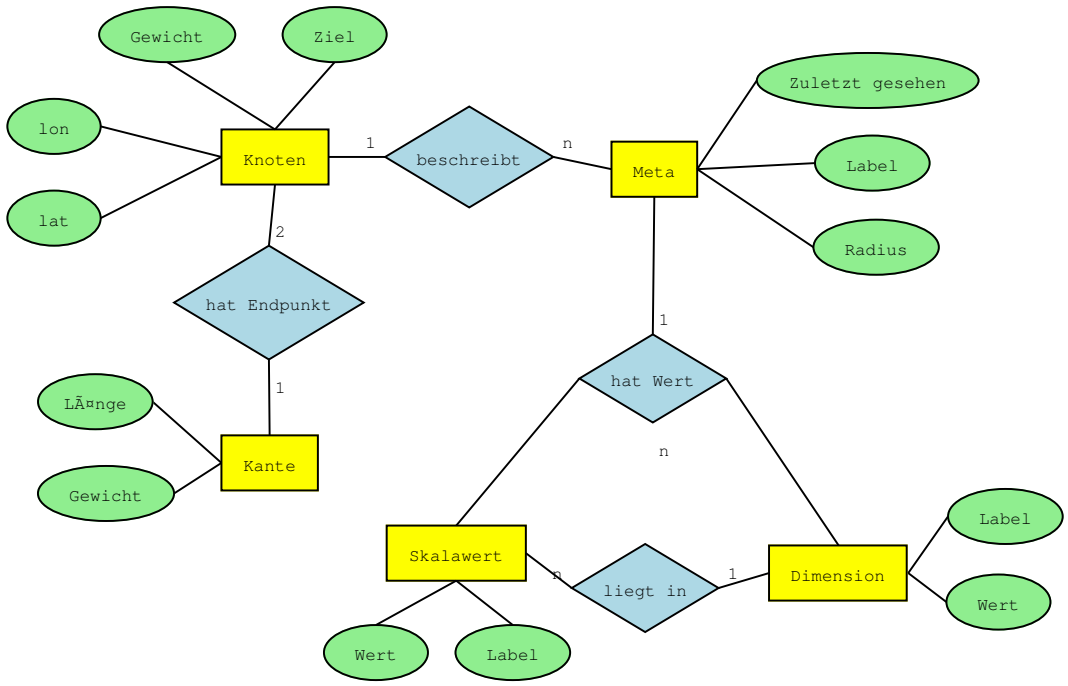


Abbildung A.1: ER–Diagramm der bei NAVI verwendeten Datenbank

Die Datenbank, in der die Straßenkarte mit den Meta–Informationen gespeichert ist, entspricht dem ER–Diagramm aus Abbildung A.1.

A.1.1 NAVI Meta Datenstruktur

Über die NAVI–Meta Datenstruktur werden die Umgebungsdaten vom StreetMap Service an den Filter Service weitergegeben. Folgende Felder werden übergeben:

units Enthält mehrere Filterdaten der Form:

- **dimension** Die ID der Dimension.
- **value** Der Wert für diese Dimension.

meta Enthält die Metainformationen.

meta_ref Die ID des Datenbankobjektes (zur Zeit nur zum debuggen).
lat Die lat–Koordinate des Datenbankobjektes (zur Zeit nur zum debuggen).
lon Die lon–Koordinate des Datenbankobjektes (zur Zeit nur zum debuggen).
radius Den Radius, in dem die Daten gültig sind. (zur Zeit nur zum debuggen).
label Der Text, der ausgegeben wird.
distance Die Distanz zum Objekt.
angle Die Richtung in der das Objekt liegt.

A.2 Tabs

Die Vibrationstabs (Abb. A.2, S. 71) sind umgebaute Handyvibratoren [1] aus denen die interne Schaltung durch Kabel ersetzt wurde. Am Ende der Kabel wurde ein Audiostecker angebracht, der in die Schaltbox (Abb. A.3, S. 72) gesteckt werden kann. Zur Zeit werden nur zwei Vibratoren verwendet, doch für spätere Funktionen wurden vier Anschlüsse vorgesehen. Über einen LPT Steckplatz kann ein Kabel zum Druckerport des Laptops zur Ansteuerung gesteckt werden. An einen Verbindungsstecker kann in der Box eine Batteriehalterung für vier Mignon Batterien angeschlossen werden, wodurch die Tabs mit Strom versorgt werden.

Abbildung A.2: Die bei NAVI verwendeten Tabs für die Vibrationsausgabe

A.2.1 Zugriff auf LPT

Um unter Linux auf die Parallele Schnittstelle zuzugreifen sind einige wenige Befehle notwendig:
`open ("dev/parport0", O_RDWR)` öffnet die Schnittstelle.
`ioctl (fd, PPCLAIM)` reserviert den Zugriff.
`ioctl (fd, PPWDATA, buffer)` setzt die in buffer [2] gespeicherten Bits, wodurch die Tabs zu vibrieren beginnen. Setzt man die Bits wieder auf 0, so endet die Vibration.
In der Schaltbox sind die Pins 1, 3, 5, 7 verlötet.

[1] Bei früheren Handys war noch kein Vibrationsalarm eingebaut. Diese kleinen Sticks wurden in der Tasche getragen und fingen bei einem Anruf an zu vibrieren. Dabei wurde durch die GSM–Signale bei einem Anruf die Vibration ausgelöst.
[2] buffer ist eine 2 Byte Datenstruktur, in der das Low–Byte die Bitkodierung enthält.

Abbildung A.3: Schaltbox mit Audioanschlüssen und LPT Steckplatz für die Verbindung zum Laptop. Ansicht von oben auf den eingelöteten Chip und das Batteriefach

A.3 Verarbeitung der GPS–Daten

Damit wir während der Entwicklungsphase nicht jedesmal die Teststrecke abgehen mussten, haben wir eine Beispielstrecke abgespeichert. Mit Hilfe des Programms ttylog[3], das auf dem COM–Port eingehende Daten ausgibt, konnten wir die Original NMEA–Daten in eine Datei abspeichern. Bei jeder Kreuzung wurde ein Waypoint im GPS–Empfänger abgespeichert.

Mit dem Programm **GPSMan** [12] wurden die Waypoints (Abb. A.4, S. 73) eingelesen exportiert und dann in SQL Datenbefehle konvertiert.

Anmerkung: *ttylog kann in der aktuellen Version die Baudrate 4800, in der die NMEA Daten vom Garmin Etrex übermittelt werden, nicht einlesen. Diese Funktionalität haben wir selber dazuprogrammiert und dem Autor des Programms zugeschickt, weshalb dieser Fehler in zukünftigen Versionen behoben sein sollte.*

A.4 libjsw Kalibrierung

Die libjsw Bibliothek benötigt eine Konfigurationsdatei, in der die Eigenschaften des Joypads wie Maximalwerte der Achsen gespeichert sind. Diese Datei kann mit dem Programm jscalibrator [29] (Abb. A.5, S. 74) erstellt werden.

[3] ttylog ist in den meisten Linux–Distributionen enthalten

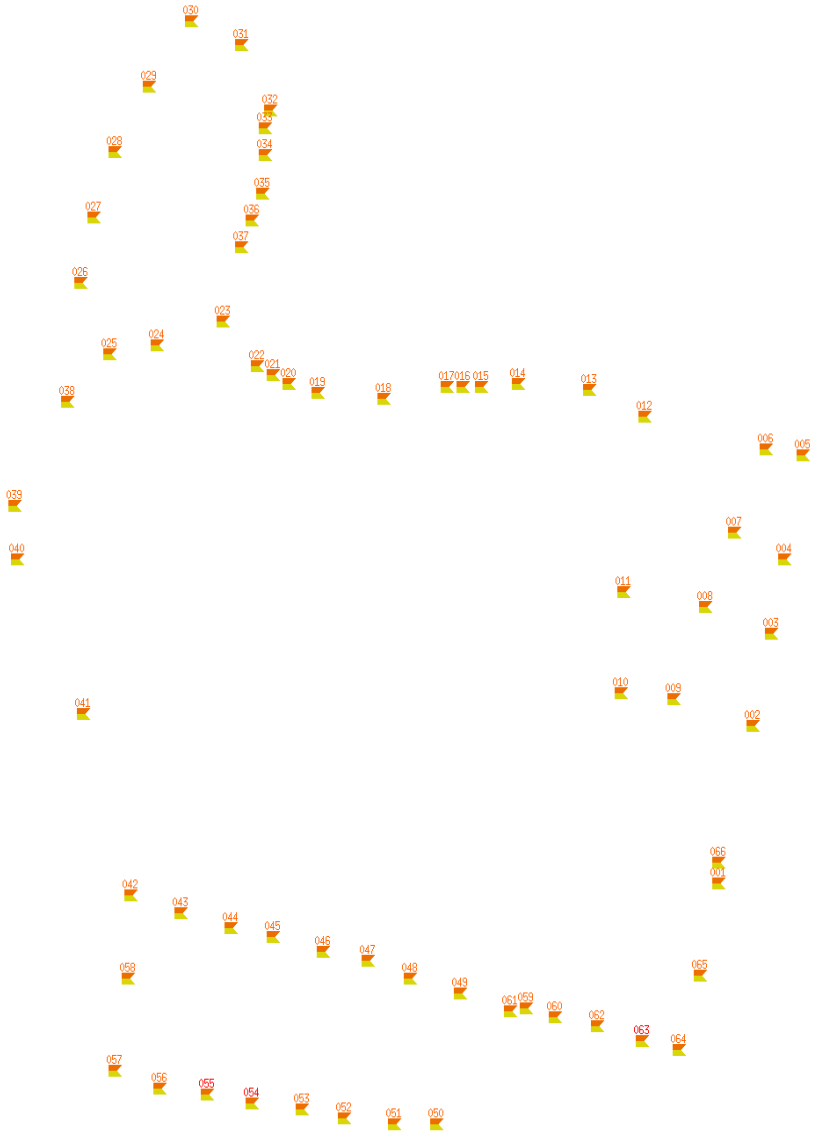


Abbildung A.4: Ausgelesene Kreuzungspunkte im Programm GPSMan (siehe A.3)

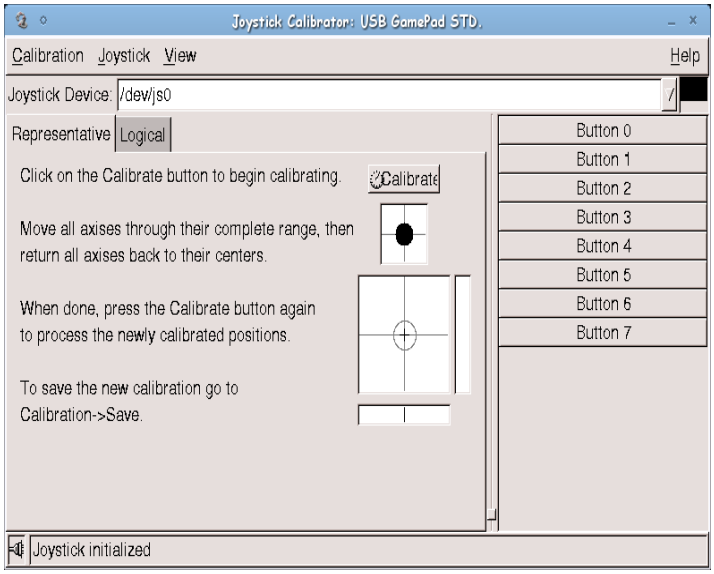


Abbildung A.5: jscalibrator, zum Erstellen der Konfigurationsdatei für das Joypad

jscalibrator Datei für das bei NAVI benutzte Joypad:

```
# Joystick calibration file.
# Generated by Joystick Calibrator version 1.5.0.
#
BeginJoystick = /dev/js0
  Name = USB GamePad STD.
  LastCalibrated = 1064837858
  BeginAxis = 0
    Minimum = 0
    Center = 128
    Maximum = 255
    NullZone = 32
    Tolorance = 0
    CorrectionLevel = 0
    DeadZoneMinimum = 128
    DeadZoneMaximum = 128
    CorrectionalCoefficientMinimum1 = 0.000000
    CorrectionalCoefficientMaximum1 = 0.000000
    CorrectionalCoefficientMinimum2 = 0.000000
    CorrectionalCoefficientMaximum2 = 0.000000
  EndAxis
  BeginAxis = 1
    Minimum = 0
    Center = 128
    Maximum = 255
```

```
    NullZone = 32
    Tolorance = 0
    Flip
    CorrectionLevel = 0
    DeadZoneMinimum = 128
    DeadZoneMaximum = 128
    CorrectionalCoefficientMinimum1 = 0.000000
    CorrectionalCoefficientMaximum1 = 0.000000
    CorrectionalCoefficientMinimum2 = 0.000000
    CorrectionalCoefficientMaximum2 = 0.000000
  EndAxis
EndJoystick
```

A.5 Linuxinstallation und Konfiguration für NAVI

NAVI wurde auf Debian Sid mit Kernel 2.6.0-test6 und auf Suse 8.1 mit Kernel 2.4.21 getestet. Auf dem Rechner muss DWARF laufen und folgende Checkliste von Softwarekomponenten wird benötigt:

libjsw Wird für die Joypad Unterstützung benötigt.

libreadline Wird bei der Alternativeingabe anstatt eines Joysticks verwendet.

libflite Für Sprachausgabe.

libsqlite Für die Datenbankunterstützung.

Qt Für die Debugausgabe von NAVI.

Damit alle Hardwarekomponenten von Navi funktionieren müssen folgende Module geladen werden und Rechte vorhanden sein:

sound Die Soundkarte muss funktionsfähig sein. Der Benutzer muss auf das Device schreiben dürfen.

joydev Device für den Joystick. Das Device muss nur lesbar sein.

parport_pc RAW–Device für den Parallelen Port. Da Taktile Ausgabe ausgegeben wird, muss der Benutzer Schreibrechte haben.

8250-pci Für die Serielle Schnittstelle an der der GPS–Empfänger angeschlossen ist. Der Benutzer muss vom Device lesen dürfen.

Literaturverzeichnis

[1] *Boost, a universal C++ Template Library*. http://www.boost.org/.

[2] *CORBA*. http://www.corba.org/.

[3] *Debian, Debian Policy Manual*. http://www.debian.org/doc/debian-policy/.

[4] *Etrex Summit Specifications*. http://www.garmin.com/products/etrexsummit/spec.html.

[5] *Flite, a small fast run time synthesis engine*. http://www.speech.cs.cmu.edu/flite/.

[6] *GALILEO, Projekt der Europäischen Union*. http://europa.eu.int/comm/dgs/energy_transport/galileo/intro/gps_de.htm.

[7] *Garmin eTrex Summit*. http://www.garmin.com/products/etrexsummit/.

[8] *GNU/Linux*. http://www.kernel.org/.

[9] *GNU/readline*. http://www.gnu.org/directory/readline.html.

[10] *GPSd*. http://wwwbruegge.in.tum.de/projects/lehrstuhl/twiki/bin/view/DWARF/ProjectNaviGPSD.

[11] *GPSDrive*. http://freshmeat.net/projects/gpsdrive/.

[12] *GPSMan, GPS Manager*. http://www.ncc.up.pt/gpsman/.

[13] *iPAQ Navigation System In Action*. http://www.teamwarrior.com/ins/action.htm.

[14] *KDE, Kommon Desktop Environment*. http://www.kde.org/.

[15] *LiveWire Traffic*. http://www.teamwarrior.com/livewire/traffic.htm.

[16] *National Marine Electronics Association*. http://www.nmea.org/.

[17] *NOPPA,Navigation and Guidance for the Blind*. http://www.vtt.fi/tuo/53/projektit/noppa/noppaeng.htm.

[18] *Object Management Group (OMG)*. http://www.omg.org/.

[19] *OpenSLP*. http://www.openslp.org/.

[20] *PAC Mate BNS*.
http://www.freedomscientific.com/fs_products/PACmate_bns.asp.

[21] *Personal Guidance System*. http://www.geog.ucsb.edu/pgs/main.htm.

[22] *QT Designer*. http://www.trolltech.com/products/qt/designer.html/.

[23] *QT, Trolltech*. www.trolltech.com/.

[24] *SQLite Database Library*. http://www.hwaci.com/sw/sqlite/
http://freshmeat.net/projects/sqlite/.

[25] *Talking Signs, Infrared Communications System*. http://www.talkingsigns.com.

[26] *Torch, machine-learning Library*. http://www.torch.ch/.

[27] *Trekker Version 1.0 User's Guide*.
http://www.visuaide.com/trekker_userguide.html.

[28] *TREKKER,A GPS system for the Blind and visually impaired*.
http://www.visuaide.com/gpssol.html.

[29] *UNIX Joystick Driver Wrapper Library*.
http://freshmeat.net/projects/libjsw/.

[30] *Was sind map datums?* http://www.kowoma.de/gps/geo/mapdatum.htm.

[31] R. BATUSEK and I. KOPEEEK, *User Interfaces for Visually Impaired People*, in Proceedings of the 5th ERCIM Workshop on 'User Interfaces for All', no. 21 in Short Papers: Assistive Technologies, ERCIM, 1999, p. 6.

[32] D. CAPOVILLA, G. HARRASSER, and G. KLINKER, *Entwicklung eines Katalogs mit Rahmenbedingungen für Orientierungs und Navigationshilfen blinder und sehbehinderter Menschen im Strassenverkehr*, 2002.

[33] E. W. DIJKSTRA, *A note on two problems in connexion with graphs.*, Numerische Mathematik, 1, 1959, pp. 269–271.

[34] S. ERTAN, C. LEE, A. WILLETS, H. TAN, and A. PENTLAND, *A Wearable Haptic Navigation Guidance System*. http://www.ece.purdue.edu/~hongtan/
Hongtan-pub/PDFfiles/C24_Ertan_wearable1998.pdf.

[35] R. G. GOLLEDGE, *Human wayfinding and cognitive maps*, in Wayfinding behavior: Cognitive mapping and other spatial processes, R. G. Golledge, ed., Johns Hopkins Press, Baltimore, MD, 1999.

[36] R. G. GOLLEDGE, J. M. LOOMIS, R. L. KLATZKY, A. FLURY, and X. L. YANG, *Designing a personal guidance system to aid navigation without sight: progress on the GIS component*, in International Journal of Geographical Information Systems, vol. 5, 1991.

[37] G. HARRASSER and F. REITMEIR, *Navigation Aid for Visually Impaired*.
http://wwwbruegge.in.tum.de/projects/lehrstuhl/twiki/bin/view/
DWARF/ProjectNavi.

[38] S. HELAL, S. MOORE, and B. RAMACHANDRAN, *Drishti,An Integrated Navigation System for Visually Impaired and Disabled*. `http://www.harris.cise.ufl.edu/projects/publications/wearableConf.pdf`.

[39] A. F. LANGE, *An Introduction to the Global Positioning System*, in Proceedings of the Thirteenth Annual ESRI User Conference, vol. 2, Palm Springs, CA, 1993.

[40] J. M. LOOMIS, R. G. GOLLEDGE, and R. KLATZKY, *Navigation System for the Blind: Auditory Display...* `http://citeseer.nj.nec.com/541514.html`.

[41] T. MAUCHER, K. MEIER, and J. SCHEMMEL, *Interactive Tactile Graphics Display*. `http://www.kip.uni-heidelberg.de/vision/public/ISSPA.pdf`.

[42] K. MIESENBERGER, J. KLAUS, and W. ZAGLER, eds., *Computers helping people with special needs: 8th International Conference, ICCHP 2002, Linz, Austria, July 15–20, 2002: proceedings*, vol. 2398 of Lecture Notes in Computer Science, New York, NY, USA, 2002, Springer-Verlag Inc.

[43] I. J. PITT and A. D. N. EDWARDS, *Improving the Usability of Speech-Based Interfaces for Blind Users*, in Second Annual ACM Conference on Assistive Technologies, Vision Impairments – II, 1996.

[44] F. REITMEIR, *Benutzerorientierter Datenfilter für Umgebungsinformationen als Erweiterung eines Navigationssystems für sehbehinderte und blinde Mitmenschen*, 2003.

[45] J. SEILER, *Ein Garmin eTrex-Datenkabel selbstgebaut*. `http://www.jens-seiler.de/etrex/datenkabel.html`.

[46] T. STROTHOTTE, H. PETRIE, V. JOHNSON, and L. REICHERT, *MoBIC user needs and preliminary design for a mobility aid for blind and elderly travellers*, 1995.

[47] T. STROTHOTTE, H. PETRIE, V. JOHNSON, and L. REICHERT, *MoBIC: user needs and preliminary design for a mobility aid for blind and elderly travellers*, in Proceedings of 2nd TIDE Congress (Paris, La Villette, 26–28 April 1995), 1995.

[48] VARIOUS, *Proceedings of the IEEE and ACM International Symposium on Augmented Reality - ISAR 2001*, 2001.