Universität Zürich
Institut für Informatik

Dynamic and Distributed
Information Systems

Indoornavigation unterstützt durch Magnetfeldsensorik

Diplomarbeit 11. Dezember 2006

Daniel Suter
von Auw AG, Schweiz

Matrikel Nr: 02-701-522
daniel@suter.bz

Betreuer: **Peter Vorburger**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.unizh.ch/ddis

Danksagung

Ich möchte mich bei all jenen herzlich bedanken, die das Zustandekommen dieser Diplomarbeit ermöglicht haben. Grosser Dank gebührt Prof. Dr. Abraham Bernstein und Peter Vorburger welche mir die Möglichkeit gaben mich mit dieser interessanten Thematik auseinander zu setzen. Peter Vorburger danke ich ganz besonders für das hervorragende Coaching. Während der gesamten Arbeit unterstützte er mich tatkräftig mit seinem Wissen und seiner Zeit. Die vielen Gespräche motivierten mich sehr und die unzähligen Ideen, die er einbrachte, waren von grosser Hilfe.

Ich danke ebenfalls allen Personen, die mir das Studium an der Universität Zürich ermöglicht haben. Namentlich meine Eltern Monika und Anton sind mir immer mit Rat und Tat zur Seite gestanden und haben damit einen ganz wesentlichen Beitrag für meine berufliche und private Laufbahn geleistet.

Zusammenfassung

Bei der Orientierung in unbekannten Umgebungen verlässt sich der Mensch seit Jahrtausenden auf die Kunst der Navigation. In den letzten 20 Jahren haben computerunterstützte Positionierungssysteme diese zusätzlich erleichtert. Viele Benutzer bekunden jedoch Mühe, Karten zur Orientierung in Einklang mit der realen Umwelt zu bringen. Die vorliegende Arbeit geht dieses Problem mit der Einbettung von Kompassfunktionalität in ein elektronisches Navigationssystem an. Ausgehend von einem mobilen Gerät (PDA) beschreibt diese Arbeit den Prozess von der physischen Anbindung bis hin zur Visualisierung und Einbettung einer Magnetfeldsensorik. Das Resultat ist eine funktionsfähige Navigationssoftware, welche durch ein Experiment unter Realbedingungen erfolgreich getestet wurde. Die vorliegende Arbeit dient als Basis für weitere Forschung im Bereich der Navigationsunterstützung.

Abstract

While moving in unknown environments mankind has been relying for thousands of years on the art of navigation. In the last 20 years the emergence of computer-aided positioning systems have facilitated this additionally. However, many users state difficulties to harmonize maps with the real world. The available thesis addresses this problem with embedding compass functionality into an electronic navigation system. On the basis of a mobile device (PDA) this work describes the process of the physical interfacing up to the visualisation of magnetometer data. The result is a functioning navigation software, which was successfully tested by means of a field experiment. The available work serves as basis for further research within the area of navigation support.

Inhaltsverzeichnis

1

Einleitung

Das englische Wort „Navigation“ ist synonym mit den Deutschen Begriffen „Schiffahrt“ , „Nautik“ oder „Schiffsführung“. Denn bereits 4000 v. Chr wurden in Indien und Ägypten Navigationstechniken für die Seefahrt entwickelt. Dabei spielte die sogenannte Koppelnavigation eine entscheidende Rolle, nämlich das Navigieren unter Berücksichtigung von Geschwindigkeit, Zeit und Kurs. Zu dieser Zeit wurde der Kurs über Astronavigation bestimmt. Der von daher stammende Begriff der Navigation bedeutet im engeren Sinne „die Kunst ein Schiff von einem Abgangsort zu einem Bestimmungsort zu führen“. Heutezutage kann man diesen Begriff ein wenig weiter fassen, und versteht darunter die „Steuermannskunst“ zu Wasser, zu Lande und in der Luft. Allgemein geht es darum, sich in einem geographischen Raum zu orientieren um ein besimmtes Ziel zu erreichen.

Einer der Kernpunkte bei der Navigation ist die Richtungsbestimmung. Von den Anfängen der Seefahrt über das Altertum, bis hin zum frühen Mittelalter wurden die Gestirne als Richtungsweiser benutzt. In Homer’s Odysseus werden verschiedenste Sternbilder zur Navigation erwähnt und auch in der Bibel, so zum Beispiel im Buch Hiob (Hiob 9.9 und 38.31), werden die Sterne als Navigationshilfe gebraucht. Gewisse Quellen lassen vermuten, dass die Chinesen bereits im 3. Jahrhundert v. Chr. kompassähnliche Systeme benutzt haben. „Im Jahr 2634. v. Chr. hatte Kaiser Hoang-ti eine Rebellion niederzuwerfen und wurde durch Nebel verhindert seine Feinde zu verfolgen. Er stellte einen Wagen auf, der die Südrichtung anzeigte, steuerte geraden Kurs durch den Nebel und vollendete seinen Sieg“. Auch wird berichtet, dass im 4. Jahrhundert n. Chr. die Chinesen mit Kompass ausgerüstete Schiffe Indische Häfen und die Ostküste Afrikas besuchen liessen. Aus europäischer Sicht ist der Kompass im Altertum unbekannt gewesen. Erst im Mittelalter ist er zur Anwendung gekommen. Laut europäischer Geschichtsschreibung findet sich die erste gesicherte Quelle über die Existenz des Kompass’ in den Schriften des Mönchs Alex-

ander Neckham (1157 - 1217). Laut ihm sollen Seefahrer bei schlechter Sicht einen Magnetstein mit einem Stück Schilfrohr auf Wasser in ein Gefäss gelegt haben. Das Schilf zeigte dann mit seinem Ende Richtung Norden. Traditionsgemäss wird jedoch Flavio Gioja aus Positano als Erfinder des Magnetkompass´ bezeichnet. Er soll diesen im Jahr 1302 erfunden haben. [Freiesleben, 1978]

Ab dem Beginn des 20. Jarhunderts hat die Entdeckung der Funkpeilung die Navigation zusätzlich erleichert und so die damaligen Möglichkeiten zur Positionsbestimmungen revolutioniert. Durch konsekutive Peilungen war es auch möglich, die Fahrtrichtung genau zu bestimmen. 1978 dann wurde vom US Verteidungsministerium der erste NAVSTAR GPS Satellit in seine Umlaufbahn gebracht und bis 1985 folgten zehn weitere. Ab 1993 erreichte das System mit 24 Satelliten die „Initial Operational Capability“ und 1995 wurde das Global Positioning System (GPS) offiziell freigegeben, auch für den zivilen Gebrauch.

Spätestens seit der Einführung des GPS, spielt Computertechnik eine entscheidende Rolle bei der Navigation. Dabei wurden die Verfahren weiterentwickelt und verfeinert.

1.1 Motivation

Eines wichtiges Instrument bei der Navigation ist die Karte. Früher waren die Navigierenden auf verschiedene Techniken angwiesen, um die Karte in Einklang mit der realen Welt zu bringen. Dazu wurden oft besondere Merkmale wie Leuchttürme, Gestirne oder später auch Instrumente wie der Kompass benutzt. Durch die computergestützte Navigation wie GPS oder andere Verfahren wurde die Positionierung auf der Karte stark vereinfacht. Das Problem, die Karte mit der realen Umwelt abzugleichen, ist trotzdem noch nicht komplett gelöst. Es ist nach wie vor der Mensch, der die Entscheidung fällen muss, wohin er sich in der realen Welt bewegen muss um ein gegebenes Ziel auf der Karte zu erreichen. Bei Autonavigationsystemen wurde diesem Problem zum Beispiel mit sprachlicher Wegweisung begegnet. Die bei der **Indoornavigation** verwendeten Geräte fordern jedoch andere Lösungen, die eine dynamische Unterstützung der Navigation ermöglichen würden. Von grossem Nutzen wäre ein dynamisches Verfahren, das die virtuelle Karte der Gehrichtung des Benutzers anpasst.

1.2 Ziel der vorliegenden Diplomarbeit

Das Ziel dieser Diplomarbeit ist, aufzuzeigen, wie durch die Verwendung eines Magnetfeldsensors dynamische Navigationstechniken möglich sind. Die Idee ist die Einbindung solcher Techniken in das bestehende MobileGame der Forschungsgruppe Informations Mangagement (IM) der Universität Zürich. Dabei soll in einer ersten Phase die Machbarkeit gezeigt werden und erst in einer zweiten Phase die Einbettung des Sensors in die bestehende Applikation erfolgen. Die dritte Phase sieht die Durchführung eines Experimentes vor, welches ebenfalls in Kooperation mit der Forschungsgruppe IM realisiert werden soll. In diesem Experiment soll untersucht werden, ob dynamische Unterstützungsansätze zur Navigation auf der Basis von Magnetfeldsensoren die herkömmlichen Ansätze übertreffen.

1.3 Anwendungsgebiete

Eine dynamische Unterstützung der Navigation könnte in einer Vielfalt von Anwendungungsgebieten zum Einsatz kommen. Wenn man an den Indoorbereich denkt, wäre es zum Beispiel möglich, dass Servicetechniker von IT-Diensleistungsunternehmen, welche bei einem Kunden einen Serverausfall beheben sollen, in schnellst möglicher Zeit, dynamisch, an den Serverstandort geführt werden. Andere Anwendungen sind im Gebiet von Mienen und Bergwerken denkbar. Generell könnten dynamische Ansätze die Orientierung optimieren und dem Benutzer gezielter die nötigen Informationen vermitteln ohne seine Aufmerksamkeit unnötig zu absorbieren.

1.4 Struktur und Sprachgebrauch der vorliegenden Arbeit

Nach der *Einführung* in das Thema und in seine verwandten Arbeiten wird die Ausgangslange der Arbeit dargelegt. Dazu wird auf die einzelnen zu *verwendenden Komponenten* näher eingegangen. Im Anschluss daran werden verschiedener Ansätze geprüft, wie eine Kompassfunktionalität mit den gegebenen Voraussetzungen realisiert werden kann. Die Erkenntnisse dieser *Machbarkeitsanalyse* fliessen in die darauf folgende *Implementation* ein. Diese beinhaltet eine Abänderung des bestehenden MobileGame Systems. Die während der Implementierung geleistete Arbeit soll anschliessend durch ein *Experiment*, bestehend aus verschiedenen Spieldurchgängen evaluiert werden. Die Arbeit wird durch eine *Schlussfolgerung* und *Ausblicke* auf zukünftige Arbeiten abgerundet.

Anglizismen

Obwohl diese Arbeit in Deutscher Sprache geschrieben ist, werden des öfteren Englische Wörter zur Anwendung kommen. Im Fachgebiet der Informatik ist das eine gebräuchliche Weise um komplexe Sachverhalte durch einen oftmals englischsprachigen Jargon zu vereinfachen. Darauf soll in dieser Arbeit nicht verzichtet werden.

Männliche und weibliche Formen

Um diese Arbeit nicht zusätzlich zu komplizieren wird davon abgesehen jeweils die männliche und die weibliche Form zu verwenden. Die männliche Form schliesst im hier verwendeten Sprachgebrauch das weibliche Pendant implizit mit ein.

1.5 Einordnung und Abgrenzung zu verwandten Arbeiten

Die vorliegende Arbeit ist in einem Feld verschiedener Themengebiete angesiedelt. Wie aus dem Titel entnommen werden kann, geht es um die Thematik der „Indoornavigation“. Konkret bedeutet dies die Navigation in Gebäuden und folglich ohne direkten Sichtkontakt mit Satelliten. Es gibt eine Viefalt von Forschungszweigen, die dabei eine Rolle spielen. Diese werden nachfolgend in zwei Kategorien unterteilt. Nämlich erstere, welche sich vor allem mit den **technischen Grundlagen** der Indoornavigation befassen. Sie gehen unter anderem der Frage nach, wie eine präzise Positionierung ohne die Hilfe von Satellitensystemen wie das GPS möglich ist. Die zweite Kategorie von Forschungszweigen befasst sich weniger mit den technischen Details der Indoornavigation, sondern mit deren vielfältigen **Anwendungsgebiete**. Dazu gehört auch das Gebiet des Mobile Learning. Für jede dieser Kategorien sollen nun einige relevante Arbeiten vorgestellt werden.

1.5.1 Technische Grundlagen der Indoornavigation

Das Problem, wie man technisch seine eigene Position, Geschwindigkeit und Lage in einem Gebäude bestimmen kann, wurde mit verschiedenen Ansätzen zu lösen versucht. Es gibt zum einen ein Forschungsgebiet, dass sich mit sogenannten Inertialen Navigationssystemen (INS) beschäftigt. Ein INS funktioniert nach dem Prinzip der Trägheitsnavigation. Es versucht zum Beispiel durch die Messung von Beschleunigung, Winkelbeschleunigung und magnetischer Strah-

lung den zurückgelegten Weg zu berechnen. Auf der anderen Seite gibt es auch den Ansatz, die eigene Position über den Gebrauch von Wireless LAN (WLAN) Signalen zu berechnen. Zu beiden Gebieten seien nachfolgend einige Arbeiten erwähnt.

INS mittels Extrapolation der Geschwindkeit und Position

In der Arbeit von [Tschanz, 2005] wird prinzipiell mit der selben Hardware gearbeitet wie sie auch von dieser Arbeit verwendet wird. Die von Sensoren gemessenen Beschleunigungs-, Magnetfeld- und Winkelbeschleunigungsdaten werden von Tschanz durch verschiedene Berechnungsmethoden zu Navigationsinformationen verarbeitet. Mit Hilfe der Winkelbeschleunigung und des Magnetfeldsensors werden zusätzlich auch Störungen des Magnetfeldsensors kombiniert. Die Errechnung der Navigationsdaten, konkret der Geschwindigkeit und der Position, erfolgt mittels Extrapolation. Die Arbeit von Tschanz hat vielversprechende Resultate geliefert. Der verwendete Ansatz der Extrapolation hatte jedoch auch mit einigen Problemen zu kämpfen. Unter anderem führte die verwendete statische Kalibrierung zu ungenauen Positionsinformationen.

INS mittels Interpolation von Ortsinformationen

Während die vorher erwähnte Arbeit den Ansatz der Extrapolation verfolgte, ging [Bachmann, 2006] neue Wege bei der Positionsbestimmung. Er entwickelte einen Ansatz, bei welchem durch eine periodische Ortsinterpolation eine dynamische Kalibrierung bestimmt werden kann. Der Vorteil dieser Methode ist, dass die Positionsangaben mit einer erhöhten Genauigkeit errechnet werden können. In seiner Arbeit hat Bachmann bewiesen, dass dieser Ansatz theoretisch realisierbar ist. Der Nachteil ist, dass die hohe Komplexität der hergeleiteten Gleichungen und der dementsprechende Rechenaufwand eine praktische Anwendung bislange verhindert hat.

Kombination von INS und GPS

Den teilweise ungenauen Angaben bei der Navigation mit INS begegneten [Ladetto et al., 1999] und [Ford et al., 2001] mit der Kombination von GPS-Positionangaben mit jenen vom INS. Der Vorteil dieses Ansatzes ist, dass das INS durch die Versorgung mit GPS-Daten periodisch neu kalibriert werden kann und so genauere Angaben liefert. Der Nachteil ist jedoch, dass bei längerem Ausbleiben des GPS Signals diese Kalibrierung nicht vorgenommen werden kann und die vom INS gelieferte Positionierung sehr ungenau werden kann.

Indoornavigation mittels WLAN Signalen

Eine weitere Methode der Positionsbestimmung ist die Verwendung von Wireless LAN (WLAN) Signalen. Durch Messungen der Signalstärke und der verwendeten Access Points kann ein Triangulierungsverfahren zur Anwendung gebracht werden, das die Position eines WLAN Benutzers bestimmen kann. Der Vorteil ist, dass dieses Verfahren bei einer vorgängig guten Kalibrierung eine relativ genaue Positionierung ermöglicht. Der Nachteil liegt darin, dass es nur in Gebieten mit WLAN Abeckung funktioniert. Dieser Ansatz der Navigation wurde in vielen Arbeiten erforscht und dokumentiert so zum Beispiel in [Schweigert et al., 2005].

1.5.2 Anwendungen im Bereich der Indoornavigation

Neben den oben erwähnten technischen Grundlagen gibt es wie erwähnt auch eine Reihe anwendungsspezifischer Themen der Indoornavigation. Ein dabei tangiertes Gebiet ist das Mobile Learning. Mobile Technologie, wie die im vorgängigen Abschnitt beschriebene, ermöglicht es Forschung in natürliche Umgebung einzubetten. So zum Beispiel **Untersuchungen des menschlichen Lernens**. Nachfolgend werden einige relevante Arbeiten zu diesem Thema vorgestellt.

Mobile Spiele

Es gibt eine Reihe von Szenarien, wo das Potenzial mobiler Spiele für das Lernen genutzt werden kann. So zum Beispiel in Museen oder auch im Tourismusbereich. [Göth et al., 2004] hat dazu einen Ansatz entwickelt, mit welchem Experimente diesbezüglich gemacht werden können: Ein Mobiles Spiel (genannt MobileGame oder mExplorer), das es erstsemestrigen Studenten ermöglichen soll, auf spielerische Art und Weise den Universitäts Campus kennenzulernen. Es wurde ein prototypisches Spiel entwickelt und eine Reihe von Experimenten durchgeführt. Die Resultate und weitere Erkenntisse wurden in [Schwabe and Göth, 2005a] und [Schwabe and Göth, 2005b] dokumentiert. Eine wichtige Entdeckung die Göth gemacht hat, war das sogenannte Fokusproblem ([Göth et al., 2006]). Der nächste Abschnitt beinhaltet mehr Informationen dazu.

Das Fokusproblem

Ein von [Göth et al., 2006] häufig beobachtetes Problem bei mobilem, technologieunterstütztem Lernen war, dass die Teilnehmer durch die Technologie zu sehr in Anspruch genommen wurden und dadurch den Fokus nicht auf das eigentlich Objekt des Lernens richteten. Die Technologie

absorbierte also die Aufmerksamkeit der Teilnehmer. Göth bringt in seiner Arbeit verschiedene mögliche Gründe für das Problem an, die man in [Göth et al., 2006] nachlesen kann.

1.5.3 Einordnung dieser Arbeit

Die Indoornavigation tangiert eine Reihe von Themengebieten. Es geht nun darum, die Position der vorliegenden Diplomarbeit hinsichtlich dieser Themenvielfalt zu bestimmen. Diese Diplomarbeit bewegt sich an der Schnittstelle zwischen technischer Umsetzbarkeit und der Frage nach einer sinnvollen Anwendung im Bereich des Mobile Learnings. Es geht nicht darum eine neue Methode für ein INS zu entwickeln. Vielmehr wird versucht, durch eine Kombination bestehender Technologien einen Prototyp zu schaffen mit welchem weitere Forschung im Bereich des mobilen Lernenes durchgeführt werden kann. Insbesondere soll es auch möglich sein, durch Experimente zu testen ob das Fokusproblem allenfalls durch dynamische Navigationsansätze reduziert werden kann. Diese Arbeit beschränkt sich dabei auf kompassartige Orientierungshilfen. Welche Technologien dabei konkret verwendet werden, behandelt das nächste Kapitel.

2

Verwendete Komponenten

Die vorliegende Arbeit basiert auf einer Ausgangslage. Diese besteht unter anderem aus den vorgegebenen Hard- und Software Komponenten. Aus diesem Grund werden in diesem Kapitel die harware- und softwaretechnischen Rahmenbedingungen der vorliegenden Diplomarbeit erläutert. Zuerst wird der verwendete PDA, namentlich der iPAQ 4700hx, vorgestellt. Anschliessend wird die verwendete Kompassfunktionalität erklärt und wie diese realisiert wird. Und zum Abschluss wird ebenfalls die zur Entwicklung verwendete Software erläutert.

2.1 iPAQ 4700hx

Das bereits bestehend MobileGame System wurde für eine Verwendung mit dem iPAQ 4700hx (siehe Abbildung 2.1) von Hewlett Packard (HP) ausgelegt. Aufgrund mehrerer schon früher durchgeführten MobileGame Experimenten verfügt das Institut für Informatik bereits über eine entsprechende Menge solcher Geräte. Die Evaluation von in Frage kommenden PDAs und der anschliessende Entscheid für dieses Gerät wurden in vorgängigen Arbeiten vorgenommen. Der interessierte Leser sei hierzu an die Arbeit von [Brunner, 2005] und [Göth, 2003] verwiesen. Der iPAQ hx4700 besteht aus einem Gehäuse aus Magnesiumlegierung, verfügt über einen 4 Zoll Touchscreen und 64MB SDRAM. Das vorinstallierte Betriebssystem ist Windows Mobile 2003 SE (Second Edition). Eines der Hauptauswahlkriterien war der verhältnismässig grosse Bildschirm kombiniert mit einer handlichen Grösse. Das ist wichtig um die Dynamik des MobileGames nicht zu sehr zu hemmen, denn bei einem zu kleinen Bildschirm muss der Benutzer unter Umständen stehenbleiben um seine aktuelle Position auf der Karte richtig erkennen zu können. Die technische Spezifikation des Gerätes kann im Anhang B.2 auf Seite 96 gefunden werden.

Abbildung 2.1: iPAQ hx4700 von [Hewlett Packard, 2006]

2.2 Sensorboard

Die in der Einführung erwähnte Kompassfunktionalität wird durch eine zusätzliche Komponente realisiert. Es handelt sich dabei um eine Reihe von Sensoren, die auf einem kleinen Brett zusammengefasst werden. Diese Komponente als Ganzes wird als Sensorboard bezeichnet und bildet das Herzstück, auf welchem diese Arbeit aufbaut. Entwickelt wurde es von Peter Vorbuger[1]. Das Gerät besteht aus einer Reihe von Teilkomponenten, die dem Benutzer verschiedene Messewerte liefern können. Zusammengefasst stehen folgende Messwerte zur Verfügung:

- Temperatur
- Beschleunigung
- Winkelbeschleunigung
- Gasmessungen
- Magnetische Feldstärkemessungen

Das Zusammenspiel der verschiedenen Teile wird durch einen Prozessor mit integrierten 12-bit Analog Digital Wandler (ADC) gesteuert. Der ADC konvertiert die vom Sensor gelieferten Spannungswerte (zw. 0 und 2.5V) in eine digitale Repräsentation, nämlich in sogenannte „Messpunkte“. Jeder Sensor liefert als Output einen Wert zwischen 0 und 4096 (= 2^{12}). Der Prozessor

[1] Weitere Informationen zu Peter Vorburger: http://www.ifi.unizh.ch/ddis/people/vorburger/

ermöglicht verschiede Outputgeschwindigkeiten. Bei dieser Arbeit wurde mit Geschwindigkeiten von 15-20 Messwerte/Sekunde gearbeitet. Diese Arbeit befasst sich in erster Linie mit dem Magnetfeldsensor des Sensorboards. Deshalb liegt der Fokus vor allem auf dieser Funktionalität. Eine gute Übersicht und detailiertere Informationen findet man in [Bachmann, 2006].

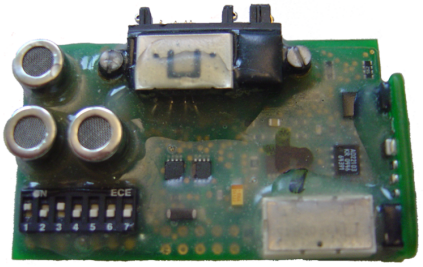

Abbildung 2.2: Sensorboard [Bachmann, 2006]

2.2.1 Format der Sensordaten

Der Sensor überträgt die digitalen Daten in Form der genannten Messpunkte über eine TTL/UART Schnittstelle. Die Daten bilden eine kommaseparierte Liste von total 12 Werten. Jeder der zwölf Werte besteht aus drei Byte ASCII und bildet einen hexadezimalen Wert. Konvertiert man diese hexadezimalen in dezimale Werte, erhält man einen Wertebereich von 0 bis 4096. Das Listing C.1 zeigt anhand eines Beispiels wie der vom Sensorboard ankommende Datenstrom aussieht.

Listing 2.1: Beispiel für Output des Sensorboard

```
FFA,820,81F,8B6,803,A4A,A44,17C,2E8,00A,006,00D
FFA,817,80B,8AD,807,A4C,A44,172,2F2,FFB,FFB,00A
FFA,820,811,8B2,804,A46,A3B,165,2F0,000,FFB,FFB
FFA,807,81C,8B3,808,A4C,A40,169,2ED,FFB,004,004
FFA,80B,81E,8B2,804,A4B,A40,16B,2F9,007,007,004
```

Wichtig ist dabei, welcher Wert mit welchem Sensor korrespondiert. Der Datenstrom ist wie folgt zuzuordnen (vergleiche Listing 2.1 von rechts nach links):

1. Temperatursensor

2. Beschleunigungssensor y-Achse
3. Beschleunigungssensor x-Achse
4. Beschleunigungssensor z-Achse
5. Winkelbeschleunigungssensor y-Achse
6. Winkelbeschleunigungssensor x-Achse
7. Winkelbeschleunigungssensor z-Achse
8. Gassensor 1
9. Gassensor 2 / 3
10. Magnetfeldsensor x-Achse
11. Magnetfeldsensor z-Achse
12. Magnetfeldsensor y-Achse

Von besonderem Interesse für diese Arbeit sind die Werte auf Position 10,11 und 12, nämlich jene des Magnetfeldsensor. Im folgenden Abschnitt wird näher darauf eingegangen.

2.2.2 Magnetfeldsensor

Auf dem Sensorboard wurde ein Magnetfeldsensor von Honeywell eingebaut (Modell HMC1053). Dieser Sensor misst die von der Erde abgegebene magnetische Strahlung. Die Strahlung variert in ihrer Richtung und Stärke je nach dem, wo auf der Erdoberfläche man sich befindet. Richtung und Stärke können als dreidimensionaler Vektor repräsentiert werden. Wenn nur die X- und Y-Werte dieses Vektors betrachtet werden, kann man auf einfache Weise die zweidimensionale Kompassrichtung bestimmen.

Kalibrierung

Der Output des Magnetfeldsensors bewegt sich für jede der drei Achsen des Koordinatensystems in einem Bereich von rund 0.5V[2]. Das entspricht 800 unterschiedlichen Messpunkten im ADC-Output[3]. Da die generelle Bandbreite des ADC-Inputs von 0 bis 2.5V reicht, liefert das Sensorboard, wie in der Einführung dieses Kapitels erwähnt einen im Bereich von 0 bis 4096 Messpunkte. Das bedeutet, dass die vom Sensorboard erhaltenen Werte vor deren Weiterverarbeitung korrigiert werden müssen. Beim Magnetfeldsensor ist es das Ziel, jeden erhaltenen Wert so zu

[2]Unter Normalbedingungen bzw. auf das Erdmagnetfeld angewendet.
[3]Ein Messpunkt entspricht 0.61035mV

korrigieren, dass ein Wertebreich von -400 bis 400 resultiert (vergleiche mit Bild 1 auf Abbildung 2.3), so dass die Werte um einen Nullpunkt verteilt sind. Dazu genügt es, wenn man vom gelieferten Sensorrohwert einen Offset subtrahiert. Die genauen Offset Werte sind vom jeweiligen Sensorboard (es wurden mehrere Exemplare hergestellt) abhängig und unterscheiden sich deshalb. Der Offset pro Achse kann relativ einfach bestimmt werden in dem der Sensor intensiv über eine bestimmte Zeit[4] in alle drei Achsenrichtungen bewegt wird und für jede Achse der gemessene Maximal- und Minimalwert festgehalten wird. Im Anschluss daran kann der Offset wie folgt berechnet werden:

$$\text{Offset} = \text{Minimalwert} + \left(\frac{\text{Maximalwert} - \text{Minimalwert}}{2} \right) \tag{2.1}$$

Von dem vom Sensorboard erhaltenen Output muss folglich immer der erhaltene Offset abgezogen werden. So erhält man einen für die weitere Verarbeitung geeigneten Sensorwert. Eine mögliche weitere Verarbeitung stellt die Berechnung der Kompassrichtung dar. Darüber gibt der nächste Abschnitt genauer Auskunft[5].

Kompassrichtung

Um eine zweidimensionale Kompassrichtung zu berechnen, genügen zwei der drei Sensorwerte, nämlich jene der X- und der Y-Achse. Wenn man den Sensor parallel zur Erdoberfläche hält und gleichförmig in einem Kreis dreht, sieht der Sensoroutput aus wie in Abbildung 2.3 (Bild 1) dargestellt. Bild 2 zeigt ein Beispiel eines realen Sensoroutputs. Die Kompassrichtung als Wert in Grad kann mit folgenden Gleichungen berechnet werden [Honeywell, 1995]:

$$y = \begin{cases} 90 - \arctan \frac{x}{y} \cdot \frac{180}{\pi} & \forall \ y > 0 \\ 270 - \arctan \frac{x}{y} \cdot \frac{180}{\pi} & \forall \ y < 0 \\ 180 & \forall \ y = 0, x < 0 \\ 0 & \forall \ y = 0, x > 0 \end{cases} \tag{2.2}$$

[4] Grössenordnung: 30 Sekunden

[5] Weitere Informationen zum Magnetfeldsensor bieten die Application Notes von Honeywell (siehe Literaturverzeichnis).

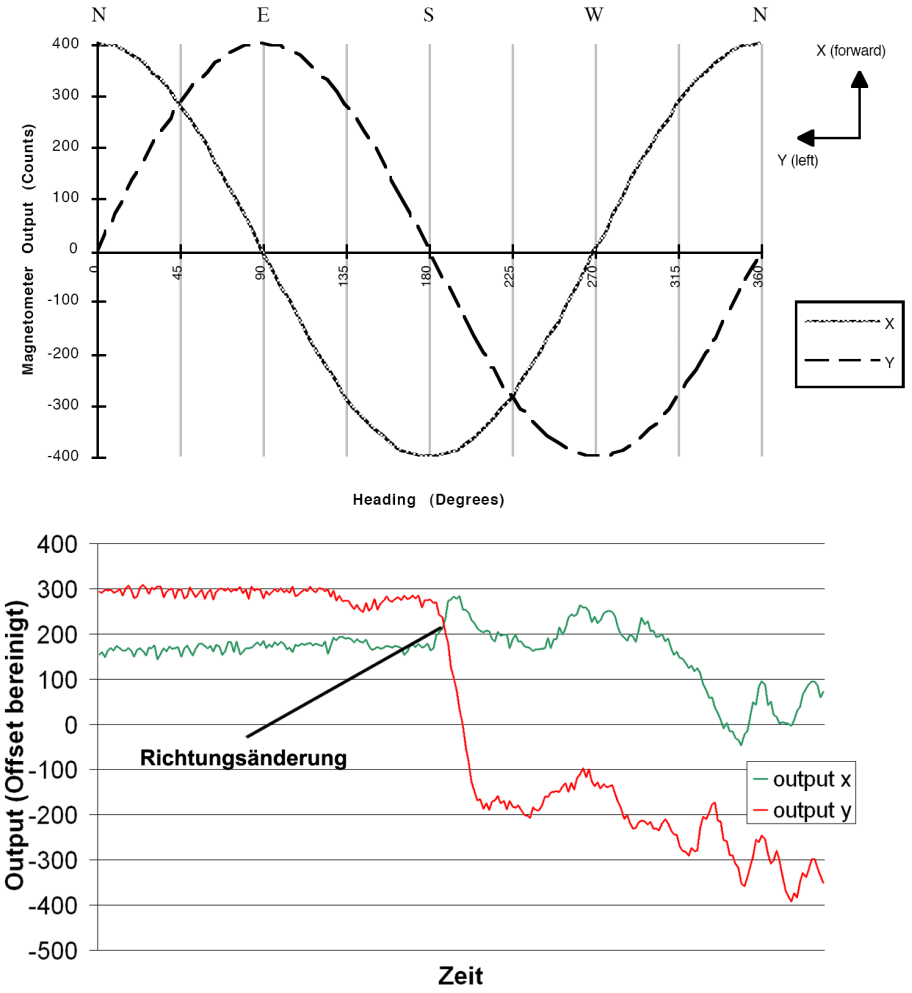

Abbildung 2.3: Kompassrichtung anhand von Magnetfeldsensordaten: Bild 1 (oben) gleichförmige Kreisbewegung des Sensors [Honeywell, 1995] Bild 2 (unten) real gemessener Sensoroutput bei Richtungsänderung (15 Sekunden)

2.3 Software

Bei der Durchführung dieser Arbeit kam eine Vielfalt von Software zum Einsatz. Dieser Abschnitt behandelt jedoch ausschliesslich die wichtigsten Teile davon. Wie in der Einführung erwähnt, sollen die hier gemachten Erkenntnisse in das bestehende MobileGame System der Forschungsgruppe Informationsmanagement der Universität Zürich eingebaut werden. Da eine komplette

Neuentwicklung der Software nicht zur Diskussion stand, war man gezwungen mit den bereits früher getroffenen Designentscheidungen weiterzuarbeiten. Was das bedeutet, wird im anschliessenden Abschnitt näher erläutert.

2.3.1 Java Konfiguration

Für Entwicklungen im Mobile Bereich stehen heutzutage eine Reihe von Architekturen zur Verfügung welche sich anbieten würden. So hat zum Beispiel [Kägi, 2002] in seiner Arbeit evaluiert, dass gerade im Zusammenhang mit der Auswertungen von Sensordaten das von Microsoft angebotene .NET Compact Framework eine hervorragende Alternative ist. Das zur Verfügung stehende MobileGame System verwendet aber einen anderen Ansatz und ist komplett in Java geschrieben. Aufgrund dieser Situation war man bei dieser Arbeit gezwungen, den selben Weg zu gehen. Wie das bereits bestehende MobileGame demonstriert, ist auch Java eine valuable Alternative zur Entwicklung verteilter Architekturen. Da, wie man später in der Arbeit noch im Detail sehen wird, das MobileGame eine Client Server Architektur aufweist, werden hier beide Aspekte kurz betrachtet.

Client

Für die zur Verfügung stehenden iPAQs, welche mit Windows Mobile 3.0 ausgerüstet sind, gibt es verschiedene Java Runtime Environments (JRE), welche in Frage kommen würden. Man hat sich bei der Entwicklung des MobileGames für das Angebot von IBM entschieden (siehe auch [Brunner, 2005]). Die damals angebrachten Gründe für diese Entscheidung waren primär die guten Eigenschaften der J9-Runtime von IBM hinsichtlich Geschwindigkeit und Integration mit der Entwicklungsumgebung. Die J9 bildet in Kombination mit dem IBM Websphere Device Developer (WSDD) eine integrierte Entwicklungsumgebung, welche dem Entwickler eine Reihe von Aufgaben erleichtert. Die Entwicklung der grafischen Benutzeroberfläche basiert auf dem Standard Widget Toolkit (SWT) von Eclipse. Auch das SWT wird durch die J9 bestens unterstützt. Das war ein weiterer Grund für diese Entscheidung. Im Detail wurden auf der Client Seite folgende Versionen verwendet:

- **JDK** J2ME Personal Profile 1.0
- **JVM** IBM J9 (5.7.2)
- **SWT** SWT Version 3.1.2 (ARM Edition)

Server

Der verwendete MobileGame Server basiert auf einer regulären Java Stanard Edition. Im vorliegenden Fall wurde JDK 5.0 in Kombination mit dem SWT 3.2 verwendet.

3

Machbarkeitsanalyse

Im Kapitel 2 wurden die technischen Rahmenbedingungen dieser Arbeit vorgestellt. Dieses Kapitel baut nun darauf auf und analysiert die Möglichkeiten mit den gegebenen Voraussetzungen die gesetzten Ziele zu erreichen. Die kritischen Punkte dabei sind:

- **Sensoranbindung:** Die Sensorplatine muss an den iPAQ angeschlossen und deren Datenstrom muss gelesen und interpretiert werden können.
- **Bildverarbeitung:** Um die Sensordaten im Sinne der Aufgabenstellung visuell aufbereiten zu können sind verschiedene Bildbearbeitungsoperationen notwendig. Die im Vergleich zu handelsüblichen PCs reduzierte Rechenleistung und Hauptspeicherkapazität des vorliegenden iPAQs erschwert eine sinnvoll performante Bild- und Grafikmanipulation zusätzlich.

3.1 Sensoranbindung

Unter dem Begriff Sensoranbindung wird im Rahmen dieser Arbeit die Anbindung des externen Sensorboardes via serieller Schnittstelle an das Gerät, sowie die Umwandlung und Interpretation des Sensordatenstroms verstanden. Dieses Problem lässt sich in verschiedene Teilprobleme gliedern. Zum einen geht es um die (1) physische Anbindung des Sensors an den iPAQ. Da die Sensoren bisher noch nie mit einem iPAQ verwendet wurden, ist hier der Schlüsselpunkt, eine geeignete Hardwareschnittstelle zu bauen. Weiter geht es auch darum, mit der gegebenen Entwicklungsumgebung eine (2) softwaretechnische Anbindung des Sensors zu implementieren und diesen anschliessend (3) mit einer sinnvollen[1] Geschwindigkeit zu lesen und zu interpretieren.

[1] Wie man später noch sehen wird, sind mit einer „sinnvollen“ Geschwindigkeit mindestens 10 Sensorwerte/Sekunde gemeint.

3.1.1 Physische Anbindung

Das im Kapitel 2 vorgestellte Sensorboard ist mit einem TTL/UART kompatiblen Ausgang ausgerüstet welcher im Rahmen vorgängiger Arbeiten, wie zum Beispiel in jener von [Bachmann, 2006], mit einem Stecker für den Palm Treo versehen wurde. Der Palm Treo kann TTL/UART kompatiblen Datenstrom direkt lesen, während der verwendete iPAQ, gemäss seiner Spezifikationen nur RS232 Datenströme zu lesen in der Lage ist. Der vorgesehene Eingang für die Schnittstelle weist zudem ein proprietäres Pin-Layout auf, welches weder jenem vom Palm Treo noch dem Standard Layout von RS232 entspricht. Es sind also zwei Schritte notwendig, um eine Datenverbindung vom Sensor zum iPAQ herzustellen:

1. Konvertierung der TTL/UART Signale in RS232 Signale (und vice versa)
2. Anschluss an einen iPAQ kompatiblen Stecker

Die erste dieser beiden Aufgaben wurde mit Hilfe eines Analog Devices Chips vom Typ ADM203[2] gelöst. Dieser nimmt eine Konvertierung von TTL/UART Signalen in RS232 Signale (und umgekehrt) vor (man siehe auch [Analog Devices, 2001]. Der zweite Schritt war, der Anschluss der Pins an einen iPAQ kompatiblen Stecker. Abbildung 3.1 auf Seite 17 zeigt den schematischen Aufbau der herkömmlichen RS232 Schnittstelle und jenen vom iPAQ verwendeten Cradle Connector.

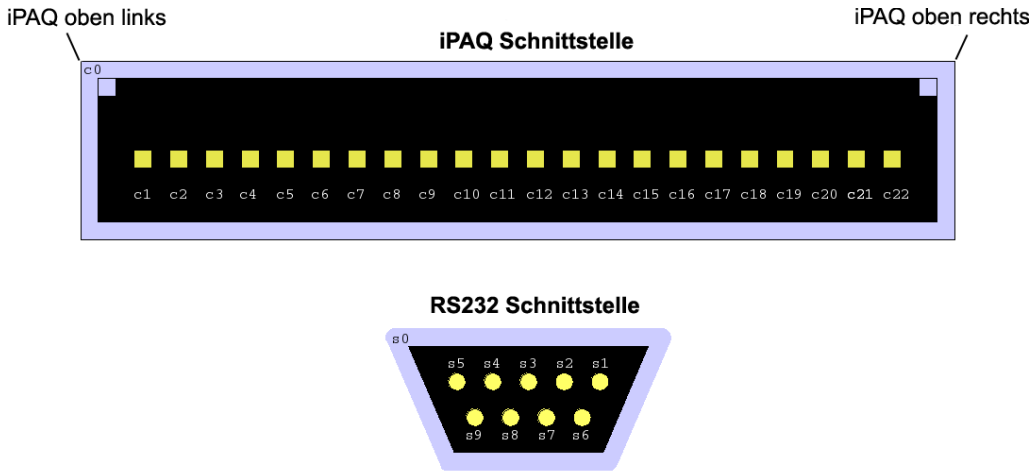

Abbildung 3.1: Die iPAQ Schnittstelle und die RS232 Schnittstelle im Vergleich (nach [Opdenacker, 2006])

[2]Der ADM203 Chip ist in der Lage die unterschiedlichen Spannungslevel von TTL und RS232 zu überbrücken. Mehr Informationen findet man unter: http://www.analog.com

Um die unterschiedlichen Layouts zu verbinden wurde folgende Pinbelegung verwendet (Tabelle 3.1):

iPAQ	RS232
S2	C15
S3	C16
S5	C13

Tabelle 3.1: Pinbelegung für einen iPAQ-RS232 Stecker

Das Resultat der beschriebenen zwei Schritte ist ein, von Peter Vorburger erstelltes, Zwischenstück. Auf der einen Seite ist dieses mit einem iPAQ kompatiblen Stecker (für RS232 Daten) versehen und auf der anderen Seite mit einen zum Sensorboard kompatiblen (Palm Treo) Stecker (für TTL/UART Daten). Der interessierte Leser findet ein detailiertes Schaltschema im Anhang B.1. Damit ist das Problem der physischen Anbindung des Sensorboards an iPAQ gelöst.

3.1.2 Softwaretechnische Anbindung

Neben der korrekten Verbindung des Sensorboards mit dem iPAQ stellt sich die Frage, ob die vorgegebene J9 Laufzeitumgebung von IBM es dem Programmierer erlaubt, direkt den seriellen Port des iPAQ anzusprechen. Denn, wie bereits angedeutet, sind J2ME Laufzeitumgebungen in deren Funktionsumfang eingeschränkt. Das Ziel war, mittels der `javax.comm` API von Sun Microsystems, welche als Erweiterung zur J2SE angeboten wird, auf den RS232 Port zugreifen zu können. Von Sun selbst wird jedoch keine Version für Windows Mobile angeboten. Eine Analyse vorhandener `javax.comm` Implementationen ergab, dass vier alternative Implementationen für Windows Mobile verfügbar sind (Tabelle 3.2 auf Seite 19). Zwei der verfügbaren Implementationen sind nur mit zeitlich beschränkte Evaluationslizenzen kostenlos erhältlich. Die von James Nord angebotene Implementation des RXTX-Treibers konnte während dieser Arbeit nie fehlerfrei implementiert werden. Deshalb wurde die `javax.comm` Implementierung von Michael Hobot verwendet, welche zurzeit noch in einer frühen Entwicklungsphase ist. Sie funktionierte jedoch bei sämtlichen Tests dieser Arbeit fehlerfrei. Die Alternative von Michael Hobot hat aber auch ihre technischen Grenzen. Das von der `javax.comm` Spezifikation vorgesehene Listener-Event-Modell konnte nicht verwendet werden. Statt dessen wird im später entworfenen Prototyp eine Schleife mit `read()` Aufrufen verwendet. Tests zeigten, dass diese Implementationsweise verlustfrei arbeitet und somit dem Zweck dieser Arbeit vollkommen genügt.

Treiber	Hersteller	Lizenz	Website
RXTX	Michael Hobot	GPL	http://mho.republika.pl/java/comm/ http://www.rxtx.org
MBS5	ProSyst	30 Tage Evaluation	http://dz.prosyst.com
CEJavaComm	James Nord	Freeware	http://www.teilo.net
SerialMagic	Serialio.com	30 Tage Evaluation	http://serialio.com

Tabelle 3.2: Javax.comm Implementationen für Windows Mobile

3.2 Graphische Aufbereitung

Um die empfangen Kompassdaten visuell aufzubereiten, ist es nötig, die vorhandenen Rahmenbedingungen auszuloten. Wesentliche Limiten sind in der verwendeten Hardwarekonfiguration zu finden. Dabei ist an den *Hauptspeicher* (64MB SRAM), die *Prozessorleistung* (Intel PXA270-Prozessor mit 624 MHz), die reduzierte *Displaygrösse* (640x480 Pixel) oder an die *Speicherzugriffszeit* zu denken. Zusammengefasst sind dies die **hardwaretechnischen Voraussetzungen**. Darauf aufbauend kommen weitere limitierende Faktoren. Nämlich die vorgebene Java Laufzeitumgebung und das Standard Widget Toolkit (SWT) (Beides wird in Kapitel 2 beschrieben). Dies sind die **softwaretechnischen Voraussetzungen**. Aufbauend auf diesen Voraussetzungen bleiben dem Entwickler verschiedene grundsätzliche Techniken bzw. **Darstellungsarten**, um Kompassdaten zu visualisieren. Das Resultat dieser Visualisierung ist die **Benutzerwahrnehmung**, die abhängig von der verwendeten Darstellungsart varieren kann. Diese Situation ist in Abbildung 3.2 auf Seite 23 schematisch zusammengefasst. Um die Struktur und das weitere Vorgehen dieser Arbeit klar zu halten, werden die Begriffe Darstellungsart und Benutzerwahrnehmung in den folgenden beiden Abschnitten noch präzisiert.

Darstellungsart

Der Begriff „Darstellungsart“ ist sehr weitläufig. An dieser Stelle soll er deshalb eingegrenzt werden, damit für den weiteren Verlauf der Arbeit eine einheitliche Terminologie besteht. Wie bereits oben erwähnt, geht es um die graphische Darstellung von Kompassdaten. Bei dieser graphischen Aufbereitung hat der Programmierer verschiedene grundsätzliche Arten zur Verfügung, die er anwenden kann. Unabhängig davon, welche Art der Entwickler wählt, ist es das übergeordnete Ziel, dem Benutzer ein Gefühl der Orientierung zu geben, ähnlich wie ein Kompass. Um dieses

Ziel zu erreichen, wurden drei unterschiedliche Arten der Darstellung genauer untersucht. Diese drei Arten sollen an dieser Stelle eingeführt werden. Aus Gründen der Einfachheit werden sie Darstellungsart I, II und III genannt.

- **Darstellungsart I: Einfache Grafikoperationen** Bei der Darstellungsart I, beschränkt man sich lediglich auf einfache grafische Operationen, wie etwa das Zeichnen eines Punktes, einer Linie, einfacher geometrischer Figuren oder Kombinationen davon. Es wird insbesondere auf die Verwendung von zusätzlichen Elementen, wie Bilder, verzichtet.
- **Darstellungsart II: Online Rotation** Die Darstellungsart II benutzt im Gegensatz zur Darstellungsart I als zusätzliche Elemente Bilder bei der Darstellung. Um dem Benutzer das Gefühl von Orientierung zu vermitteln, werden diese Bilder (z.B. Karten) während der Laufzeit (deshalb *online* Rotation) gedreht.
- **Darstellungsart III: Offline Rotation** Die dritte Darstellungsart verwendet, wie die Darstellungsart II, Bilder als zusätzliche Elemente Bilder zur Darstellung. Auch hier sollen die Bilder rotiert werden, um dem Benutzer seine aktuelle Orientierung zu verdeutlichen. Im Unterschied zur vorgängigen Darstellungsart wird die Rotation jedoch nicht während, sondern vor der Laufzeit berechnet (deshalb *offline* Rotation genannt) und dann während der Laufzeit als gespeicherte Bilder abgerufen.

Zusammengefasst kann man sagen, dass diese Arbeit sich auf die drei oben definierten Darstellungsarten beschränkt und diese auf Ihre Eignung für die Visualierung von Kompassdaten überprüft. Es soll hier noch erwähnt werden, dass Darstellungsart II und III in den meisten Fällen substituiv zum Einsatz kommen können.

Benutzerwahrnehmung

Das Thema Darstellungsarten wurde im vorhergehenden Abschnitt behandelt. Generell gesagt, beschreiben diese das Vorgehen um ein bestimmtes Resultat zu erreichen. Dabei ist das Resultat dasjenige, was der Benutzer wahrnimmt. Deshalb folgt nun das Thema der Benutzerwahrnehmung. Auch dieser Begriff soll präzisiert werden.

In der vorliegenden Arbeit wird die Benutzerwahrnehmung in zwei Typen unterteilt. Erstens jene Visualisierungen, die der Benutzer in **Echtzeit** wahrnimmt und zweitens, die Darstellungen, welche nicht echtzeittauglich sind und nur im sogenannten **Pullmodus** zur Verfügung stehen.

Aus Benutzersicht gibt es folglich zwei Modi: Echzeitmodus und Pullmodus.

Zuerst zum **Echtzeitmodus**. Es gibt viele Definitionen von Echtzeit, wobei die meisten davon sehr ähnlich sind. Hier kommt folgende Definition zur Anwendung:

> „Der Begriff Echtzeit legt lediglich fest, dass ein System auf ein Ereignis innerhalb eines vorgegebenen Zeitrahmens reagieren muss. Der Begriff sagt nichts über die Geschwindigkeit oder Verarbeitungsleistung eines Systems aus.“ [Wikipedia, 2006]

Im Rahmen dieser Arbeit wird für den Begriff Echtzeit folgende Definition verwendet: Eine Änderung der Kompassposition soll innerhalb von 100 Millisekunden nach Signaleingang dargestellt sein. 100 Millisekunden entsprechen einer Rate von 10 Bildern pro Sekunde. Die üblichen Rate von 24 Bilder pro Sekunde (FPS), wie sie als Grenze bei Animationen und Filmen normalerweise verwendet wird, ist im Kontext dieser Arbeit keine sinnvolle Begrenzung. Die Arbeit zielt nicht darauf ab, dem Benutzer das Gefühl von lebensechter Bewegung zu geben, sondern sie will ein Feedback über seine Richtungsänderung in einem vernünftigen Zeitrahmen zu vermitteln. Da ein Benutzer mit sehr hoher Wahrscheinlichkeit deutlich weniger als zehnmal pro Sekunde signifikante Richtungsänderungen vornehmen wird, werden die zehn FPS in diesem Sinne als Grenze für Echtzeit verwendet. Ist eine Darstellungsart echtzeittauglich, macht es Sinn, dass Aktualisierungen ohne Zutun des Benutzers vonstatten gehen. Das heisst, eine Aktualisierung der Kompassposition passiert aus Benutzersicht automatisch.

Nachdem nun der Begriff Echzeitmodus klar abgegrenzt worden ist, soll auch das Verständnis des zweit genannte Types, nämlich des **Pullmodus** noch geklärt werden. Unter dem Pullmodus werden in dieser Arbeit folglich die vom Benutzer mit einer grösseren Verzögerung als 100 Millisekunden wahrgenommenen Visualisierungen verstanden. Da diese Darstellungen eine längerer Verzögerung aufweisen, macht es keinen Sinn, dass Aktualisierungen ohne Zutun des Benutzers vonstatten gehen. Der Benutzer muss eine Aktualisierung aktiv anfordern, damit sie ihm angezeigt wird. Deshalb spricht man von einem „Pullmodus“. Tabelle 3.3 vergleicht die beiden unterschiedlichen Typen der Benutzerwahrnehmung und fasst sie zusammen.

Echtzeitmodus	Pullmodus
• Sobald neue Sensordaten verfügbar sind, werden sie ohne Zutun des Benutzers visualisiert.	• Der Benutzer muss explizit eine Aktualisierung der Sensorinformation anfordern.
• Die Software reagiert in einem vorgesehenen maximalen Zeitrahmen, das heisst es gibt eine maximal mögliche Verzögerung. Diese beträgt im Falle der vorliegenden Arbeit 100 Millisekunden.	• Es besteht keine Limite hinsichtlich der Verzögerung bei der Darstellung.

Tabelle 3.3: Pull- und Echtzeitmodus im Vergleich

Zusammenfassung

Nach diesen ausführlichen Erklärungen der verwendeten Begrifflichkeiten sollen diese, im Sinne einer Zusammenfassung, in einen einheitlichen Rahmen gesetzt werden. Abbildung 3.2 veranschaulicht den theoretischen Aufbau. Wie zu Beginn erwähnt, basiert diese Machbarkeitsanalyse auf bestimmten **softwaretechnischen** und **hardwaretechnischen Voraussetzungen** (in der Grafik rot dargestellt). Auf diesen aufbauend sollen drei **Darstellungsarten** evaluiert und auf Ihre Machbarkeit hin getestet werden. Die für den Benutzer wichtigen Faktoren bei diesen Tests sind unter dem Begriff **Benutzerwahrnehmung** zusammengefasst. Sie geben an, ob eine Darstellungsart **echtzeit**tauglich (Definition siehe oben) ist, oder eben lediglich in einem sogenannten **Pullmodus** realisierbar ist.Um dies zu prüfen werden nun für die drei definierten Darstellungsarten prototyphafte Implementierungen vorgenommen und mit diesen Messungen durchgeführt. Als Messkriterien wurden die **Verzögerung** und der **Speicherverbrauch** identifiziert. Die nachfolgenden drei Unterkapitel geben darüber Auskunft, wie die Implementierung vorgenommen wurde, was dabei die Besonderheiten waren und welche Resultate die Messungen zu Tage geführt haben.

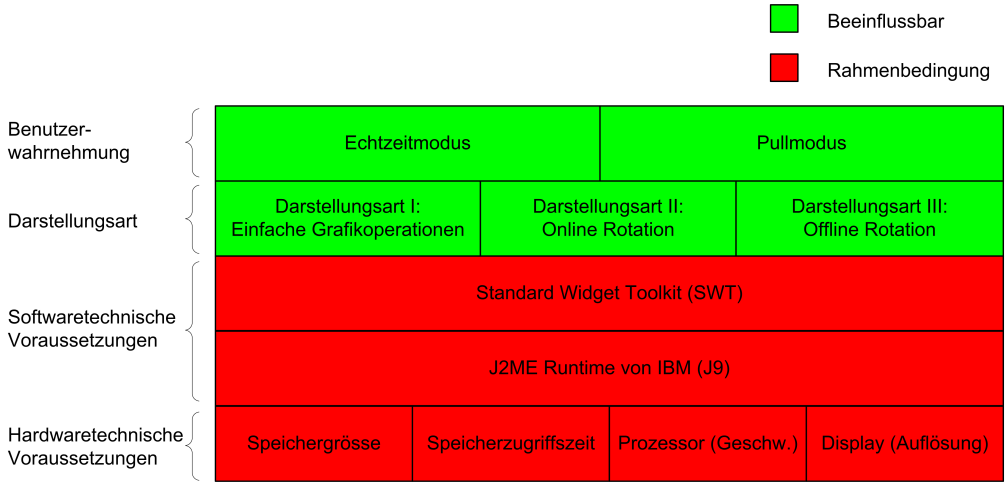

Abbildung 3.2: Grafische Aufbereitung: Schematische Darstellung der Programmiersituation

3.2.1 Darstellungart I: Einfache Grafikoperationen

Die **Idee** der Darstellungsart I besteht darin, ausschliesslich einfache Grafikoperationen zu verwenden, um Kompassinformationen darzustellen. Einfache Grafikoperationen beschränken sich auf die durch die verwendete Graphikbibliothek SWT (für detailiertere Informationen zum SWT siehe Kapitel 2.3.1) direkt zur Verfügung gestellten Funktionen, die das Zeichnen von einfachen geometrischen Figuren (wie Punkte, Linien, etc.) ermöglichen. Zusätzliche Elemente wie Bilder dürfen bei dieser Darstellungsart nicht verwendet werden.

Grundlage

Das Standard Widget Toolkit (SWT) wurde in erster Linie entwickelt, um grafische Benutzerschnittstellen (GUI) zu programmieren. Dabei stellt es eine grosse Auswahl an Elementen zur Verfügung um benutzerfreundliche GUIs zu konstruieren. Das SWT wurde seit seiner Erstentwicklung im Jahr 2001 kontinuierlich verbessert und bietet heute viele Erweiterungen im Graphikbereich an. So zum Beispiel Java Advanced Imaging, OpenGL Support oder Java2D Unterstützung. Die schlechte Nachricht dabei ist, dass all diese grafischen Erweiterungen längst nicht auf allen Platformen zur Verfügung stehen. Pocket PC bzw. Windows CE Platformen gehören dabei zu den grössten Verlierern, denn für diese ist noch nicht einmal der normale Funktionsumfang der SWT Bibliothek erhältlich, sondern es steht nur eine reduzierte Variante zur Verfügung. Die J2ME-Version von SWT unterstüzt deshalb nur einfachste 2D Grafikoperatione, wie das Zeich-

nen von Linien, Punkten und einfachen Figuren wie Kreise, Rechtecke und Polygone. Das sind die Eckpunkte welche bei dieser Arbeit zu berücksichtigen sind. Dieses Unterkapitel soll prüfen, ob diese simplen SWT Grafikoperationen eine realistische Möglichkeit zur Visualisierung von Kompassdaten sind.

Um das zu tun wurden einige prototyphafte Implementierungen vorgenommen, welche verschiedene Messungen zulassen. Der Quellcode befindet sich auf der beiliegenden CD-ROM (Anhang E auf Seite 104). Bevor nun das Thema der Messungen dargelegt wird, sollen noch einige technische Besonderheiten erwähnt werden, welche bei dieser Darstellungsart zu berücksichtigen sind. Der technisch weniger interessierte Leser kann den Abschnitt überspringen.

Besonderheiten

Experimente mit SWT Grafikoperationen verdeutlichten zwei Punkte, welche für eine flimmerfreie Echtzeitvisualisierung wesentlich sind. Zum einen sollte bei jeglichen Grafikoperationen Double Buffering[3] verwendet werden. Das bedeutet konkret, dass sämtliche Zeichenoperationen zuerst auf einem nicht sichtbaren Grafic Context (GC), ausgeführt werden und erst danach als Ganzes sichtbargemacht werden. Das führt dazu, dass die für die Grafikoperationen benötigte Berechnungszeit nicht gleich mitvisualiert wird, denn das würde zu Flimmern führen. Ein weiterer erwähnenswerter Punkt ist, dass SWT vor einer Aktualisierung standardmässig das ganze Display leert, also weiss übermalt. Auch das führt zu Flimmern. Dieses Verhalten von SWT kann jedoch mit der Option `SWT.NO_BACKGROUND` unterbunden werden. So wird einfach der neue Displaybuffer darüber gezeichnet, ohne dabei vorher das Display weiss zu übermalen.

```
Shell shell = new Shell(parentShell,SWT.NO_BACKGROUND);
```

Messungen

Der Vorteil bei solchen Grafikoperationen ist, dass sie sehr wenig zusätzlichen Hauptspeicher brauchen (ausser für das Double Buffering). Bei vorgenommenen Messungen stieg der für die Darstellung verwendete Hauptspeicher nicht über 300KB und kann deshalb als Messkriterium vernachlässigt werden. Interessant für die Machbarkeit ist jedoch die Verzögerung, welche bei den Grafikoperationen entsteht. Messungen während dieser Arbeit haben ergeben, dass mit dem

[3] Unter Double Buffering versteht man eine Technik bei der die Bildberechnung und Darstellung getrennt werden um Flackern zu vermeiden.

gegebenen Hardware und Software Setup, in jedem geprüften Fall weniger als 60 Millisekunden benötigt werden um eine Grafik darzustellen. Das erklärt sich dadurch, dass SWT Zugang zum sogenanntes „Native Interface“ hat. Konkret bedeutet das, dass SWT direkt die grafischen Elemente des Betriebssystems nutzen kann.

Zusammenfassung

Zusammenfassend kann man sagen, dass die SWT Grafikoperationen zwar eingeschränkt sind, jedoch die Performanz in unserem Fall eine Echtzeitvisualisierung der Sensordaten ermöglicht. Bisher haben wir uns auf einfache Grafikoperationen beschränkt, ohne dabei externe Bilder zu verwenden und zu manipulieren. Hinsichtlich einer Erweiterung des bestehenden MobileGames sollten aber auch komplexere Darstellungsarten in Betracht gezogen werden.

3.2.2 Darstellungsart II: Online Rotation

Die Rotation ist in der Navigation eine häufiges Muster. So rotiert zum Beispiel eine Kompassnadel oder ein Zirkel bei der Peilung auf einer Seekarte. Die **Idee** im vorliegende Fall ist, die Richtungsänderung des Gerätes zu visualisieren, ähnlich wie ein Kompass. Angenehm wäre es zum Beispiel wenn ein Benutzer eine sich nach der Laufrichtung ausrichtende Karte auf dem Bildschirm hätte. Man kann dies auch als die Darstellung von Kontextinformationen sehen. Das für J2ME verfügabare SWT Paket ermöglicht keine direkten Transformationen an Bildern. So bleibt nur die Möglichkeit, selbst einen Rotationsalgorithmus zu programmieren. Der Begriff „Online" Rotation meint hier die Technik eine gegebene Pixelgrafik bzw. ein externes Bild während der Laufzeit zu rotieren und darzustellen. Im Gegensatz dazu wird bei einer Offline Rotation zur Laufzeit lediglich ein bereits vorgängig rotiertes Bild aufgerufen. In den folgenden Abschnitten werden die theoretische Grundlage der Online Rotation und deren Besonderheiten erläutert. Anschliessend werden alternative Algorithmen der Online Rotation gegenübergestellt. Danach werden Messungen beschrieben, welche Aufschluss über die vorhandenen Einflussfakoren auf die Performanz dieser Algorithmen geben sollen.

Theoretische Grundlage

Will man einen Punkt in einem Koordinatensystem um den Nullpunkt drehen, so verwendet man eine Matrixrotation. Ein Bild wird bei einer Matrixrotation als eine Reihe von Nullvektoren interpretiert. Jedes Pixel entspricht einem Vektor. Für jeden Vektor werden dann die neuen Koordinaten bei einer vom Nullpunkt ausgehende Drehung um eine bestimmte Anzahl von Grad berechnet. Dies geschieht durch die Multiplikation eines Vektors mit der so genannten Drehmatrix, welche wie folgt definiert ist:

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.1}$$

Soll also der Vektor $\vec{a_{alt}}$ um den Winkel α gedreht werden so kann der resultierende Vektor $\vec{a_{neu}}$ wie folgt berechnet werden:

$$\vec{a_{neu}} = \mathbf{R} \cdot \vec{a_{alt}} \tag{3.2}$$

Winkel α muss dabei im Bogenmass übergeben werden. Eine Illustration dieser Rotation befindet sich in Abbildung 3.3 auf Seite 27. Wenn man die Gleichung 3.2 nach $\mathbf{x_{neu}}$ und $\mathbf{y_{neu}}$ auflöst, resultiert:

$$\mathbf{x_{neu}} = \mathbf{x_{alt}} \cos\alpha - \mathbf{y_{alt}} \sin\alpha \tag{3.3}$$

$$\mathbf{y_{neu}} = \mathbf{x_{alt}} \sin\alpha + \mathbf{y_{alt}} \cos\alpha \tag{3.4}$$

Das sind die theoretischen Grundlagen der Matrixrotation. In der praktischen Anwendung gibt es jedoch eine Besonderheit zu beachten, auf welche im folgenden Abschnitt eingegangen wird.

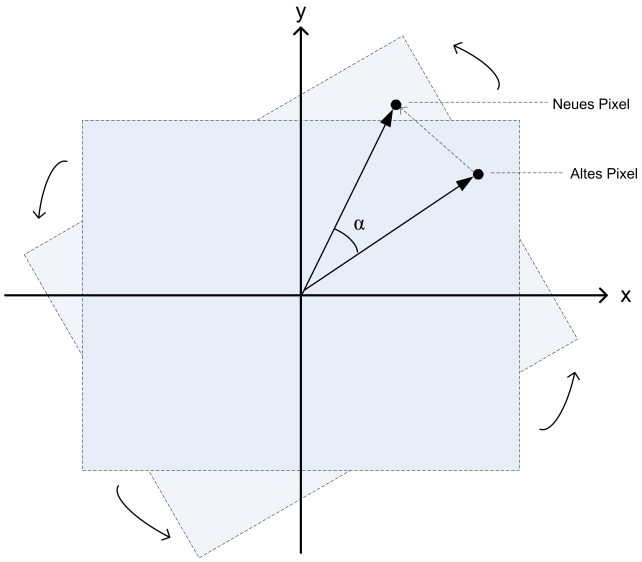

Abbildung 3.3: Rotation mittels Drehmatrix

Besonderheiten

Betrachtet man die Gleichung 3.3 als Funktion $\mathbb{R} \mapsto \mathbb{R}$ so ist diese für gegebenes α und $\mathbf{y_{alt}}$ bijektiv bzw. eineindeutig. Dasselbe gilt natürlich für Gleichung 3.4. Daraus lässt sich schliessen, dass die Matrixdrehung eine Funktion $\mathbb{R} \times \mathbb{R} \mapsto \mathbb{R} \times \mathbb{R}$ ist, die auf der euklidischen Ebene jedem reellwertigen Vektor exakt einen Vektor zuordnet. Soweit die Theorie. In der Praxis,

nämlich bei der Programmierung, haben wir es nicht mit einem euklidischen Raum der Form $\mathbb{R} \times \mathbb{R}$ zu tun, sondern mit einer Ebene vom Typ $\mathbb{N} \times \mathbb{N}$. Das bedeutet wir haben es mit einem diskreten Defintionsbereich zu tun, der aus natürlichen Zahlen besteht. Sinus und Cosinus sind jedoch reelle Funktionen. Konkret heisst das, dass die theoretische Bildmenge aus reellen Zahlen besteht, während man in der Praxis nur einen ganzzahlige Pixelraum zur Verfügung hat. Eine einfach Möglichkeit ganzzahlige Resultate zu erhalten, ist durch Rundung. Dadurch geht jedoch die erwähnte Bijektivität verloren. In einfachen Worten ausgedrückt kann es also passieren, dass zwei unterschiedliche Pixel der Definitionsmenge auf das selbe Pixel der Zielmenge abgebildet werden. Oder noch schlimmer, dass umgekehrt ein Pixel der Zielmenge überhaupt nicht durch die Funktion angesprochen wird. So entstehen Lücken und das Bild wird verzerrt (Abbildung 3.4).

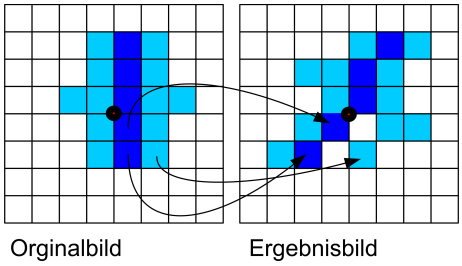

Abbildung 3.4: Direkte Rotation: Es entstehen Lücken

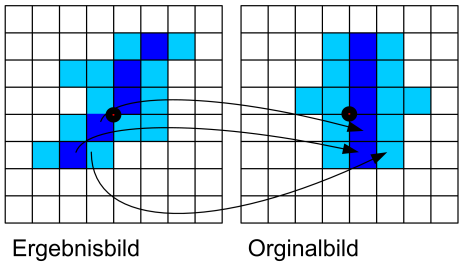

Abbildung 3.5: Indirekte Rotation: Lücken können vermieden werden

Das Problem der Lücken kann umgangen werden, in dem man eine indirekte Transformation vornimmt und für jedes Pixel im Ergebnisbild den Farbwert im Orginalbild sucht (Abbildung 3.5). So ist sichergestellt, dass jedes Pixel des Ergebnisbildes mit einem Farbwert belegt ist und keine Lücken enstehen. Trotzdem ist der Verzerrungseffekt noch nicht gelöst. Denn auch bei einer indirekten Transformation wird normalerweise keine ganzzahlige Pixelposition gefunden. Die Frage

lautet folglich: Welcher Farbwert soll am Ergebnisbildpunkt verwendet werden? Wenn man die reellen Pixelangaben einfach rundet führt das nach wie vor zu Verzerrungseffekten. Diese Effekte könnten jedoch durch Interpolation der Farbwerte vermindert werden. Das bedeutet, den Farbwert eines Ergebnisbildpunktes berechnet man aus den gewichteten Farbewerten benachbarter Orginalbildpunkte. Wichtig ist, dass man dies für die drei RGB Kanäle getrennt tut.
Eine Bild zu drehen bedeutet, dass für jedes Pixel die Gleichungen 3.3 und 3.4 gelöst werden müssen. Bei einem Bild mit 600x600 Pixel wären das also 360'000 Berechnungen. Um die Echtzeittauglichkeit der Online Rotation zu prüfen, wurden deshalb verschiedene Messungen durchgeführt.

Auswahl des Rotationsalgorithmus

Auf Basis der oben beschriebenen theoretischen Grundlagen wurden zwei verschiedene Algorithmen der Online Rotation entwickelt:

- **Variante I:** Online Rotation mit anschliessender Zeichnung: Hier wird ein Bild im Speicher Pixel für Pixel gedreht und erst danach als Objekt mit der SWT `drawImage` Methode gezeichnet.

- **Variante II:** Online Rotation mit gleichzeitiger Zeichnung: Auch bei dieser Variante wird ein Bild Pixel für Pixel gedreht. Jedoch wird nach der Berechnung der neuen Pixelposition auch sofort die Zeichnung des Pixels vollführt. Die Idee dabei ist, dass auf diese Weise nur einmal mit einer Schleife das Bild durchlaufen werden muss. Die Annahme ist natürlich, dass die `drawImage` Methode von SWT ein Bild ebenfalls Pixel für Pixel zeichnet und es dabei mit einer Schleife durchgeht.

Die beschriebenen zwei Algorithmen der Online Rotation wurden mit Zeitmessungen evaluiert. Das Ziel war es herauszufinden, wie lange es auf dem iPAQ dauern würde um ein Bild zu laden, zu drehen und auf den Bildschirm zu zeichnen. Die Messungen haben ergeben, dass bei einem 600x600 Pixel Bild die Online Rotation von Variante 1 auf dem iPAQ 3 Sekunden dauert. Dieselbe Rotation auf einem handelsüblichen Notebook getestet, dauerte 0.2 Sekunden. Die Resultate bei Online Rotationen der Variante 2 sind hinsichtlich der benötigten Laufzeit nicht besser. Genau genommen ist das Gegenteil der Fall. Wird während der Berechnung der Bildpunkte gleichzeitig noch gezeichnet, benötigt die Software ein Vielfaches von der Zeit um ein gedrehtes Bild auf den Bildschirm zu bringen. Abbildung 3.6 auf Seite 30 vergleicht die beiden vorgestellten

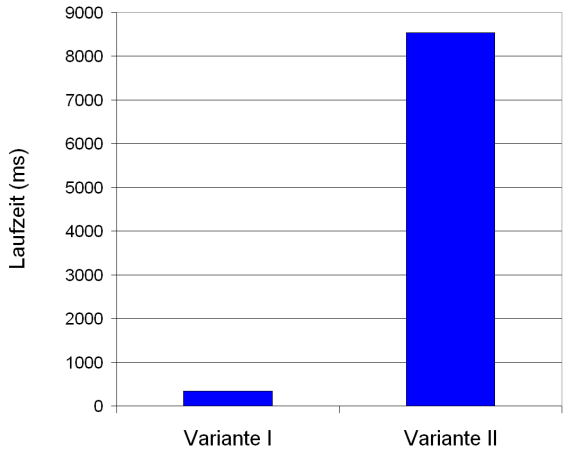

Abbildung 3.6: Zeitmessung für zwei unterschiedlichen Algorithmen zur Online Rotation

Varianten der Online Rotation. Dabei handelt es sich um die totale Laufzeit (wobei jedoch bei beiden Varianten die meiste Zeit zur Rotationsberechnung benötigt wird) um ein 100x100 GIF-Bild zu drehen und abzubilden.

Erklären könnte man das damit, dass SWT auf hardwarenahe Programmbibliotheken zurückgreift und möglicherweise für die Zeichnung eines Bildes Methoden verwendet, die dem Anwendungsprogrammierer innerhalb von SWT nicht direkt zur Verfügung stehen. Aufgrund der vorgenommenen Messungen kann man davon ausgehen, dass mit SWT wider Erwarten Variante I der Online Rotation effizienter ist als die Variante II. Deshalb basieren sämtliche weiteren Messungen im Bereich der Online Rotation auf diesem Algorithmus (Variante I).

Zeitverbrauch je Prozessschritt

Der vorhergehende Abschnitt hat die Entscheidungen für einen von zwei möglichen Algorithmen für die Rotation dargelegt. In der folgenden Analyse soll der Algorithmus nun in mehrere Prozessschritte zerlegt werden. Dabei soll analysiert werden, welche Schritte prozentual welchen Zeitanteil der gesamten Laufzeit benötigen. Das Diagramm in Abbildung 3.7 auf Seite 31 zeigt die prozentuale Verteilung der Zeit, welche für die verschiedenen Prozessschritte benötigt wird, wenn der Prototyp mit einem 600x600 Pixel[4] GIF-Bild getestet wird. Auffallend dabei ist, dass neben der Rotation auch Prozessschritt Nr. 2, nämlich das „Bild laden“, einen verhältnismässig

[4] 600x600 Pixel entsprichen ziemlich genau einer durchschnittlichen Karte wie sie in vergangenen MobileGame Experimenten verwendet wurde.

hohen Anteil hat. Da jedoch bei der Online Rotation das Bild nur einmal geladen werden muss, nämlich zu Beginn, kann das Laden als Prozessschritt vernachlässigt werden.

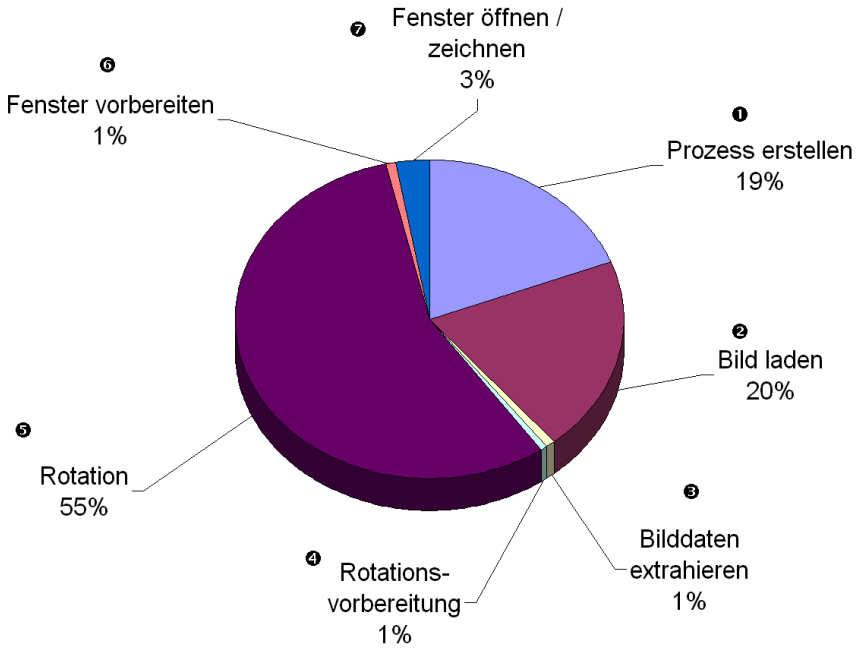

Abbildung 3.7: Zeitverteilung bei der Rotation und Darstellung eines 600x600-Pixel Bildes

Im Anschluss an die bisherigen Analysen wurden zwei Zeitmessungen vorgenommen. Es sollte geprüft werden, welche Einflussfaktoren es auf die Laufzeit des Algorithmus gibt. Es wurde zum einen die Bildgrösse und zum anderen der Drehwinkel als Einflussfaktor in Betracht gezogen.

Messung I: Zeitverbrauch mit Einflussfaktor Bildgrösse

Bei dieser Messung wurde auf die benötigte Laufzeit fokussiert, wenn man ceteris paribus die Bildgrösse bzw. die Anzahl der Pixel verändert. Das Resultat wird durch die Abbildung 3.8 auf Seite 32 visualisiert. Es zeigt deutlich, dass nur zwei Prozessschritte durch die Bildgrösse beeinflusst werden: Die Rotation und das Laden des Bildes.

Messung II: Zeitverbrauch mit Einflussfaktor Rotationswinkels

Es wurden im Verlauf dieser Projektphase auch noch weitere mögliche Einflussfaktoren auf die Laufzeit einer Online Rotation untersucht. So zum Beispiel der Winkel, um welchen rotiert wird.

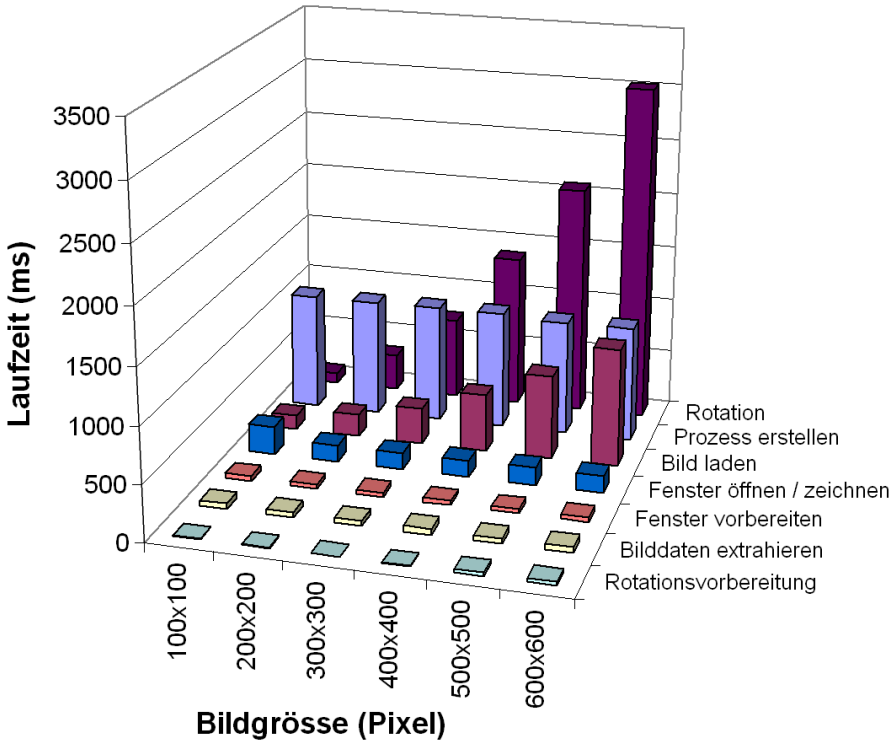

Abbildung 3.8: Zeitmessung: Einfluss der Bildgrösse auf die Laufzeit

Die Feststellung dabei war, dass die Bildgrösse den einzigen Faktor mit Einfluss auf die Laufzeit darstellt. Die Messergebnisse sämtlicher Messungen befinden sich auf der beiliegenden CD-ROM (Anhang E auf Seite 104).

Zusammenfassung

In den oben beschriebenen Abschnitten wurde auf verschiedene Aspekte der Darstellungsart II, nämlich der „Online Rotation“ eingegangen. Zuerst wurden die theoretische Grundlage und die damit verbundenen Besonderheiten erläutert. Anschliessend wurden zwei alternative Algorithmen (oben als Variante I und Variante II bezeichnet) der Online Rotation miteinander verglichen. Dabei wurde entschieden, weitere Messungen nur noch mit der zweiten Variante durchzuführen. Eine weitere Analyse hat die Online Rotation in einzelne Prozessschritte zerlegt und ihre prozentuale Laufzeit betrachtet. Im weiteren wurden dann der Einfluss von Bildgrösse und der Drehwinkel auf die Laufzeit gemessen. Mit den gemachten Messungen steht fest, dass 600x600 Pixel auf dem mobilen Gerät in minimal 3 Sekunden gedreht werden können. Die Resultate fasst Tabel-

le 3.4 auf Seite 33 zusammen. Die Darstellungsart II wurde nun eingehend betrachtet. An nächster Stelle steht die Darstellungsart III, die Offline Rotation.

Messung	**Einflussfaktor**	**Erkenntnisse**
Messung I	Bildgrösse	Die Laufzeit nimmt proportial mit der Anzahl Pixel zu. Für die Drehung wird im Mittel 0.00833 Millisekunden pro Pixel benötigt.
Messung II	Drehwinkel	Es wurde festgestellt, dass der Drehwinkel keinen Einfluss auf die Laufzeit hat

Tabelle 3.4: Übersicht über Zeitmessungen bei der Online Rotation

3.2.3 Darstellungsart III: Offline Rotation

Die Offline Rotation ist, im Gegensatz zur oben beschriebenen Online Rotation, ein Verfahren das Bilder vorgängig als rotierte Dateien speichert und dann während der Laufzeit in den Hauptspeicher lädt. Die **Idee** ist, dass die Applikation vom zeitlichen Aufwand für die Rotation befreit (siehe auch Abbildung 3.7). Es beginnen jedoch andere Faktoren eine kritische Rolle zu spielen. Während bei der Online Rotation ein Bild nur einmal geladen werden muss und dann nur noch entsprechend gedreht wird, muss im Falle der Offline Rotation für jede mögliche Drehung, sprich Kompassposition, ein neues Bild geladen werden. Das bedeutet, dass nun Einflüsse wie Hauptspeicher und Ladezeit Fokus der Untersuchung werden. Zuerst wird die theoretische Grundlage des Ladeprozesses eines Bildes erkläutert. Anschliessend wird der Hauptspeicherverbrauch analysiert. Dann werden verschiedene Einflussfaktoren auf den Ladeprozess mittels Messungen untersucht.

Theoretische Grundlage

Wie bereits beim Vorstellen der Hardware ausgeführt wurde, stehen dem Programmierer ungefähr 40MB[5] Hauptspeicher zur Verfügung. Die bei dieser Arbeit vorgegebene API zur Grafikprogrammierung ist wie bereits erwähnt SWT. Beim Laden einer Bilddatei kreiert SWT ein Image-Objekt, welches die Bilddaten beinhaltet. Dieses kapselt die Rohdaten der Bilder als ein Array von ganzzahligen Werten, bzw. für jeden Bildpunkt wird ein Integerwert festgehalten. Dieser Integerwert markiert die Position der Farbtabelle, an welcher der RGB Farbwert für den Bildpunkt gespeichert ist. Diese Erkenntnis ermöglicht es, den Speicherbedarf eines SWT Image-Objektes genau zu berechnen und das unabhängig vom verwendeten Bildformat (Gleichung 3.5).

$$\text{Speicherbedarf eines Image-Objektes (in KB)} = \left(\frac{\text{Höhe in Pixel} * \text{Breite in Pixel}}{1024}\right) \tag{3.5}$$

So hat also ein 700x700 GIF Bild, das komprimiert gespeichert 50KB benötigt, als Image-Objekt eine Grösse von 478 KB. Das ist eine Zunahme um beinahe Faktor Zehn. Abbildung 3.9 verdeutlicht diesen Sachverhalt schematisch.

[5] Der iPAQ hx4700 verfügt über 64MB SRAM, der je nach Konfiguration zu ca. 60% als Programm- und zu ca. 40% als Datenspeicher fungiert.

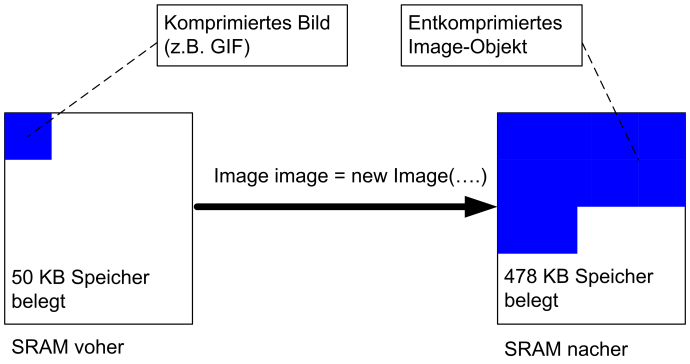

Abbildung 3.9: Speicherbedarf bei der Erstellung eines Image-Objektes

Zusammenfassend kann man sagen, dass einzig die Anzahl Pixel eines Bildes den Hauptspeicherverbrauch eines Image-Objektes bestimmen und dass Farbtiefe und Bildformat darauf keinen Einfluss haben. Der folgenden Abschnitt soll verdeutlichen, was das für die Machbarkeit bedeutet.

Analyse des Hauptspeicherverbrauchs

Die theoretischen Grundlagen haben deutlich gemacht, dass SWT Image-Objekte Bilddaten unkomprimiert im Hauptspeicher halten. Die zentrale Idee der Offline Rotation ist, dass sämtliche vorgedrehte Karten bei Applikationsstart in den Speicher geladen werden, um während der Laufzeit sofort angezeigt werden zu können. Dabei muss auch berücksichtigt werden, dass im MobileGame in der Regel mehr als eine Karte verwendet wird. Es stellt sich nun die Frage, ob diese Idee realisierbar ist. Um zu das prüfen, wird in diesem Abschnitt der Hauptspeicherverbrauch untersucht, in Abhängigkeit von Anzahl Karten, Anzahl Bildpunkte und Feinheitsgrad der vorgedrehten Karten. Der Feinheitsgrad gibt das Ausmass der Winkelveränderung an, für welches eine neue Karte zur Verfügung steht. Zum Beispiel bedeutet ein Feinheitsgrad von 4°, dass jeweils eine vorgedrehte Karte für 4°, 8°, 12°, ..., 360° zur Verfügung steht. Total wären das 90 gespeicherten Karten. Tabelle 3.5 gibt eine Übersicht über den theoretischen Speicherbedarf von gepufferten und vorgedrehten Karten.

Das aktuelle MobileGame von Christoph Göth arbeitet mit sieben Karten die im Durchschnitt 645 x 604 Bildpunkte gross sind. Bei einem Feinheitsgrad von 12° (was schon sehr Lückenhaft ist) würden also rund 72 MB Hauptspeicher belegt. Möchte man nur immer die vorgedrehten Bilder einer Karte puffern, würde das ungefähr 10 MB RAM benötigen. Jedoch wäre jeder Kartenwech-

Anzahl Karten	Theoretische berechenbarer Hauptspeicherbedarf (MB)											
	500x500 Pixels				600x600 Pixels				700x700 Pixels			
	6°	8°	10°	12°	6°	8°	10°	12°	6°	8°	10°	12°
1	14	11	9	7	21	15	12	10	28	21	17	14
5	72	54	43	36	103	77	62	52	140	105	84	70
7	100	75	60	50	144	108	87	72	196	147	118	98
9	129	97	77	64	185	139	111	93	252	189	151	126

Tabelle 3.5: Theoretischer Hauptspeicherverbrauch in MB bei gepufferten Karten

sel mit einer längeren ladebedingten Wartezeit verbunden, da dann die neuen Bilder während des Spieles geladen und gepuffert werden müssten. Das würde zu einem stockenden Spielverlauf führen. Es gilt auch zu bedenken, dass 10 MB bereits viel ist, angesichts der Tatsache, dass das MobileGame allein zwischen 30 MB RAM benötigt.

Es wird klar, dass im Hinblick auf eine Verwendung beim MobileGame der Hauptspeicher zu knapp ist, um sämtliche Karten oder eine nützliche Teilmenge davon zu puffern. Das bedeutet, dass eine künftige Anwendung mit Darstellungsart II mehrere Ladezprozesse beinhalten würde. Deshalb wird in den nachfolgenden Messungen untersucht, welche Parameter die Dauer des Ladeprozess von Bilder beinflussen. Die Idee ist, dass man diesen optimieren kann.

Messung I: Ladezeit mit Einflussfaktor Bildformat

Messung I untersucht den Einfluss des Bildformates auf den Ladeprozess. Bei SWT wird ein Bild mit einer Funktion geladen und daraus ein Image-Objekt erstellt. Es handelt sich dabei um den bereits in Abbildung 3.7 auf Seite 31 beschriebenen Prozessschritt „Bild laden“. Aus der Sicht des Programmierers ist dies ein geschlossener Prozessschritt. Im Hintergrund führt Java natürlich eine Reihe von Operationen durch, welche dann am Ende das besagte Image-Objekt erzeugen. Dieser Aufbau entspricht den Prinzipien der Modularität, nach denen der Programmierer zur Verwendung eines Moduls keine Kenntnisse über dessen inneren Aufbau haben muss [Glinz, 2004]. Bei SWT widerspiegelt sich dieses Prinzip zum Beispiel darin, dass der Programmierer sich nicht darum kümmern muss, welches Bildformat er übergibt, denn das resultierende Image-Objekt wird immer gleichermassen erstellt und beinhaltet eine Farbtabelle sowie für jedes Pixel einen Integerwert, welcher der Indexposition der Farbtabelle entspricht. Dabei können dem Kon-

struktor des Image-Objektes fünf Dateiformate übergeben werden: GIF, JPG, TIF, PNG und BMP. GIF, JPG und PNG sind komprimierende Bildformate. TIF und BMP[6] hingegen komprimieren die Bildinformationen nicht. Wenn Java ein komprimiertes Bild lädt, muss es zuerst einen Dekomprimierungsalgorithmus verwenden, um das Bild dann anschliessend unkomprimiert und als Image-Objekt im Hauptspeicher zur Verfügung zu haben. Dekomprimierungen können sehr zeitaufwändig sein. Das bedeutet, dass die für den Prozessschritt „Bild laden“ gemessene Zeit signifikant durch die Dekomprimierung verfälscht werden kann. Mit Experimenten wurde der Einfluss des Bildformates auf die Ladezeit erforscht, um anschliessend das für den Zweck dieser Arbeit beste Bildformat auszuwählen. Die Resultate lassen sich kombiniert mit den Resultaten der Messung II in Abbildung 3.10 auf Seite 38 finden. Man kann dem Diagramm entnehmen, dass die Zeit um ein Image-Objekt aus den beiden Formaten BMP und TIF zu erstellen, verhältnismässig tiefer ist. Das liegt daran, dass es sich bei BMP und TIF um nichtkomprimierte Bildformate handelt. Die Bilddateien müssen also nicht wie bei JPG und GIF mit rechenaufwändigen Dekomprimierungsalgorithmen verarbeitet werden. PNG ist ein sehr junges Bildformat und ist, im Gegensatz zu GIF ein offener Standard. Deshalb lässt sich auch erklären, dass bei PNG, trotz Dekomprimierung, die Ladezeit nahezu wie bei BMP und TIF ist. Die geringste gemessene Zeit um aus diesem 800x800 Bild ein Image-Objekt zu erstellen, war die für das Format TIF. Sie beträgt 351 Millisekunden (von der RAM Disk).

Messung II: Ladezeit mit Einflussfaktor Speichermedium

Die Idee dieser Messung ist herauszufinden, ob die unterschiedlichen Speichermedien des iPAQ einen Einfluss auf die Ladzeit eines Bildes haben. Der iPAQ hx4700 verfügt neben den 64 MB SRAM auch über einen SD Card Slot. Die 64 MB SRAM können ebenfalls als Datenspeicher verwendet werden. Den SD Card Slot wurde für diese Messung mit einer Kensigton Ultimate 2 GB SD Card[7] belegt, die laut Herstellerangaben über 21 MB/s Lesegeschwindigkeit verfügen sollte. Das Ziel dieser Messung war, die wirklich zur Verfügung stehende Lesegeschwindigkeit der beiden Speichermedien zu messen. Dafür wurden Lesetests für identische Bilder aber mit verschiedenern Bildformaten gemacht. Abbildung 3.10 verdeutlicht die Resultate. Man sieht, dass für das selbe 800x800 TIF-Bild von der SD Card eine doppelt so lange Ladezeit (771 Millisekunden) benötigt wird als für einen Zugriff via SDRAM. Um allenfalls mehr Erkenntnisse über die zugrundeliegenden Dekomprimierungsalgorithmen zu erhalten, soll in einem weiteren Schritt

[6] Hinweis: Mit BMP ist gemeint Standard BMP und nicht BMP RLE

[7] Für mehr Informationen zu Kensington und deren Produkte: http://www.kensington.com/

der Einfluss der Bildgrösse, also der Anzahl Pixel, auf die Ladezeit erfasst werden.

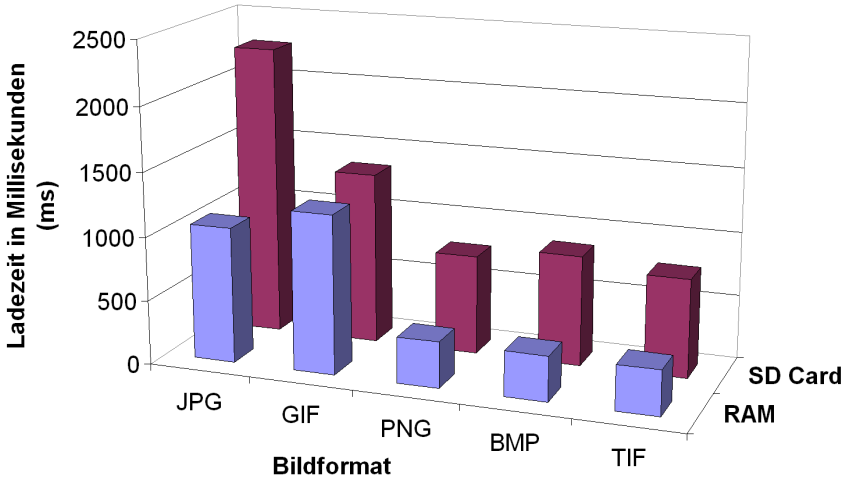

Abbildung 3.10: Ladezeitenvergleich: Interner SDRAM vs. externe SD Card

Messung III: Ladezeit mit Einflussfaktor Bildgrösse

Diese Messung will testen, wie die Ladezeit von der Bildgrösse (Anzahl Pixel) abhängt. Zur Durchführung wurden vergleichbar komplexe Bilder mit quadratisch zunehmender Fläche verwendet. Für sämtliche fünf Bildformate wurde dabei die Ladezeit gemessen und verglichen. Messung III macht klar, dass die Entwicklung der Ladezeit linear zu der Bildfläche verläuft. Die Abbildung 3.11 auf Seite 39 zeigt, dass quadratisch zunehmende Bildgrössen zu quadratisch ansteigenden Ladezeiten führen. Neben der Bildgrösse soll nun auch noch der Bildinhalt als möglicher Einflussfaktor auf die Ladezeit in Betracht gezogen werden. Die nächste Messung realisiert das durch Berücksichtigung verschieden komplexer Bildinhalte.

Messung IV: Ladezeit mit Einflussfaktor Bildkomplexität

Um die These zu untermauern, dass ein Grossteil der gemessenen Ladezeit auf die Komprimierung zurückgeführt werden kann, wurde Messung IV gemacht. Zwei Schwarzweissbilder, welche denselben Schwarz- und Weissanteil aufweisen, sich jedoch in der Farbverteilung unterscheiden, wurden dabei als Referenz verwendet (siehe Abbildung 3.12 auf Seite 39). Wie man erahnen kann, ist Bild 2 massiv einfacher zu komprimieren, als Bild 1. Der Grund dafür liegt darin, dass Bild eins weniger regelmässige Muster enthält, als Bild 2. Beide Bilder bestehen aus 800x800 Pixel.

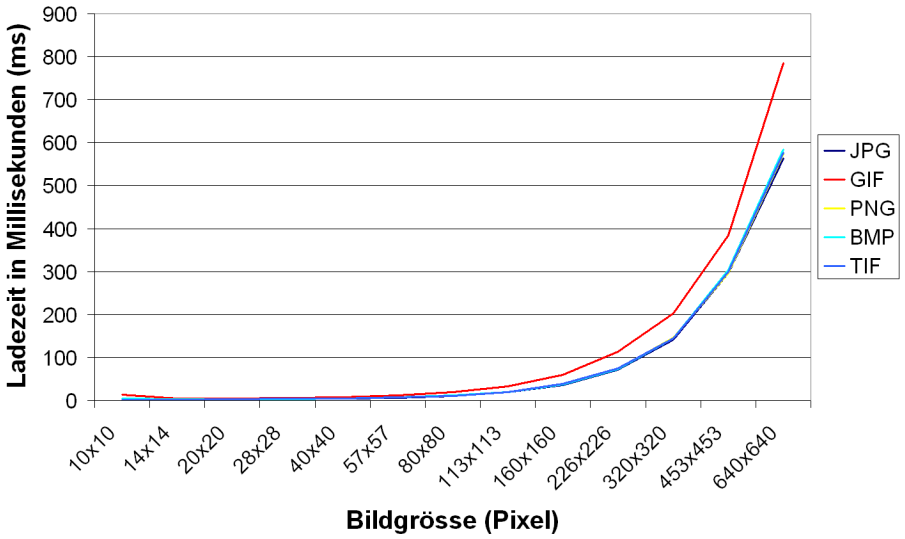

Abbildung 3.11: Entwicklung der Ladezeit im Vergleich zur Bildgrösse

Abbildung 3.13 auf Seite 40 zeigt das Resultat. Bei PNG Bildern sind die Einflüsse der Komprimierung am markantesten und bei GIF sind die Unterschiede nicht auffallend. BMP und TIF weisen wie erwartet identische Werte auf sind unabhängig von der Bildkomplexität. Dieses Experiment unterstützt die These, dass ein Grossteil der gemessenen Zeit zur Dekomprimierung gebraucht wird und dass die schwache Prozessorleistung des iPAQ diese Unterschiede noch zusätzlich vergrössern. Interessanterweise reagiert der offene Standart PNG am sensibelsten auf die unterschiedliche Komplexität. Um das Bild abzurunden, wurden nicht nur Ladetests mit SWT durchgeführt, sondern auch proprietäre Pocket PC Benchmarks berücksichtigt um Aufschluss

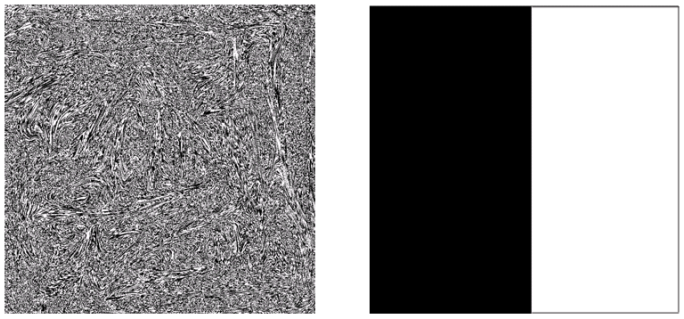

Abbildung 3.12: Ladezeiten unterschiedlich komplexer Bilder: Bild 1 (l.) ist aufwändiger zu komprimieren als Bild 2 (r.)

über die realisierbare Lesegeschwindikeit der verfügbaren Speichermedien zu erhalten.

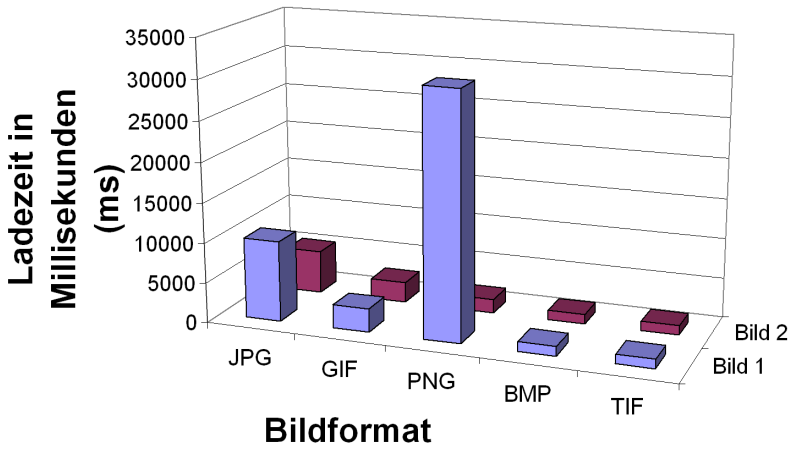

Abbildung 3.13: Ladezeitenvergleich: regelmässige Muster (Bild 2) vs. unregelmässige Muster (Bild 1)

Weitere Benchmarks

Die Idee dieser weiteren Messungen ist es, die effektive Lesegeschwindigkeit der verwendeten Speichermedien zu testen. Die Lesegeschwindigkeit wurde mit zwei verschiedenen Benchmarks[8] gemessen, welche dabei nahezu identische Resultate lieferten. Diese sehen wie folgt aus:

- **SD Card:** 1.6 MB/s (max. Lesegeschwindikeit)
- **SRAM:** 32.3 MB/s (max. Lesegeschwindikeit)

Die Resultate zeigen, dass die aus den Messungen I - IV resultierenden Geschwindigkeiten deutlich unter der theoretisch möglichen Lesegeschwindigkeit liegt.

[8] Es wurden folgende Benchmarks verwendet: Spb Benchmark (http://www.spbsoftwarehouse.com) und Pocket Mechanic Paket (http://www.antontomov.com)

Zusammenfassung

Die verschiedenen Abschnitte dieses Unterkapitels haben die Darstellungsart III, die Offline Rotation, näher beleuchtet. Zuerst wurde die theoretische Grundlage erläutert und anschliessend eine Analyse des Hauptspeicherverbrauches durchgeführt. Es wurde diagnostiziert, dass SWT Image-Objekte, unabhängig vom zugrundeliegenden Bildformat, unkomprimiert im Hauptspeicher liegen und dadurch für eine mobile Applikation verhältnismässig viel Hauptspeicher benötigen. Messung I bis IV haben die wesentlichen Einflussgrössen auf die Dauer des Ladeprozesses untersucht. Die gewonnene Erkenntnisse sind, dass neben der Bildgrösse vor allem die Bildkomplexität die Dauer dieses Prozesses bestimmen. Aufgrund des limitierten Hauptspeichers und der eigenschaften des Ladeprozesses ist eine echtzeittaugliche Implementierung der Offline Rotation zurzeit nicht praktikabel. Trotzdem bietet sich die Offline Rotation als eine im Pullmodus funktionierende Darstellungsart an. Mit der Offline Rotation wurde nun auch die letzte der drei möglichen Darstellungsarten untersucht. Im folgenden Abschnitt soll nun Fazit über die gemachten Entdeckungen gezogen werden.

3.2.4 Fazit

Aufgrund der gemachten Untersuchungen ist es nun möglich die zu Beginn gestellte Frage nach der Machbarkeit der graphischen Aufbereitung besser zu beantworten. Die in der Einführung erwähnten Echtzeitverfahren benötigen Darstellungstechniken, welche in weniger als 100ms ein Bild zeichnen. Die Darstellungsart I „Einfache Grafikoperationen“ ermöglicht eine Darstellung, die diese Laufzeit sogar noch unterschreitet. Das heisst, die Echzeittauglichkeit ist gegeben. Natürlich könnte dieses Verfahren auch im Pullmodus eingesetzt werden. Bei Darstellungsart II und Darstellungsart III wurde entdeckt, dass die Laufzeit primär von der Bildgrösse abhängt. Bei der Darstellungsart III müssen auch die unterschiedlichen Ladezeiten verschiedener Bildformate berücksichtigt werden. Darstellungsart II ist unabhängig vom Bildformat und benötigt durchschnittlich 0.0087 Millisekunden pro Pixel. Sämtliche diese Faktoren werden in der Abbildung 3.14 auf Seite 42 veranschaulicht. Das macht die Online Rotation scheinbar zum unattraktivsten Verfahren. Man sieht, dass mit der Offline Rotation Bilder bis 160x160 Pixel in Echtzeit gedreht werden können, bei der Online Rotation bis rund 100x100 Pixel.

Um damit in Echtzeit drehende Karten zu implementieren, ist diese Grenze zu niedrig. In Anlehnung an das bestehende MobileGame kann man davon ausgehen, dass die durchschnittliche Kartengrösse 624x604 beträgt. 160x160 Pixel als Limite würden die Funktionalität des Mo-

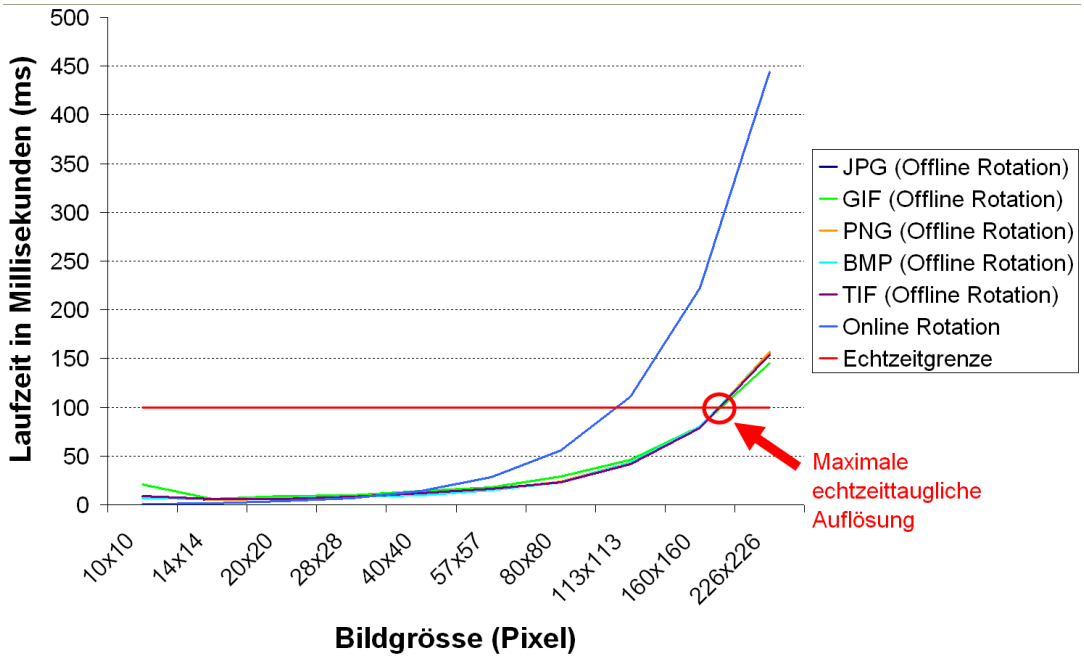

Abbildung 3.14: Laufzeit: Vergleich von Darstellungsart II und III

bileGame auf unakzeptierbare Weise einschränken. Echzeitverfahren können aber durchaus zur Anwendung kommen, nämlich dann wenn es nicht darum geht ganze Karten zu drehen sondern lediglich darum Richtungsinformationen zu vermitteln. Dabei könnte eine Option sein, dass man anzeigt, wo sich Norden befindet oder dass dem Benutzer seine aktuelle Laufrichtung auf der Karte eingezeichnet wird. In Anlehnung an die zu Beginn erläuterte Programmiersituation (vgl. Abbildung 3.2) fasst Tabelle 3.6 die Erkenntnisse der Machbarkeitsanalyse zusammen.

	Echtzeitverfahren	**Pullverfahren**
Darstellungsart I: Einfache Grafikoperationen	realisierbar	realisierbar
Darstellungsart II: Online Rotation	nicht realisierbar	realisierbar
Darstellungsart III: Offline Rotation	nicht realisierbar	realisierbar

Tabelle 3.6: Resultat der Machbarkeitanalyse

4

Implementation

Mit der Machbarkeitsanalyse wurde die Grundlage für eine Implementation erstellt. Es wurde herausgearbeitet, welche Möglichkeiten mit den gegebenen Rahmenbedingungen bestehen und wo man auf Grenzen stösst. Dies ermöglicht nun eine gezielte Implementation der gewünschten Navigationsfunktionen. Dabei wird in drei Schritten vorgegangen: Zuerst wird auf die verwendete Methodik (1) hingewiesen. Anschliessend wird die Kernfunktionalität in Form eines Prototypes erstellt (2). Dieser Prototyp soll in das bestehende MobileGame eingebaut werden (3). Das Ziel ist, dass das Endprodukt bzw. das erweiterte MobileGame über vier verschiedene Darstellungsvarianten verfügt, aus denen der Kunde dieser Arbeit, Christoph Göth und die Gruppe Informations Management, auswählen können.

4.1 Methodik

Nachdem im Kapitel 3 grundlegende Messungen der Verzögerung und des Speicherbedarfes vorgenommen wurden, kann man anhand des gezogenen Fazits entscheiden, welche Richtung man bei der Entwicklung eines funktionsfähigen Prototypes für die Interpretation und Visualisierung von Sensordaten einschlagen soll. Bereits jetzt ist klar, dass der Prototyp in das bestehende MobileGame eingebaut werden soll. Beim MobileGame handelt es sich um eine verhältnismässig umfangreiche Software. Sie besteht aus deutlich mehr als 500 Zeilen Quellcode, was laut [Glinz, 2004] die Grenze zwischen Gross- und Kleinsoftware ist. Glinz weist drauf hin, dass bei der Entwicklung und Integration von grösserer Software spezielle Gesetztesmässigkeiten herrschen, welche unbedingt zu berücksichtigen sind. So gilt besonders bei grösseren Software Projekten, dass „Immaterialität das Arbeiten mit Software schwieriger macht, als dasjenige mit materiellen tech-

nischen Produkten vergleichbarer Komplexität. Die Risiken sind schwieriger zu erkennen; ad-hoc-Vorgehensweisen führen schneller ins Desaster.“ Um dem zu begegenen wird zuerst ein voll funktionsfähiger evolutionärer Prototyp entwickelt, der unabhängig vom MobileGame ist. Erst in einem zweiten Schritt soll dieser Prototyp dann in das MobileGame eingebaut werden. Die generelle Architektur dieses Prototypes wird im kommenden Abschnitt dargelegt.

4.1.1 Ueberlegungen zur Architektur

Generell legt die vorgegebene Verwendung der Programmiersprache Java bereits eine objektorientierte Architektur nahe. Laut [Glinz, 2004] ist eine Architektur „die Organisationsstruktur eines Systems oder einer Komponente“. Gemäss der Architektur des bisherigen MobileGames, welche ebenfalls objektorientiert ist, besteht der MobileGame Client aus einer 2-Tier-Architektur [Brunner, 2005]. Das scheint jedoch nicht korrekt zu sein. In Wahrheit besteht er aus einer 3-Tier-Architektur. Neben einer Präsentations- und einer Logikschicht beinhaltet der Client ebenfalls eine Datenschicht. Viele Daten werden dem Client zwar vom Server zur Verfügung gestellt. Jedoch hält der Client auch eigene Daten die es ihm ermöglichen bei Netzwerkunterbrüchen weiter zu funktionieren (Mehr Informationen zur MobileGame Architektur findet man im Abschnitt 4.3.3 auf Seite 56). Schauen wir nun die neue Situation mit der am Client befestigten Sensorik an, stellen wir fest, dass die Sensordaten ebenfalls direkt beim Client erhoben und verarbeitet werden. Das bedeutet, der Prototyp wird als 3-Tier-Architektur erstellt. Konkret wird das Model-View-Controller (MVC) Muster angewendet. Dieses Entkopplungsmuster ist eine gute Art in einer objektorientierten Programmiersprache eine 3-Tier-Architektur umzusetzen. Denn genau darum geht es bei dieser Idee, um Entkopplung. Jede der Schichten bzw. der Komponenten soll möglichst einfach austauschbar sein. Nachdem die generelle Methodik definiert war, wurde die Kernfunktionalität in Form eines evolutionären Prototypes erstellt. Das nächste Unterkapitel eräutert die Funktionen und die Struktur dieses Prototypes.

4.2 Kernfunktionalität

Das Ziel dieses Unterkapitels ist, näher auf den erstellten Prototypen einzugehen. Dabei werden zuerst die erstellen Java Klassen erläutert und danach die angewendeten Konzepte, insbesondere die verwendeten Design Pattern, aufgezeigt.

4.2.1 Klassenübersicht

Verwendet wurde ein System von acht Klassen welche nach dem Geheimnisprinzip gekapselt und soweit wie möglich funktional in sich geschlossen sind. Tabelle 4.1 beschreibt deren Funktionen. Der Quellcode sämtlicher Klassen befindet sich auf der beiliegenden CD-ROM (Anhang E auf Seite 104).

Klasse	Beschreibung
`SensorReader`	Die Klasse `SensorReader` greift via seriellen Port auf das Sensorboard zu, steuert dieses und verarbeitet die erhaltenen Daten
`ListenerStore`	Beim `ListenerStore` können sich andere Klassen registrieren, welche bei einem neuen Sensorinput informiert werden möchten. Das eignet sich vor allem um ein GUI aktuell zu halten.
`RecordStore`	Der `RecordStore` verwaltet die erhaltenen Daten. Jeder erhaltene Sensordatensatz wird in einem `Record` gespeichert. Der `RecordStore` verwaltet sämtliche Records und ermöglicht es Auswertungen über die bisher erhaltenen Daten zu machen.
`Record`	Ein `Record` repräsentiert einen vom Sensor erhaltenen Datensatz.
`SensorDemonstration`	Die zentrale GUI Klasse des hier präsentierten Prototyps. Sie erlaubt es dem Benutzer veschiedene Einstellungen vorzunehmen und so den Prototypen zu steuern.
`CalibrationShell`	Die `CalibrationShell` ermöglicht es den Sensor in den Kalibrierungsmodus zu setzten und zeigt dem Benutzer die nötigen Informationen während der Kalibrierung an.
`CompassShell`	Der Prototyp hat zwei Möglichkeiten um die Kompassdaten zu visualisieren. Die `CompassShell` ist eine davon. Sie zeigt dem Benutzer an in welcher Richtung sich Norden befindet.
`VectorShell`	Der Prototyp verfügt über zwei Möglichkeiten um die Kompassdaten zu visualisieren. Die `VectorShell` repräsentiert die Sensordaten als zweidimensionalen Vektor in einem Koordinatensystem

Tabelle 4.1: Klassenbeschreibung des Prototyps

Um einer späteren Generation von Programmierer ein einfacheres Verständnis des Quellcodes und Systemaufbaues zu ermöglichen, ist es kommun, dass man bei der Programmierung Design Patterns anwendet. Drei davon werden nachfolgend etwas näher beleuchtet um allfällige Weiterentwicklungen zu vereinfachen. Bereits im Unterkapitel 4.1.1 wurde erwähnt, dass eine ausreichende Entkoppelung der einzelnen Schichten mittels einem Model-View-Controller (MVC) Pattern erreicht werden kann. Darüberhinaus wurde auch das Observer[1] Pattern verwendet. Für die beiden Patterns wird nachfolgend deren generelle Idee sowie für diese Arbeit spezifischen Anwendungsbereich erklärt.

4.2.2 Design Pattern I: Model View Controller

Beim MVC Pattern wird ein System in die drei Bereiche Daten (Model), Steuerung (Controller) und visuelle Aufbereitung (View) aufgeteilt. Die **Idee** ist eine Flexibilisierung des Systemes um allfällige Änderungen in der Zukunft zu erleichtern und einzelne Module des Systems wiederverwendbar zu machen. Da es sich hier um einen evolutionären Prototyp handelt, ist dieses Design Pattern hervorragend geeignet, um eine spätere Weiterentwicklung zu unterstützen. Die drei Teile des Patterns übernehmen folgende Funktionen:

- Das **Model** enthält die Daten, welche dargestellt werden sollen. Im konkreten Fall sind das die Sensordaten, welche durch die Sensorik erhoben werden. Es ist in sich geschlossen und bietet Methoden an, um auf die gespeicherten Daten zuzugreifen.
- Die **View** entspricht der Präsentation der Daten. Bei dem erstellten Prototyp wurden zwei visualisierungs Arten erstellt. Eine Vektor Visualisierung und eine kompassähnliche Visualisierung. Beide basieren jedoch auf dem selben Model.
- Der **Controller** ist für die Steuerung des Systems zuständig. Er steuert folglich den Sensor und weist das Model an, die erhaltenen Daten in geeigneter Weise zu speichern. Der Controler nimmt ebenfalls Befehle vom Benutzer entgegen. So zum Beispiel die Aktivierung des Kalibrierungsmodus.

Abbildung 4.1 macht die konkrete Aufteilung in die drei Bereiche des Architekturmusters klar. Man sieht, dass der Controller das Herzstück darstellt. Nur er ist mit den jeweiligen anderen Bereichen (Model und View) verbunden. Das bedeutet, dass die drei „Shell“ Klassen für die Visualisierung verwendet werden und nach belieben ersetzt werden können. Das soll später eine ein-

[1] Auch unter dem Namen Publish-Subscribe Pattern bekannt.

fache Integration ins MobileGame ermöglichen. Der Controller besteht aus dem `SensorReader` und dem `ListenerStore`. Der `SensorReader` bietet Methoden zur Steuerung des Sensors, nimmt die von ihm gelieferten Daten entgegen, interpretiert diese und übergibt sie dann dem Model zur Speicherung. Ändert sich das Model, wird ein `PropertyChangeEvent` ausgelöst und die beim `ListenerStore` registrierten Shells werden über die Änderung informiert.

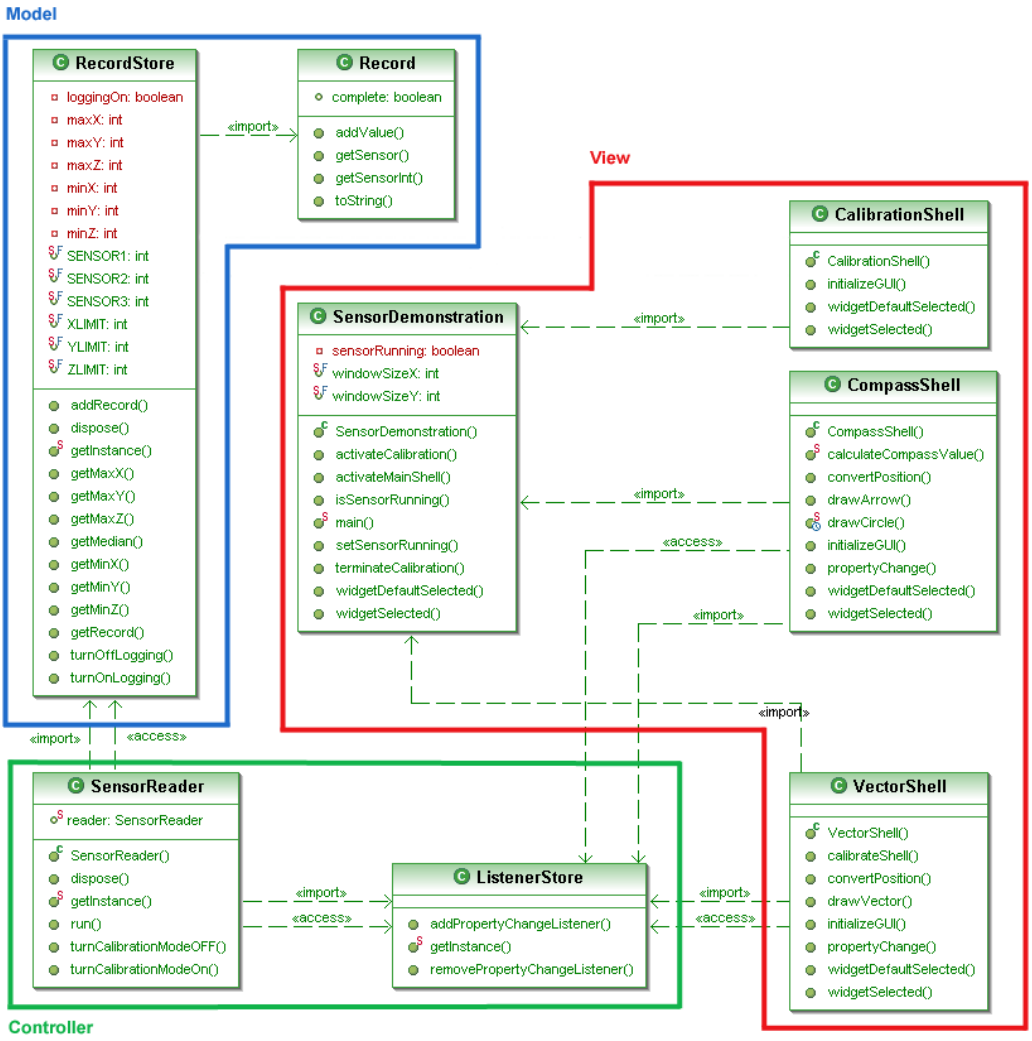

Abbildung 4.1: UML Diagramm des erstellten Prototyps

4.2.3 Design Pattern II: Observer Pattern

Die **Idee** des Observer Patterns ist, Statusänderungen eines Objektes an andere Objekte (Observer Objekte) weiterzugeben. Das Objekt dessen Status überwacht werden soll, muss dabei selbst nicht über den Aufbau der Observer-Objekte Bescheid wissen. Das ermöglicht eine lose Koppelung zwischen den einzelnen Objekten. Damit das funktioniert, müssen sich die Observer-Objekte beim observierten Objekt registrieren. So erhalten sie allfällige Statusänderungen in Form sogenannter Events. Beim vorliegenden Anwendungsfall, dem Prototyp zur Sensordatenvisualisierung, wird dieses Pattern wie folgt angewendet: Die Objekte, welche Änderungen am Zustand des Sensors beobachten wollen (also die Observer Objekte), müssen sich beim `ListenerStore` registrieren. Dieser verwaltet die Beobachter und gibt ihnen bei Änderungen Bescheid. Die Beobachter sind in diesem Fall die Objekte der Präsentationsschicht, nämlich `VectorShell` und `CompassShell`. Beide greifen auf die selben Daten zu, stellen sie aber unterschiedlich dar. Die Objekte der Präsentationsschicht können jederzeit ausgetauscht werden, ohne das dabei die Struktur der restlichen Schichten verändert werden muss. Das ist der Vorteil dieses Design Patterns. Damit soll eine möglichst einfache Integration in das MobileGame ermöglicht werden. Das ist auch bereits das Thema des nächsten Kapitels.

4.3 Integration in das MobileGame

Die Kernfunktionalitäten wurden nun in Form eines evolutionären Prototypes entwickelt und stehen für eine Integration in das MobileGame bereit. Erweiterungen bestehender Software sind oftmals prekäre Unternehmen. Es ist von zentraler Bedeutung, dass der Software Architekt die bereits bestehende Architekur verstanden hat und seine Erweiterung so integriert, dass die dahinterstehende Logik erhalten bleibt. Das ermöglicht auch in der Zukunft eine saubere Weiterentwicklung der Software. Das vorliegende Unterkapitel geht deshalb vom Konzeptuellen ins Technische. Auf konzeptueller Ebene werden zuerst die Idee des MobileGames und die neue Funktionalität behandelt. Danach wird die Architektur und technische Integration betrachtet.

4.3.1 Die Idee des MobileGames

Das in dieser Arbeit weiterentwickelte MobileGame ist ein Software System, das am Institut für Informatik (IFI) der Universität Zürich[2] über Jahre hinweg entwickelt wurde. Es wurde ständig erweitert und verbessert und einige Male sogar komplett neu entwickelt. Die zum Zeitpunkt dieser Arbeit vorliegende Version des MobileGames beruht auf der Architektur von Dennis Brunner und wurde im Jahr 2005 erstellt. Die Idee des MobileGames ist es, dem Benutzer auf spielerische Art und Weise eine ihm unbekannte Umgebung näher zu bringen. Das Konzept des MobileGames sieht vor, dass mehrere Teams, mit MobileGame PDAs ausgerüstet, in einer unbekannten Umgebung verschiedene Aufgaben erfüllen müssen. Gleichzeitg sollen die Teams untereinander interagieren. Dazu gibt es einen „Hunting Mode“ in welchem jedes Team ein anderes verfolgen muss, gleichzeit aber auch verfolgt wird. Das Software System versorgt die Teams mit ortsbezogenen Informationen wie ihrer aktuellen Position, jener von den konkurrenzierenden Teams und kontextbezogenen Annotationen. Der interessierte Leser findet detailiertere Informationen über die bereits bestehende Funktionalität des MobileGames in der Arbeit von Dennis Brunner [Brunner, 2005]. Der Fokus des nächsten Abschnittes liegt auf den neu entwickelten Funktionen. Im Kapitel 3.2 wurde die Machbarkeit verschiedener Darstellungsarten analysiert. Ein erneuter Blick auf die Tabelle 3.6 auf Seite 42 zeigt, welche Darstellungsarten als realisierbar eingestuft wurden. Aus diesen Ergebnissen werden verschiedene Funktionen abgeleitet und in das MobileGame integriert. Der nächste Abschnitt gibt einen generellen Überblick über diese neu implementierte Funktionalität. Die technische Implementation wird zu einem späteren Punkt behandelt.

4.3.2 Neue Funktionalitäten

Das MobileGame wurde mit vier neuen Look & Feel Varianten ausgestattet (hier als „Design Alternativen“ bezeichnet) welche die Navigation mittels Kompassinformation ermöglichen sollen. Zuerst werden diese aus der durchgeführten Machbarkeitsanalyse (Kapitel 3) hergeleitet. Danach werden die Details der einzelnen Design Alternativen erläutert.

Bezugnehmend auf Tabelle 3.6 auf Seite 42 werden diese aus der Darstellungsart I und Darstellungsart II abgeleitet. Darstellungsart III wurde aus verschiedenen Gründen nicht realisiert. Der Hauptgrund ist die zu starke Belastung des Hauptspeichers. Man kann besagter Tabelle 3.6 ent-

[2] Das MobileGame wurde von der Forschungsgruppe Information Management entwickelt (http://www.ifi.unizh.ch/im/imrg/)

nehmen, dass Darstellungsart I als Pull- und als Echtzeitverfahren zur Verfügung steht, während Darstellungsart II lediglich im Pullmodus realisierbar ist. Tabelle 4.2 verdeutlicht nun, was das für die verschiedenen Design Alternativen bedeutet (der genaue Inhalt der einzelnen Design Alternativen wird in den folgenden Abschnitten erklärt). Design Alternative I wurde direkt aus der Darstellungsart II, nämlich der Online Rotation abgeleitet. Das bedeutet, dass Design Alternative I nur im Echzeitmodus zur Verfügung steht (Der Grund dafür kann der geneigte Leser dem Kapitel 3.2 entnehmen). Design Alternative II bis IV werden durch die Darstellungsart I „Einfache Graphikoperationen“ realisiert. Sie stehen deshalb sowohl im Pull- als auch im Echtzeitmodus zur Verfügung.

Bezeichnung	**Verfübare Modi**	
	Echtzeit	**Pull**
Design Alternative I (Karten Rotation)		x
Design Alternative II (Kleiner Pfeil)	x	x
Design Alternative III (Grosser Pfeil)	x	x
Design Alternative IV (Zentrierter Pfeil)	x	x

Tabelle 4.2: Die erstellten Design Alternativen im Überblick

Neben den vier Design Alternativen, aus welchen der MobileGame-Verantwortliche Christoph Göth für das Experiment auswählen können wird, gibt es auch eine Reihe von zusätzlichen Funktionen. Diese werden im Abschnitt „Weitere Funktionen“ näher erläutert.

Design Alternative I: Karten Rotation

Die hier beschriebene Design Alternative dreht auf Benutzerwunsch die Karte und nordet sie. Die **Idee** ist es, dem Benutzer eine möglichst intuitive Darstellung seiner Orientierung zu bieten. Deshalb greift sie den in der Machbarkeitsanalyse herausgearbeiteten Ansatz der „Online Rotation“ (siehe Abschnitt 3.2.2) auf. Wie festgestellt wurde, ist die Online Rotation nicht echtzeittauglich und kann deshalb nur im Pullmodus zur Verfügung gestellt werden. Konkret wurde das MobileGame derart abgeändert, dass es dem Benutzer möglich ist, auf Knopfdruck, die Karte so zu drehen, dass sie automatisch genordet ist. Der **Vorteil** davon ist, dass eine gedrehte Karte sehr intuitiv ist und dem Benutzer so das gedankliche Drehen der Karte im Kopf, oder das Drehen des Gerätes erspart. Der **Nachteil** ist jedoch, dass für die Rotation rund drei bis vier Sekunden gebraucht wird. Das ist lange und verzögert den Spielfluss.

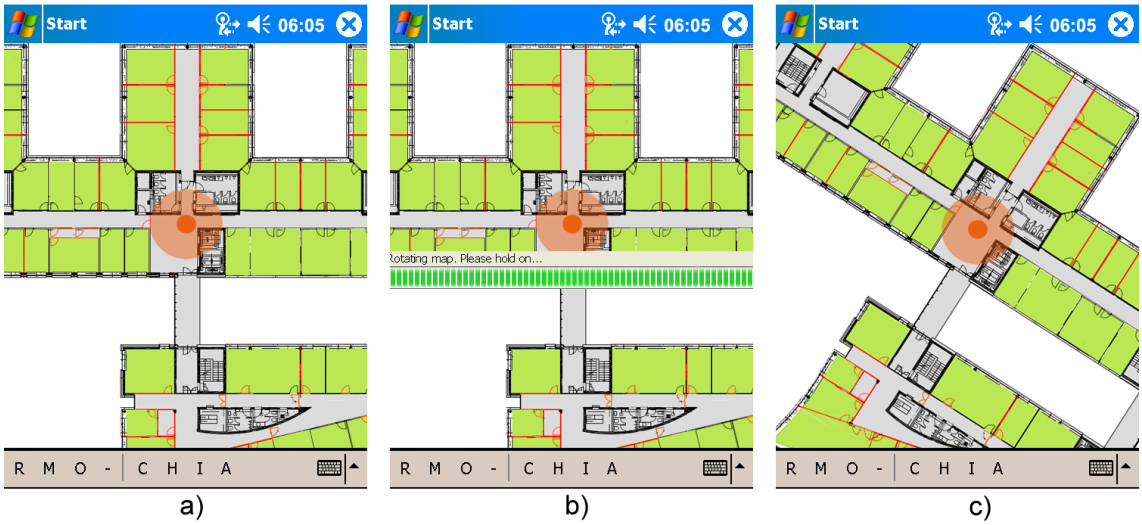

Abbildung 4.2: Screenshots der Design Alternative I: a) Ausgangslage b) Rotation wird berechnet c) Rotierte Karte

Design Alternative II: Kleiner Pfeil

Die hier beschriebene Design Alternative indiziert die Blickrichtung des Benutzers[3] in der Form eines kleinen Pfeiles der am rechten oberen Rand über die Karte gezeichnet wird. Die Idee davon ist, dass die Karte nach wie vor das dominierende Element auf dem Display ist. Der **Vorteil** ist, dass der Pfeil nur einen kleinen Teil der Karte überdeckt. Auf der anderen Seite existiert der **Nachteil**, dass diese Design Alternative weniger intuitiv erscheint.

Abbildung 4.3: Screenshots der Design Alternative II: Ein kleiner Pfeil zeigt die Blickrichtung an

[3]Es wird davon ausgegangen, dass der Benutzer den PDA ebenfalls in die Richtung seines Blickes hält.

Design Alternative III: Grosser Pfeil

Design Alternative III stellt dem Benutzer einen grösseren transparenten Pfeil zur Verfügung, der in der Mitte über der Karte platziert ist. Die **Idee** ist, dass der Pfeil dominiert und nicht mehr die Karte. Der **Vorteil** ist, dass die Karte trotzdem noch wahrgenommen werden kann. Der **Nachteil** ist, dass mehr von der Karte verdeckt wird, als bei der vorherigen Design Alternative.

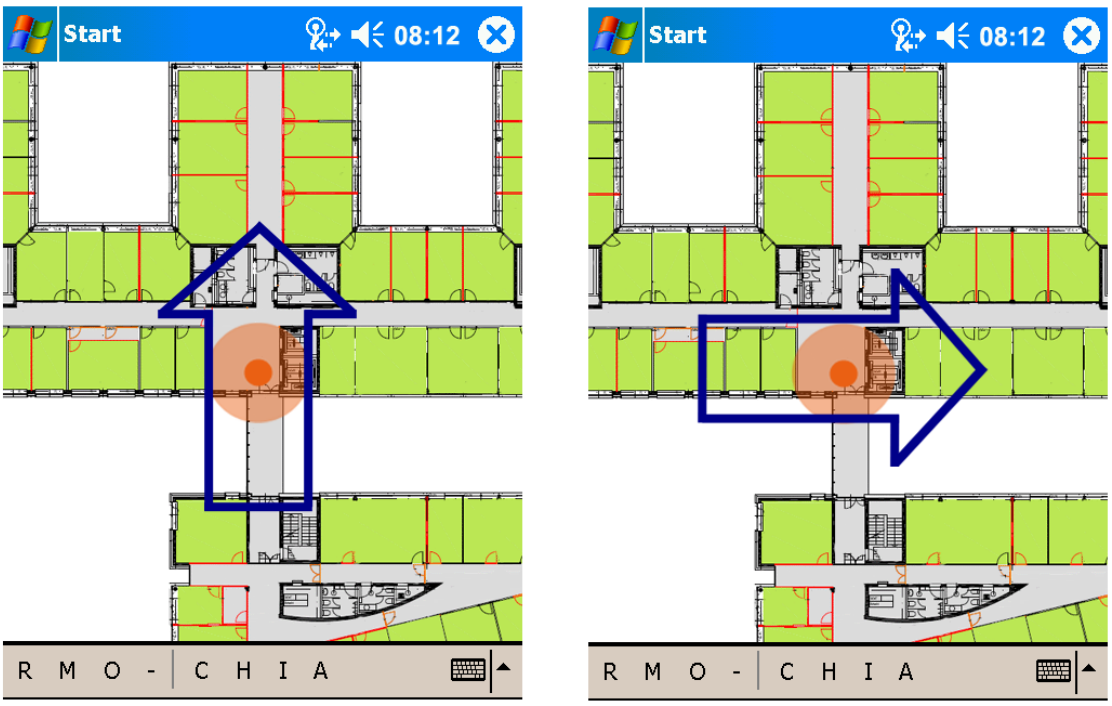

Abbildung 4.4: Screenshots der Design Alternative III: Ein grosser transparenter Pfeil zeigt die Blickrichtung an

Design Alternative IV: Zentrierter Pfeil

Bei Design Alternative IV wird ein Pfeil genau dort über der Karte platziert, wo die aktuelle Benutzerposition ist. Das bedeutet konkret, dass der Pfeil immer vom roten Punkt aus geht, der den aktuellen Standort symbolisiert. Der Pfeil soll auch hier die aktuelle Blickrichtung des Benutzers symbolisieren. Die **Idee** ist, dass es für den Benutzer möglichst intuitiv sein soll. Der **Vorteil** ist, dass der Benutzer mit einem Blick auf sein Positionssymbol auch gleichzeitig seine Blickrichtung ablesen kann. Ein **Nachteil** könnte sein, dass es Benutzer gibt, die die Bedeutung des Pfeiles falsch interpretieren.

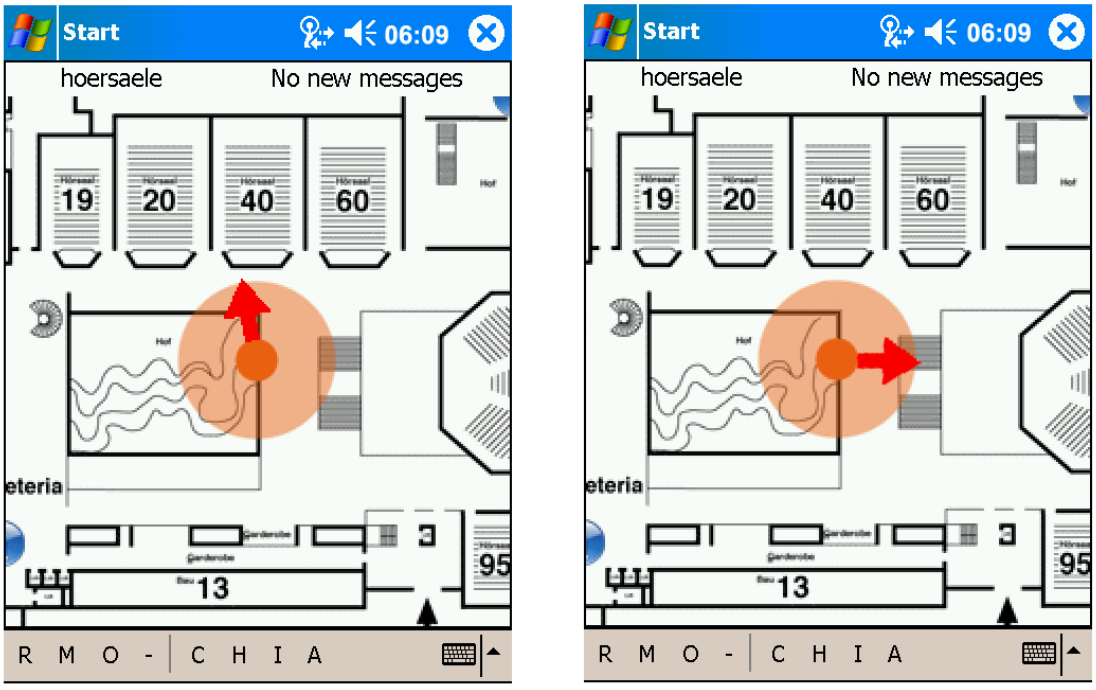

Abbildung 4.5: Screenshots der Design Alternative IV: Positionsanzeige mit integrierter Blickrichtungsanzeige

Weitere Funktionen

Die oben beschriebene Integration von Navigationsinformationen in das bestehende MobileGame zieht zusätzliche Funktionen nach sich, welche ebenfalls implementiert wurden und an dieser Stelle kurz beschrieben werden. Diese weiteren Funktionen stehen bei allen vier Design Alternativen zur Verfügung.

- **Karten Nordung:** Der Server wurde erweitert, so dass der Spielleiter bei der Erstellung eines Spieles für jede Karte bestimmen muss, wo Norden liegt. Dies ist nötig, da leider die verwendeten Karten selbst nicht genordet sind. Je nach verwendeter Karte teilt der Server dem Client die zur Karte relative Nordrichtung mit.
- **Modus Auswahl:** Der Spielleiter kann neu darüber entscheiden, ob der Sensor im Echtzeit- oder Pullmodus verwendet werden soll, oder diesen auch komlpett ausschalten. Es ist sogar möglich, das für jeden Client individuell zu bestimmen, so dass ein Spiel mit gemischten Clients gespielt werden kann.
- **Logging:** Das verwendete Sensorboard stellt bekanntlich nicht nur Magnetfeldsensordaten zur Verfügung sondern noch eine Reihe von weiteren Sensorinformationen. Diese werden während des Spiels automatisch aufgezeichnet. Das ermöglicht im Anschluss zusätzliche Auswertungen in Form von Datamining.
- **Kalibrierung:** Aufgrund der Tatsache, dass die verschiedenen Sensorboards unterschiedliche Offsets verwenden, musste eine für jeden Client individuelle Kalibrierung ermöglicht werden. Zusätzlich wurde das MobileGame um eine einfache Software erweitert, die den Offset der Magnetfeldsensoren berechnet. Das Gerät muss dazu während ca. 20 Sekunden in alle Himmelsrichtungen bewegt werden.
- **Simulationsmodus:** Der MobileGame Client verfügt auch über einen Simulationsmodus, der einen Magnetfeldsensor simuliert. Das kann bei weiterführenden Änderungen oder Tests sehr hilfreich sein, sollte einmal kein Sensorboard zur Verfügung steht.

Nachdem nun die Erweiterung der Funktionalität erklärt worden ist, soll der interessierte Leser in einem weiteren Schritt mehr über die konzeptuellen und technischen Details des MobileGames und seiner Erweiterung erfahren. Zuerst wird die Architektur des MobileGames erwähnt und basierend darauf, in einem zweiten Schritt, die technische Integration der oben beschriebenen Funktionen erklärt.

4.3.3 Die Architektur

Wie bereits erwähnt, wurde das MobileGame System in Form einer einer Client-Server Architektur realisiert. Es besteht aus einer 3-Tier-Architektur (siehe auch Abschnitt 4.1.1) die zwischen Präsentations-, Logik- und Datenschicht unterscheidet. Das Softwaresystem besteht jedoch nicht nur aus Client und Server sondern aus einer Reihe von Komponenten welche zusammen interagieren und so ein verteiltes System bilden. Diese Erkenntnis führt zu zwei möglichen Perspektiven auf die Softwarearchitektur des MobileGames: Zum einen eine Betrachtung nach Schichten (**Schichtensicht**) und zum anderen eine Aufteilung nach physisch unabhängigen Systemkomponenten (**Verteilungssicht**). Die nachfolgenden zwei Abschnitte fokussieren auf diese beiden Betrachtungsweisen und erläutern sie näher.

Verteilungssicht

Physikalisch ist die Architektur auf drei Komponenten verteilt: Den Server (beinhaltet Ekahau[4]-sowie MobileGame Server), den iPAQ und den Ekahau-Tag[4]. Ein Ekahau-Tag ist ein eigenständiges mobiles Gerät, das mit dem Ekahau Server über eine WLAN-Verbindung kommuniziert.

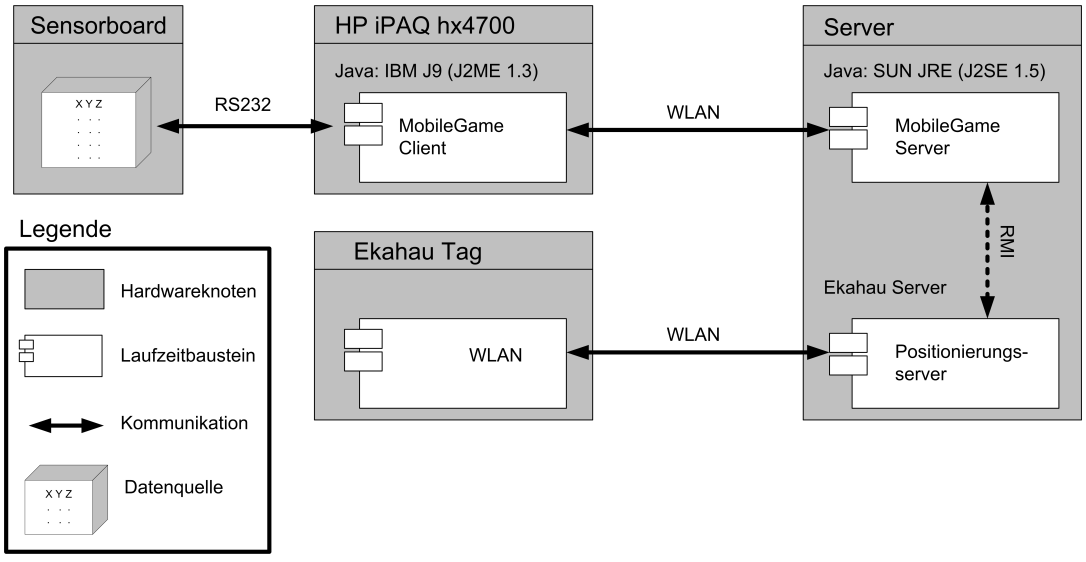

Abbildung 4.6: Verteilungssicht des MobileGame Systems (nach [Brunner, 2005])

[4]Das MobileGame System verwendet WLAN zur Positionsbestimmung. Ekahau ist das Produkt eines Drittherstellers, das diese Funktionalität realisiert. Mehr Informationen unter: http://www.ekahau.com.

Durch verschiedene Messungen, unter anderem der Signalstärke, kann der Ekahau-Server die aktuelle Position auf des Ekahau-Tags bestimmen. Der Ekahau Server übergibt diese via einer definierten Schnittstelle an den MobileGame Server, welcher die Position an die Clients übermittelt. Zusammengefasst bedeutet das, dass die eigentliche Positionierung und die restlichen Funktionen des Systems getrennt ablaufen. Abbildung 4.6 veranschaulicht diese Architektur.

Schichtensicht

Wie erwähnt, basiert das MobileGame auf einer 3-Tier Architektur. Das spiegelt sich auch im Quellcode wieder. Die einzelnen Schichten sind sorgfältig gekapselt und vereinen in sich die entsprechende Funktionalität. Die Kommunikation unter den einzelnen Schichten wurde über das Observer-Pattern, d.h. also über ein Listener/Event-Model realisiert. Die saubere Schichtentrennung und das Listener/Event-Modell ermöglicht eine nahtlose Integration des entwickelten Prototypes in die Applikation. Abbildung 4.7 verdeutlicht dies modelhaft. Eventauslösung und Callback Anfragen sind mit Pfeilen dargestellt. Die Kommunikation über das Netzwerk erfolgt mittels Serialisierung und Marshalling von Objekten[5].

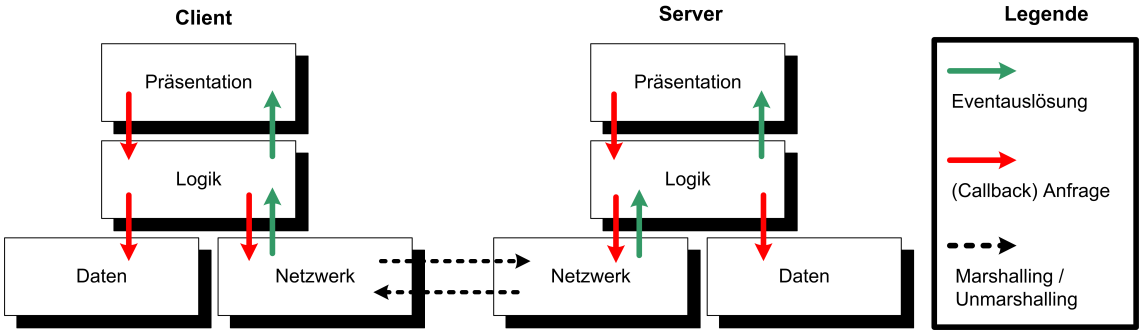

Abbildung 4.7: Schichten und Events im MobileGame System (nach [Brunner, 2005])

An dieser Stelle erkennt man einen strukturellen Unterschied des erweiterten MobileGames zum bereits bestehenden. Zuvor wurden die Daten zentral vom Server aus kontrolliert und verteilt. So wurden zum Beispiel sämtliche Positionierungsdaten und die Konfiguration des Spieles vom Server verwaltet und den Clients über das Netzwerk zugesandt. Neu ist, dass der Client über einen Sensor verfügt, der ihm kontextspezifische Informationen zur Verfügung stellt, welche für den Server nicht relevant sind. Diese Sensorinformationen werden folglich nur vom Client ver-

[5]Marshalling: Serialisierte Daten die über das Netzwerk empfangen worden sind, werden in Laufzeitobjekte übersetzt und können danach wie lokal erstellte Objekte verwendet werden.

waltet. Zusammengefasst kann man sagen, dass in der neuen Version des Clients eine wichtige Änderung in der Datenschicht vorgenommen werden musste. Nachdem nun eine eher konzeptionelles Bild des MobileGame Systems vermittelt wurde, wird in einem nächsten Abschnitt mehr auf die technische Integration fokussiert. Es soll also aufgezeigt werden, an welcher Stelle der zuvor entwickelte Prototyp integriert und an bestehende Systemstrukturen angebunden worden ist. Dabei beschränkt sich diese Arbeit, die Änderungen am Client auf zu zeigen[6]

4.3.4 Technische Integration

Um die technischen Aspekte sinnvoll zu strukturieren, wird die im vorherigen Abschnitt erklärte Unterteilung des MobileGames in drei Schichten an dieser Stelle wieder aufgegriffen und präzisiert. Javaklassen können bekanntlich zur logischen Gliederung in Pakete[7] zusammengefasst werden. So wurde das auch beim MobileGame getan. Man findet eine Reihe von Paketen vor, die sich wiederum einer spezifischen Schicht zuordnen lassen. Abbildung 4.8 stellt die wichtigsten Pakete des MobileGame Clients dar und ordnet sie den entsprechenden Schichten zu. Die Abbildung zeigt zudem auch, wie viel an den einzelnen Paketen im Rahmen dieser Arbeit geändert bzw. erweitert werden musste. Pakete die einen Wert von 100% aufweisen wurden neu hinzugefügt. Davon gibt es zwei: `engine.sensor.logic` und `engine.sensor.data`. Die nächsten beiden Abschnitte gehen näher auf die Präsentationsschicht und auf die Logikschicht ein.

[6] Am MobileGame Server wurden lediglich Details geändert, welche in keinem direkten Zusammenhang mit der verwendeten Sensorik oder der Darstellung von Navigationsinformationen stehen. Der geneigte Leser findet auf der beiligenden CD-ROM (Seite 104) den Quellcode sowie die API Dokumentation (Javadoc) des MobileGame Systems

[7] Oft wird auch das Englische Wort „Packages“ verwendet.

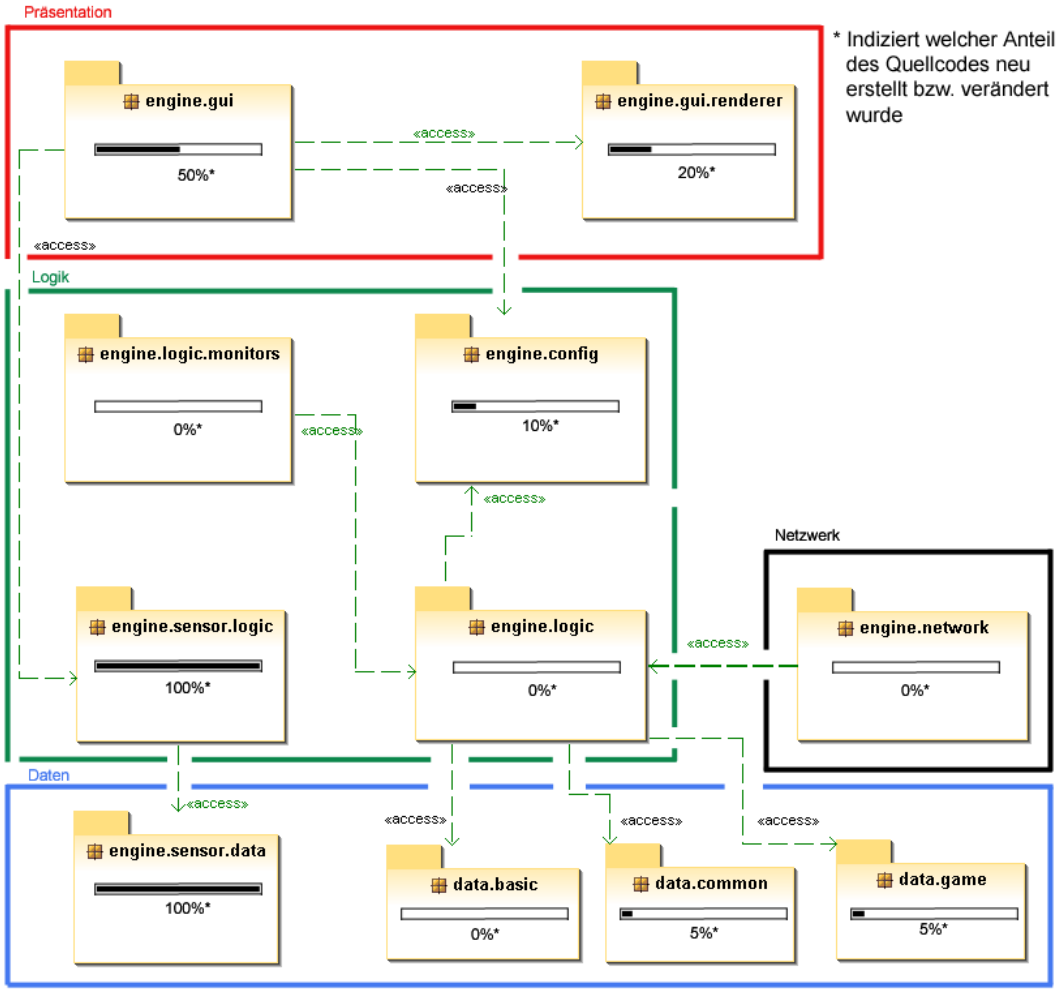

Abbildung 4.8: MobileGame Client: Übersicht über Pakete und Schichten

Präsentationsschicht

Bereits in der Abbildung 4.8 sieht man, dass ein Grossteil der gemachten Änderungen die Präsentationsschicht betroffen hat. An dieser Stelle soll deshalb näher auf diese Schicht eingegangen werden. Dazu gibt einem das UML Diagramm in Abbildung 4.9 eine Übersicht über die wichtigsten Komponenten der Präsentationsschicht. Um das Zusammenspiel dieser Komponenten zu verdeutlichen wird zwischen zwei Systemphasen unterschieden, der **Startup Phase** und der **Running Phase**. Zuerst zur **Startup Phase**. Diese besteht aus vier Schritten:

1. **Modus bestimmen:** Die erste Komponente der Präsentationsschicht, die aktiv wird, ist die Klasse `NGApplicationController`. Mit einem Zugriff auf die Logikschicht, genaugenommen auf das Paket `engine.config`, liest sie die aktuelle Konfiguration des MobileGame Systems aus. Diese gibt der Präsentationsschicht vor, in welchem Betriebsmodus sie zu starten hat. Alle weiteren Instanzierungen der Präsentationsschicht werden von der Klasse `NGApplicationController` vorgenommen

2. **Instanzierung und Konfiguration des `NGMapCanvas`:** Der `NGMapCanvas` muss, unabhängig vom Betriebsmodus, immer instanziert werden. Je nach Modus wird die Instanz jedoch anders konfiguriert.

3. **Instanzierung und Konfiguration der `SensorShell`:** Wurde einer der Pfeilmodi gewählt, wird zusätzlich auch noch eine Instanz der Klasse `SensorShell` erzeugt und im Anschluss daran konfiguriert.

4. **Registrierung beim `ListenerStore`:** In einem vierten Schritt werden die neu erzeugten Instanzen beim `ListenerStore` registriert. Das geschieht über einen Zugriff auf die Logikschicht, konkret auf das Paket `engine.sensor.logic`.

Nachdem die Instanzen der Präsentationsschicht konfiguriert und beim `ListenerStore` registriert sind, ist das System betriebsbereit. Während der **Running Phase** werden bei jeder Änderung des Sensors alle registrierten Listeners benachrichtigt. Auf diese Weise wissen die zuständigen Instanzen der Präsentationsschicht immer Bescheid über den aktuellen Status des Sensors und können diesen in definierter Form darstellen. Eine Besonderheit der Präsentationsschicht ist, die Klasse `SensorShell`. Sie wird für die Pfeilmodi bzw. Design Alternative II-IV verwendet (vergleiche Abschnitt 4.3.2). SensorShell repräsentiert eine Shell, was bei SWT einem Fenster entspricht. SWT bietet jedoch die Möglichkeit, die dargestellte Fläche eines Fensters pixelgenau zu definieren. Das eröffnet einem die Möglichkeiten Fenster mit beliebigen Formen darzu-

stellen. Im Falle vom MobileGame Client wird diese Funktion so verwendet, dass dem Fenster die Form eines Pfeiles gegeben wird. Dieses Fenster in Pfeilform wird dann über das bestehende GUI des MobileGame gelegt. So ensteht der Eindruck, dass der Pfeil auf der Karte des MobileGames liegt. Bei neuen Sensorinformationen muss die Form des Fensters wieder neu berechnet werden, da der Pfeil in eine neue Richtung zeigt. Trotz dieser eher aufwändigen Neuberechnung ist diese Methode der Visualisierung echtzeittauglich. Der grosse Vorteil dieser Methode liegt darin, dass der Pfeil sehr lose mit dem restlichen System gekoppelt ist. Es mussten also, für die Design Alternativen mit Pfeil keine grossen Eingriffe in die bestehende Präsentationsschicht vorgenommen werden.

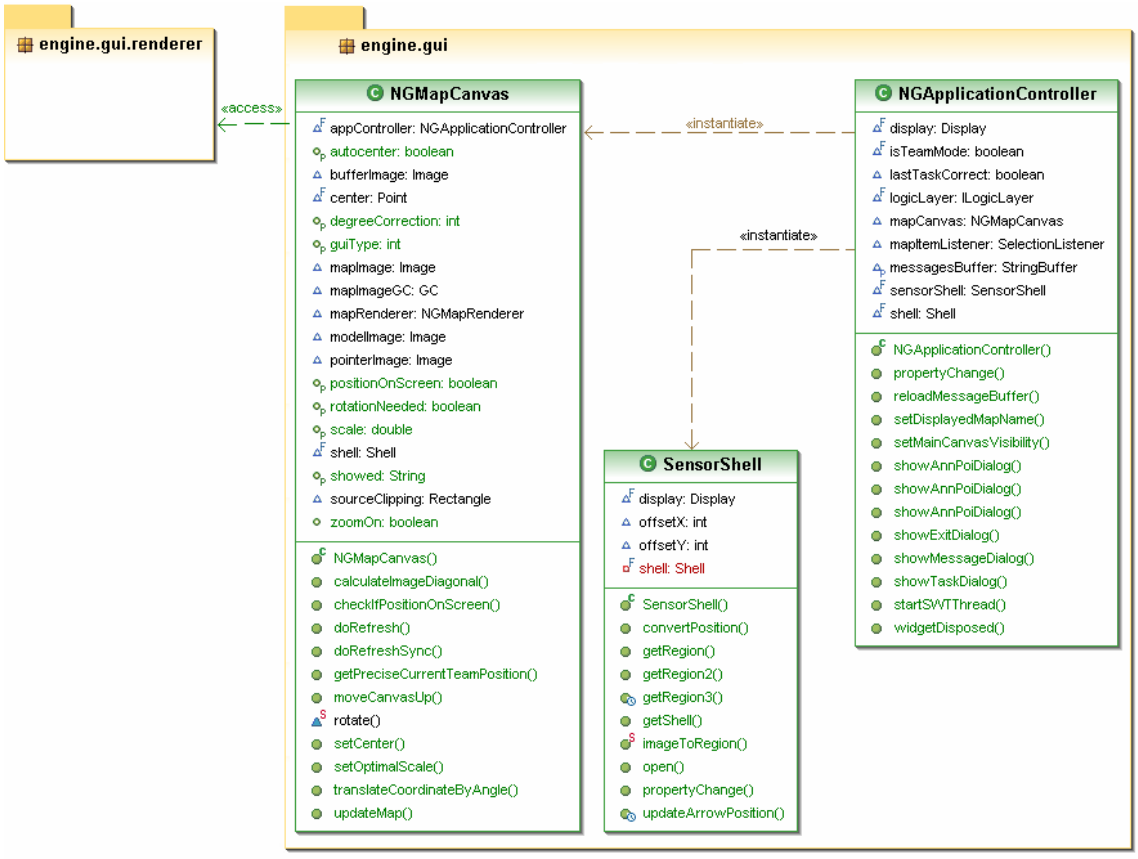

Abbildung 4.9: MobileGame Client: Auszug aus der Präsentationsschicht

Neben den bereits erwähnten Komponenten der Präsentationsschicht existiert auch das Paket `engine.gui.renderer`. Dieses Paket ist dafür zuständig, bei Positionsänderungen sämtliche Icons (eigene Position, fremde Teams, Tasks, Annotations, etc.) korrekt auf der Karte zu positio-

nieren. Das Paket musste deshalb dahingehend geändert werden, dass, wenn Design Alternative I (Map Rotation) aktiviert ist, die Drehung der Karte bei der Positionierung der Symbole miteinbezogen wird. Der technisch interessierte Leser soll an dieser Stelle auf die der Arbeit beiliegende CD-ROM verwiesen werden, auf welcher auch der Quellcode und die API Dokumentation des MobileGame Systems zu finden ist. Nachdem nun die Präsentationsschicht eingehend beleuchtet wurde, gibt der nächste Abschnitt einen Einblick in den Aufbau der Logikschicht.

Logikschicht

Bereits Abbildung 4.8 gibt einem einen Eindruck welche Pakete im Client des MobileGame System involviert sind und auf welchen Schichten die Pakete angesiedelt sind. In diesem Abschnitt soll nun der Detailierungsgrad noch ein bisschen erhöht werden. Der Fokus liegt, im Gegensatz zum vorhergehenden Abschnitt, nicht auf der Präsentationsschicht sondern auf der darunterliegenden Logikschicht. Genaugenommen wird die grösste Änderung der Logikschicht betrachtet, nämlich die Einführung des neuen Paketes `engine.sensor.logic`. Dieses kapselt zum einen sämtliche für die Steuerung des Sensors notwendigen Klassen. Zum anderen verwaltet es sämtliche registrierten Listeners (siehe auch Abschnitt „Client: Präsentationsschicht“). Zum Verwalten der Sensordaten greifen die Instanzen der Logikschicht direkt auf das der Datenschicht angehörende Paket `engine.sensor.data` zu. Die sensorspezifischen Funktionen welche die Logikschicht den höheren Schichten zur Verfügung stellt, werden in Tabelle 4.3 zusammengefasst.

Funktion	**Beschreibung**
Regelmässiger Sensorreset	Über die Zeit verändert sich die Polarisierung der Sensorelemente für die elektromagnetische Feldstärkemessung. Das kann zu verfälschten Sensorwerten führen. Um dies zu verhindern wird der Magnetfeldsensor alle 30 Sekunden zurückgesetzt. Das führt dazu, dass der erhaltenen Datenstrom vergleichbar bleibt.

Steuerung des Kalibrierungsmodus	Kalibrierung bedeutet, dass die verwendeten Offsets neu berechnet werden sollen. Soll das gemacht werden, aktiviert die Logikschicht die entsprechenden Funktionen in der Datenschicht. Dort werden dann über einen spezifischen Zeitraum Daten aufgezeichnet. Anschliessend wird wie im Abschnitt 2.2.2 beschrieben ein neuer Offset berechnet.
Steuerung der Loggingmodus	Die Logikschicht bietet eine Funktion an, um sämtliche Rohsensordaten in ein externes Logfile zu schreiben. Der eigentliche Prozess des Loggings findet in der Datenschicht statt. Jedoch wird er von der Logikschicht aus gesteuert.
Auslösung von Event	Registriert die Logikschicht veränderte Sensorwerte, löst sie einen Event aus, der alle registrierten Instanzen informiert.
Steuerung des Simulationsmodus	Um den Client auch ohne physisch vorhandenen Sensor testen zu können wurde einen Simulationsmodus eingebaut. Dieser wird von der Logikschicht aus aktiviert bzw. deaktiviert.

Tabelle 4.3: Sensorspezifische Funktionen der Logiksschicht

Abbildung 4.10 illustriert die Situation der Logikschicht auf Klassenebene. Das Zusammenspiel der einzelnen Klassen wird dabei deutlich gemacht: Die zuständigen Instanzen der Präsentationsschicht registrieren sich bei der Klasse `ListenerStore`. Bei neuen Ereignissen, konkret bei neuen Sensorwerten, wird von der `SensorReader` Klasse, ein Event „abgefeuert“. Auf der anderen Seite steuert die Klasse `SensorReader` den Sensor und gibt die erhaltenen Daten an die Datenschicht weiter. Letzteres wird über direkte Funktionsaufrufe durchgeführt. Auf eine Erläuterung der Datenschicht wird an dieser Stelle verzichtet. Die neu eingeführten Klassen `Record` und `RecordStore` wurden bereits im Kapitel 4.2 vorgestellt.

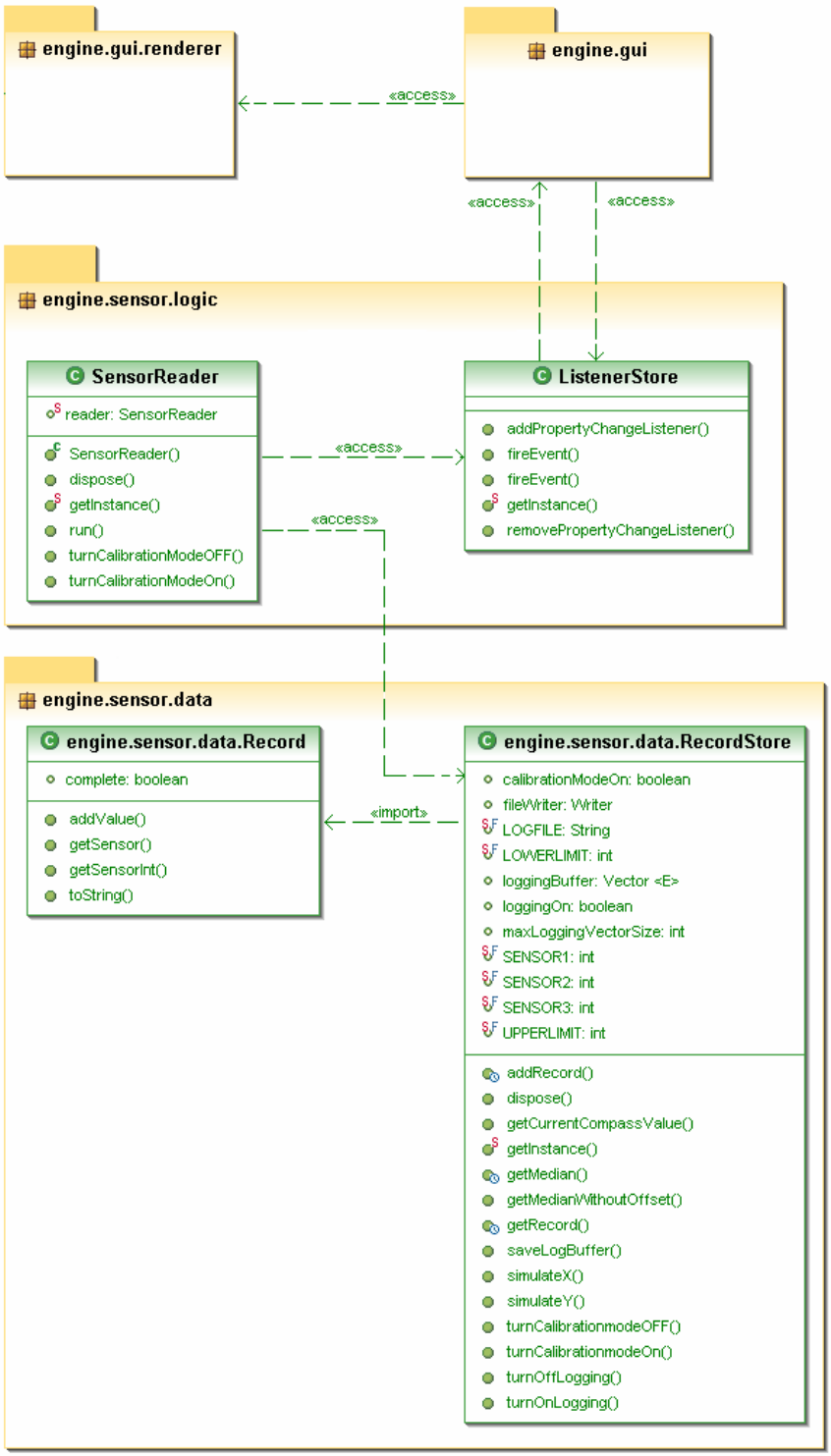

Abbildung 4.10: MobileGame Client: Auszug aus der Logikschicht

4.3.5 Benutzertests

Nachdem die Software erfolgreich integriert worden und lauffähig ist, heisst das leider noch lange nicht, dass diese auch für das geplanten Experiment genügt. Es gibt eine stattliche Anzahl von Unsicherheitsfaktoren:

- **Faktor Stromversorgung:** Es ist das erste Mal, dass das MobileGame in dieser Hardwarekonfiguration (mit einem externen Sensor, der zusätzlichen Strom bezieht) durchgeführt wird. Dabei stellt sich zum Beispiel die Frage, ob die Batterie für ein 90-minutiges Spiel ausreicht.
- **Faktor WLAN**[8]**:** Die PDAs verfügen über tendenziell schwache WLAN Antennen die von Zeit zu Zeit die Verbindung verlieren. Das muss unter realen Bedingungen getestet werden.
- **Faktor Skalierbarkeit:** Das Spiel wird normalerweise mit bis zu 10 PDAs d.h. Clients gleichzeitig gespielt. Erduldet die neue Software unter realen Bedingungen die Last?
- **Faktor externe Software-Bibliotheken:** Die nur teilweise auf den Pocket PC portierte SWT Bibliothek beinhaltet noch eine Reihe von Fehlern. Es soll sichergestellt werden, dass zur Laufzeit keine auftreten werden.
- **Faktor Sensorboards:** Das am PDA befestigte Sensorboard muss stabil befestigt werden, so dass der Dateninput nicht plötzlich abreist. Die Hardwarekonfiguration muss, obwohl es sich um einen Prototypen handelt, ein vernüftiges Niveau an Stabilität aufweisen, das ein Spiel von 90 Minuten ermöglicht.
- **Weitere Faktoren:** Neben den erwähnten Faktoren gibt es auch immer unbekannte Störfaktoren, die eintreten können. Nur durch ausführliches Testen kann auch diese Unsicherheit reduziert werden.

Neben der Reduktion des Ausfallrisikos während des Experimentes kann durch Benutzertests auch die Routine des Experimentalleiters im Umgang mit dem System erhöht werden, so dass im richtigen Experimenten-Setup alles reibungslos und zügig abläuft und die Probanden nicht durch lange Wartezeiten negativ beeinflusst werden. All diese Gründe haben dazu geführt, dass im Rahmen dieser Arbeit zwei grössere Testversuche durchgeführt wurden. Nachfolgender Abschnitt erklärt deren Konfiguration.

[8]Die Netzwerkverbindung über WLAN war schon vorher ein Bestandteil des MobileGames und liegt nicht im Verantwortungsbereich dieser Arbeit. Um ein stabiles System zu gewährleisten, wurde dieser Faktor trotzdem miteinbezogen

Durchgeführte Tests

Das veränderte MobileGame wurde mit zwei umfassenden Benutzertests ausgiebig auf Schwachstellen überprüft. Dabei wurde die Software so konfiguriert, wie sie auch beim Experiment gespielt werden sollte. Tabelle 4.4 fasst die Tests zusammen.[9]

Systemtest	**Nr. 1**	**Nr. 2**
Datum	25.10.2006	31.10.2006
Teilnehmer	3	3
Sensorboards	1	3
PDAs	3	5
Dauer	60 min.	90 min.
Ort	Uni Irchel	Uni Irchel

Tabelle 4.4: Durchgeführte Benutzertests

Resultate

Die oben beschriebenen Benutzertests führten eine Reihe von Schwachpunkten zu Tage. Es konnten auch einige intermittierende Fehler gefunden werden. Der erste derartige Fehler trat nach 60 Minuten auf. Tabelle 4.5 auf Seite 68 zeigt einen Auszug aus den gemachten Entdeckungen.

[9] Für die Benutzertests wurde Design Alternative IV verwendet. Der Grund liegt darin, dass der Entscheid bereits getroffen worden war, das Experiment mit der Design Alternative IV durchzuführen. Der Entscheid ist im Unterkapitel 5.2.1 dokumentiert.

Problem	Beschreibung	Lösung
Instabile Befestigung	Das Sensorboard wird mit einem Zwischenstückes am PDA befestigt. Diese Konstruktion war bei den Tests noch unstabil und heikel. Durch kleine Bewegungen mit dem Gerät haben ungeübte Testteilnehmer Drähte gerissen oder das Board sonst vom PDA getrennt.	Die Lösung welche von Peter Vorburger daraufhin implementiert wurde, war die zusätzliche Befestigung des Boardes mit Gummibändern am PDA. Das funktioniert einwandfrei und ohne das Experiment einzuschränken.
Kalibrierung	Es wurde während der Tests festgestellt, dass die Kalibrierung nicht vereinheitlicht werden kann. Jedes Sensorboard besitzt unterschiedliche Eigenschaften. Zudem scheint die Kalibrierung auch ortsabhängig zu sein.	Software wurde abgeändert, so dass die Kalibrierung individuell pro Sensorboard vorgenommen werden kann.
Reset Verhalten	Im Setup mit den Sensorboards wurden teilweise intermittierende Reset des PDAs durchgeführt, so dass sämtliche Daten gelöscht wurden.	Das beschriebene Verhalten konnte durch ein erneutes Anlöten der Energieversorgungsdrähte an der Batterie beseitigt werden.
„Display Disposed“ Verhalten	Es wurde während der Tests festgestellt, dass sich die Software nach intermittierenden Zeitabständen plötzlich selbst beendete.	Die sehr zeitraubende Suche nach der Ursache dieser Programmabstürze hat ergeben, dass der im Hintergrund laufende Logging-Thread einen Einfluss darauf hat. Es musste eine zusätzliche Pufferung der Sensordaten eingebaut werden, welche dann in fixen Zeitabständen von 20 Sekunden eine Logdatei aktualisiert.

Kartengrösse	Sobald bestimmte Karten geladen wurden, hat sich die Software selbst beendet.	Es konnte entdeckt werden, dass das MobileGame bereits von früheren Versionen einen Bug in sich trug, der bei Karten die höher als breiter sind, dieses Verhalten provozierte. Das Problem konnte behoben werden.

Tabelle 4.5: Resultate der Benutzertests

5

Experiment

Bereits zu Beginn dieser Arbeit war das definierte Ziel die Durchführung des endgültigen MobileGame Experimentes. Die in Kapitel 3 und 4 beschriebenen Aspekte bilden die dazu nötige technische Vorarbeit. Im aktuellen Kapitel soll nun das durchgeführte Experiment beschrieben werden. Dabei werden zunächst die Ziele des Experimentes erläutert. Aus diesen abgeleitet werden der Experimentaufbau und schliesslich die erhaltenen Resultate dargelegt.

5.1 Ziele

Die Ziele dieses Experimentes leiten sich aus jenen des MobileGames ab. Und das definierte Ziel des MobileGames ist, zu erforschen, wie Menschen lernen in zunächst unbekannten Umgebungen zu navigieren, sich zu recht zu finden, und wie man sie dabei unterstützen kann. Dieses Globalziel wurde in zwei Teile aufgegliedert [Schwabe and Göth, 2005a]:

- **Teilziel 1:** Evaluation der Zusammenhänge zwischen motivierenden Aspekten des Spieles mit der Benutzerfreundlichkeit des Systemes.
- **Teilziel 2:** Systematische Erforschung der Effekte bzw. des Nutzens des Spieles auf den Lernprozess

Bis zum heutigen Zeitpunkt hat ein Grossteil der Forschung im Bereich Mobile Learning auf das **Teilziel 1** hingearbeitet. Das gilt auch für einen Teil dieser Arbeit. Mit dem Experiment wird nämlich versucht, zu evaluieren ob die zusätzlichen Navigationshilfen, welche implementiert wurden, die Benutzerfreundlichkeit des Spieles verbessern, zum andern wird angestrebt allenfalls Hinweise für mögliche Verbesserungen zu erhalten. Neben diesen Aspekten soll jedoch auch

ein Beitrag zu **Teilziel 2** geleistet werden. Wie bereits in Kapitel 1.5.3 erwähnt, ist es das definierte Ziel dieser Arbeit, zu überprüfen, ob dynamische Navigationshilfen mit Magnetfeldsensoren das zu Beginn dieser Arbeit diskutierte *Fokusproblem* allenfalls reduzieren können. Nach den beschriebenen Zielen soll in einem nächsten Schritt der Weg dahin dargelegt werden.

5.2 Aufbau

Bei der Durchführung des Experimentes galt es verschiedene Designaspekte zu berücksichtigen. Dazu gehören die Spielregeln, das verwendete Spielgebiet, ebenso jedoch die Teamaufteilung wie die eigentliche Konfiguration der Clients. Bei letzterem Punkt stellt sich zum Beispiel die Frage welcher der vier implementieren Modi wohl am geeignetsten ist, für das Experiment.[1] Der kommende Abschnitt widmet sich dieser, und weiteren Fragen welche die Konfiguration der MobileGame Clients betreffen.

5.2.1 Client Konfiguration

Das MobileGame bietet dem Spielleiter eine Vielfalt von Einstellungsmöglichkeiten an. Zum einen werden diese zentral vom MobileGame Server aus gesteuert zum anderen jedoch auch lokal mittels einer Konfigurationsdatei (Anhang C.1 auf Seite 97 enthält ein Beispiel einer solchen Datei). Wie aus Unterkapitel 4.3.2 hervorgeht, verfügt der MobileGame Client nun über eine Reihe von Design Alternativen, um die vom Sensor erhaltenen Informationen darzustellen. Die Auswahl der für das Experiment passendsten Alternative wurde vom Spielverantwortlichen getroffen. Der nächste Abschnitt geht näher darauf ein.

Entscheid für eine Design Alternative

Die vier implementierten Design Alternativen wurden dem Projektverantwortlichen des MobileGames vorgelegt. Christoph Göth hat sich nach Absprache mit Peter Vorburger schliesslich für die Design Alternative IV (siehe Kapitel 4.3.2 auf Seite 49) entschieden. Alternative IV verfolgt den Ansatz, dass sie die aktuelle Blickrichtung des Benutzers als Pfeil ausgehend von seiner Position

[1] Anmerkung des Autors: Beim Abgabetermin dieser Arbeit (12. Dezember 2006) war das generelle Setup des Experimentes klar definiert. Es wurde jedoch erst ein Experiment mit wirklichen Probanden durchgeführt. Weitere sollen laut Angaben des Spielleiters (Christoph Göth) folgen. Das Dokumentierte bezieht sich in erster Linie auf das bereits durchgeführte Experiment.

darstellt. Alternative I hat zwar bewiesen, dass es prinzipiell machbar wäre, eine Kartenrotation in das MobileGame einzubauen, dass jedoch die begrenzte Rechenleistung des iPAQs in Kombination mit den restlichen laufenden MobileGame Komponenten eine Verzögerung von bis zu drei Sekunden verursacht. Das ist für den Benutzer eines dynamischen und mobilen Spieles zu hoch und war deshalb für die geplanten Experimente ungeeignet. Alternative IV hat gegenüber den übrigen beiden Varianten den Vorteil, dass sie sehr intuitiv zu sein scheint. Ein Benutzer sieht sich im Zentrum und einen von sich wegführenden Pfeil. Dabei könnte es für den Benutzer naheliegend sein, dass damit virtuell seine Blickrichtung indiziert wird. Ob die bei dieser Entscheidung getroffenen Annahmen korrekt waren, soll das bevorstehende Experiment zeigen. Neben der adequaten Design Alternative, war es zudem wichtig, die sensorspezfische Konfiguration des Clients zu bestimmen.

Sensorkonfiguration

Während den Benutzertests wurde entdeckt, dass keine generelle Kalibrierung vorgenommen werden kann, sondern das jeder Sensor unterschiedliche Eigenschaften aufweist. Deshalb wurden die verwendeten Sensoren vor dem Experiment individuell kalibriert. Tabelle 5.1 fasst die Resultate der Kalibrierung zusammen.

Offset	Sensor 2	Sensor 3	Sensor 4
X-Achse	1422	984	1366
Y-Achse	1343	1183	1454

Tabelle 5.1: Verwendete Sensorkalibrierung (Einheit: ADC Messpunkte)

5.2.2 Evaluations Methode

Um die gesetzten Ziele zu erreichen, musste von Beginn an definiert werden, welche Methoden bei der Evaluation des Experiments zur Anwendung kommen sollten. Dabei gab es drei Techniken, auf welche man sich stützen wollte. Es kamen Videoaufzeichnung, Befragungen und moderierte Diskussionsrunden zum Einsatz. Die genaue Vorgehensweise wird nachfolgend für jede dieser Techniken separat erläutert.

Videoaufzeichnung

Das Verhalten einiger der teilnehmenden Teams wird während der gesamten Spielzeit auf Video aufgezeichnet. Man erhofft sich davon insbesondere eine qualitative Auswertung des Verhaltens der Teilnehmer. Kann man auf einer derartigen Aufzeichnung relativ einfach erkennen, worauf der Fokus der Teilnehmer gerichtet ist, so kann nach Abschluss des gesamten Experimentes bestimmt werden, ob Teams, welche einen mit Sensor ausgestatteten PDA zur Verfügung hatten ihren Fokus mehr auf die Umgebung gerichtet haben, als jene die das Spiel mit einem „normalen“ PDA spielten.

Befragung

Im Anschluss an jedes Spiel muss jeder der Teilnehmer einen Fragebogen mit total 28 Fragen ausfüllen. Es gab folgende Kategorien von Fragen:

1. Persönliche Daten
2. Fragen zum allgemeinen Eindruck
3. Fragen zur Orientierung
4. Fragen zur Nutzung des PDAs

Aus der Perspektive dieser Arbeit ermöglicht der Einsatz dieses Evaluationsinstrumentes eine Reihe von Messungen. Damit kann die Frage, ob der Einsatz dynamischer Navigationshilfen, wie Blickrichtungsanzeige, die Orientierung erleichtern und möglicherweise der Lerneffekt verbessern, teilweise beantwortet werden. Bei jedem Spiel wurden bestimmte Gruppen mit und andere ohne den Sensor ausgerüstet. Frage 26 (siehe Abbildung 5.1), um ein Beispiel zu nennen, ist dazu sehr gut geeignet. Die Teilnehmer müssen Ihre Einschätzung über bestimmte Entfernungen auf dem Campus abgeben. Das ermöglicht es, im Anschluss an das Experiment zu evaluieren, ob bei Teilnehmern mit Kompassfunktionalität das Lernen bezüglich räumlicher Orientierung effektiver war als bei jenen ohne. Den kompletten Fragebogen findet man im Anhang C.2 auf Seite 98.

Interview / Diskussionsrunde

Nachdem sämtliche MobileGame Spielrunden durchgeführt sind, wird eine Diskussionsrunde bzw. ein elektronisch unterstütztes Brainstorming (eine sogenannte „Groupsystem Sitzung“) mit den Teilnehmern durchgeführt. Diese qualitative Evaluationsmethode hat ein generelles und ein

26. In der folgenden Tabelle finden Sie verschiedene Orte, die sich auf dem Irchel Campus befinden. Bewerte sie, wie weit diese Orte voneinander entfernt sind. Verwenden Sie dabei eine 1 für sehr nahe Orte und 7 für sehr weit entferne Orte. Bitte raten Sie nicht bei Orten, die Sie nicht kennen, sondern lassen Sie diese Felder einfach frei.

	Strickhof	Informatik-dienste	Mensa	Blaue Kuh	Kopierer	Lichthof	Info-schalter	Infosäule	Y35-F-51
Strickhof	X								
Informatik-dienste	X	X							
Mensa	X	X	X						
Blaue Kuh	X	X	X	X					
Kopierer	X	X	X	X	X				
Lichthof	X	X	X	X	X	X			
Info-schalter	X	X	X	X	X	X	X		
Infosäule	X	X	X	X	X	X	X	X	
Y35-F-51	X	X	X	X	X	X	X	X	X

Abbildung 5.1: Evaluation des MobileGame Experiments: Auszug aus dem Fragebogen

spezifisches Ziel: Erstens soll sie dem Verantwortlichen des MobileGame Projektes einen grundsätzlichen Eindruck vermitteln, wie das Spiel als Ganzes bei den Teilnehmern angekommen ist. Zweitens soll eine Liste mit Verbesserungsvorschlägen aus Benutzersicht erstellt werden, welche in eine weitere Entwicklung des MobileGames einfliessen sollen.

5.2.3 Teams

Beim ersten Experiment am 02. November 2006 wurde mit ingesamt vier Teams gespielt. Ein Team bestand aus zwei Personen. Die totale Probandenzahl betrug demnach acht Probanden. Davon haben drei Teams einen mit Sensor ausgestatteten PDA und ein Team einen PDA mit dem herkömmlichen MobileGame Client erhalten. Sämtliche acht Personen mussten nach dem Experiment den oben erwähnten Fragebogen ausfüllen. Gemäss den persönlichen Daten befanden sich sechs davon im ersten Semester, einer im dritten und einer im fünften Semester. Alle gaben an, den Campus Irchel vorher noch nicht gekannt zu haben. Sämtliche der Probanden waren männlich und ihr Alter lag zwischen 20 und 25. Zwei Teilnehmer schätzen Ihre Vertrautheit im Umgang mit Computergeräten als sehr gut (++) ein, vier der Teilnehmer gaben an gut (+) ver-

traut im Umgang mit Computergeräten zu sein und zwei davon sagten aus, dass sie mittelmässig (0) vertraut im Umgang mit Computergeräten sind. Weiter sind fünf davon gut (+) vertraut mit Handhelds/PDAs, einer davon schlecht (-) und zwei gar nicht vertraut (- -) mit Handhelds/PDAs.

5.2.4 Spielgebiet

Im dem Spiel vorgehenden Briefing wurde das Spielgebiet erklärt. Es beschränkt sich auf den Hörsaalbereich, die verschiedenen Cafeterias/Mensen, den angrenzende Aussenbereich sowie auf den Strickhof des Campus Irchel (siehe Karte in Abbildung 5.2).

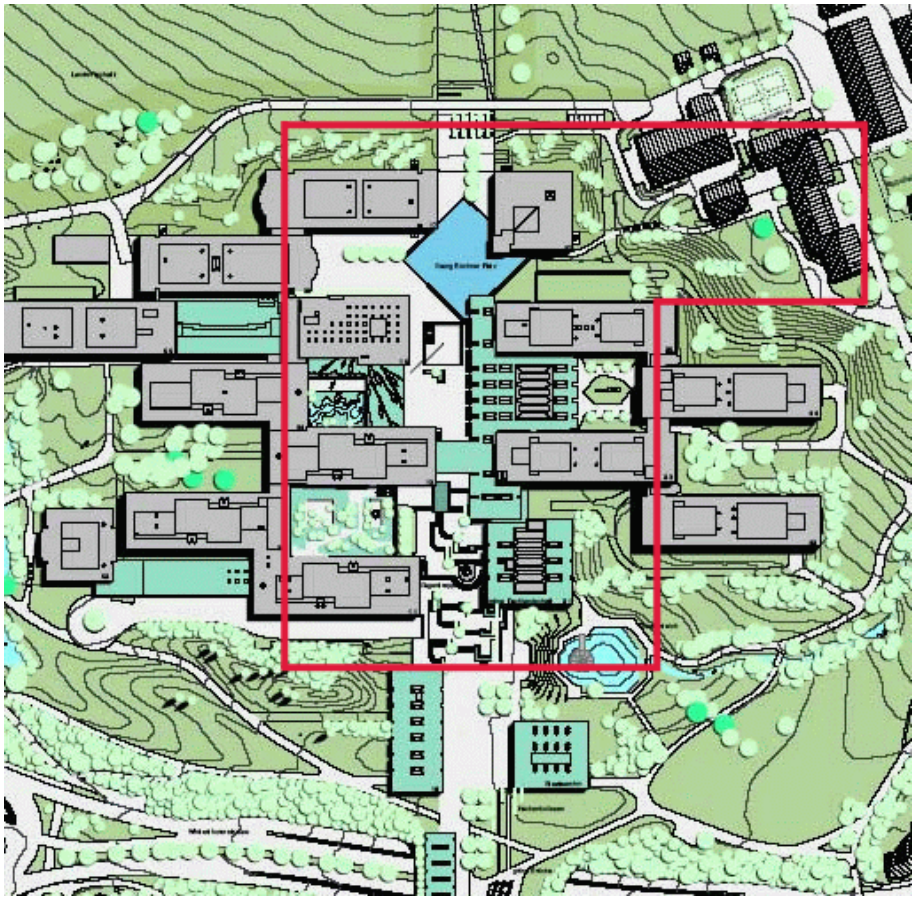

Abbildung 5.2: Spielbereich des MobileGame Experimentes auf dem Campus Irchel der Universität Zürich

5.2.5 Spielregeln

Jedes Spiel braucht Regeln. Es gibt die grundsätzliche Regel, dass jedes Team sämtliche aufgetragene Aufgaben (in MobileGame Terminologie als „Task“ bezeichnet) erfüllen muss um das Spiel zu beenden. Parallel zu dieser übergeordneten Regel gibt es auch Richtlinien, die die Interaktion der Teams definieren. Mehr dazu erfährt man im Abschnitt „Huntingmode“. Den dritten und letzten Eckpunkt des Spieles bilden die sogenannten Annotationen. Die folgenden Abschnitte gehen auf jeden dieser drei Punkte näher ein.

Huntingmode

Das Spiel wurde im sogennanten „Huntingmode“ konfiguriert. Das bedeutet, dass jedes Team die übergeordnete Aufgabe hat, ein bestimmtes anderes Team zu verfolgen und virtuell einzufangen, gleichzeitig aber auch durch ein anderes Team gejagt wird und verhindern muss, dass es selbst eingefangen wird. Das Einfangen geschieht in der virtuellen Spielwelt. Sobald das Jägerteam nahe genug beim Opferteam ist, kann es durch einen Klick auf das Positionssymbol des Opferteams dieses einfangen. Das eingefangene Team muss dem Jägerteam eine symbolische Trophäe überreichen. In diesem Experiment wurden dafür vorgängig Schokoriegel verteilt, die dann übergeben werden mussten. Neben dieser eher globalen Aufgabe mussten jedoch auch sogenannte „Tasks“ erfüllt werden.

Tasks

Mittels Tasks sollen die Spieler des MobileGame mit grundsätzlichen ortsbezogenen Informationen über das Universitätsgelände versorgt werden. Die Spieler müssen sich jedoch dieses Wissen selbst aneignen, nämlich durch das Lösen eben jener „Tasks“. Im ersten Spiel des Experimentes wurden die Teilnehmer mit folgenden Tasks konfrontiert:

1. Sucht das Buch Murphys Windows-Gesetze von Joachim Graf. Nutzt dafür den vordersten Suchcomputer und bittet die Bibliothekarin um Hilfe, falls Ihr es selbst nicht schafft. Auf Seite 43 findet Ihr die Lösung zu folgender Frage: Wie lautet das erste Gesetz zur Speicherung von benutzerdefinierten Einstellungen?

2. Hier befindet sich die Info-Säule. Man kann hier mit Hilfe des Computers nützliche Informationen rund um die Uni erhalten. Findet die Telefonnummer von Christoph Göth heraus und ruft ihn an. Von ihm erhaltet Ihr eine Aufgabe und später auch das Passwort.

3. Hier befindet sich die Walk-In Beratung der Informatikdienste. Ihr erhaltet hier Hilfe bei technischen Problemen aller Art. Lasst euch erklären, wie ihr mit einem Laptop ins WLAN einloggen könnt. Dafür erhaltet ihr ein Passwort.

Annotationen

Neben den Tasks, welche als Pflichtaufgaben gelten, haben die Teilnehmer die Aufgabe besondere Orte zu finden und diese mit virtuellen Post-its, sogenannten Annotationen zu versehen. Die Studenten sollten die Orte dabei wie folgt kategorisieren:

- Orte, an denen man sich gut mit Studenten treffen kann, um sich auszutauschen [T]
- Orte, an denen man gut in Ruhe mit dem Laptop arbeiten kann [A]
- Orte, die eine spezielle Funktion haben [F]

Diese Ort sollen mit der jeweiligen Kategorie und einer kurzen Begründung auf der Karte digital annotiert werden. Am Schluss des Spieles gewinnt das Team mit der höchsten Anzahl unterschiedlicher Orte. Das Gewinnerteam wird mit einem Preis belohnt.

5.3 Resultate

Nach der Durchführung sämtlicher Spielrunden werden die gesammelten Daten ausgewertet. Bis zum Abgabetermin dieser Arbeit wurde vom Spielleiter jedoch erst eine Spielrunde durchgeführt. Deswegen können an dieser Stelle noch keine spielübergreifenden Vergleiche und Auswertungen angestellt werden. Aus dem selben Grund wurde die erwähnte Diskussionsrunde noch nicht durchgeführt und deshalb können an dieser Stelle auch diese Resultate noch nicht dokumentiert werden. Trotz deren Unvollständigkeit sollen die wenigen bereits vorhanden Daten im Sinne eines Zwischenstandes nachfolgend ausgewertet werden. Der Fokus der Auswertung liegt auf jenen Bereichen, die das zu Beginn erwähnte Fokusproblem und die Erweiterungen des MobileGame betreffen.

5.3.1 Auswertung der Fragebogen

Insgesamt konnten zum gegebenen Zeitpunkt acht Fragebogen ausgewertet werden. Zwei dieser acht Teilnehmer haben das Spiel ohne und sechs mit Sensor gespielt. Deshalb können zum

aktuellen Zeitpunkt **keine statistisch signifikanten Aussagen** gemacht werden. Im Sinne einer Veranschaulichung werden die bereits vorhandenen Resultate ausgewertet. Dabei wird zwischen **quantifizierbaren Aussagen** und **nicht quantifizierbaren Aussagen** unterschieden.

Quantifizierbare Aussagen

Die Testpersonen mussten verschiedene Aspekte zu ihrem allgemeinen Eindruck des Spieles auf einer Skala von 1 = sehr wenig und 5 = sehr viel bewerten. Es wurde eine Auswertung gemacht für jene Teilnehmer die mit Blickrichtungsanzeige ausgestattet waren (Tabelle 5.2 auf Seite 77) und eine für jene die ohne Blickrichtungsanzeige auskommen mussten (Tabelle 5.3 auf Seite 78). Die Tabellen gliedern die Ergebnisse in die beiden Teile *Allgemeiner Eindruck* und *Orientierungsspezifische Punkte*.

	Mittelwert	**Standardabweichung**
Allgemeiner Eindruck		
Spass am Spiel	4.2	0.8
Hilfreich um Campus kennenzulernen	3.8	0.8
Neuentdecktes auf dem Campus	4.0	0.6
Kontaktpflege mit Kommilitonen	3.5	0.8
Generelle Bedienung des PDA	4.5	1.0
Orientierungsspezifische Punkte		
Positionsanzeige: Beurteilung als Navigationshilfe	4	1.1
Pfeil: Beurteilung als Navigationshilfe	2.2	1.5
Ablenkung von der Umgebung durch manuelles Refresh	2.9	1.9
Fokus auf PDA auf Kosten der Umgebung	3.2	1.5

Tabelle 5.2: Auswertung der Fragebögen der Teilnehmer mit Blickrichtungsanzeige

Zuerst zum **allgemeinen Eindruck**: Kontaktpflege mit Kommilitonen erhielt sehr wenig Punkte lag aber trotz dem noch über der Mittelmässigkeitsmarke (3). Laut Aussagen von Probanden hätten sie sich mehr Teilnehmer beim Spiel und mehr Spielzeit gewünscht. Die generelle Bedienung des PDA erhielt die besten Noten. Die Ergebnisse jener Teilnehmer mit Blickrichtungsanzeige unterscheiden sich im allgemeinen Teil nur unwesentlich von jenen die keine solche Anzeige hatten. Neben dem allgemeinen Eindruck des Spieles sind für diese Arbeit vor allem die **orien-**

	Mittelwert	Standardabweichung
Allgemeiner Eindruck		
Spass am Spiel	4	0.0
Hilfreich um Campus kennenzulernen	4	0.0
Neuentdecktes auf dem Campus	4	0.0
Kontaktpflege mit Kommilitonen	4	0.0
Generelle Bedienung des PDA	5	0.0
Orientierungsspezifische Punkte		
Positionsanzeige: Beurteilung als Navigationshilfe	4.5	0.7
Ablenkung von der Umgebung durch manuelles Refresh	3.2	1.5
Fokus auf PDA auf Kosten der Umgebung	2.9	1.0

Tabelle 5.3: Auswertung der Fragebögen der Teilnehmer ohne Blickrichtungsanzeige.

tierungsspezifischen Aspekte von Interesse. Die schlechteste Beurteilung als Navigationshilfe hat der Pfeil erhalten. Interessanterweise waren sich die Teilnehmer uneinig diesbezüglich. Das spiegelt sich auch in der höchsten Standardabweichung dieses Datenpunktes wieder. Einige der Probanden gaben an, den Pfeil eher als verwirrend zu empfinden. Nach mündlicher Nachfrage begründeten sie dies damit, dass es auf der einen Seite zwar logisch erscheint, dass der Pfeil die Blickrichtung anzeige, jedoch das Umdenken bzw. die Verbindung der tatsächlichen mit der auf der Karte angezeigten Blickrichtung gar nicht so einfach sei. Dabei sei es vor allem verwirrend, dass man selbst zwar „geradeaus“ schaue, der Pfeil jedoch zum Beispiel nach hinten rechts zeige. Es scheint also einmal mehr das Phänomen aufzutreten, dass das Übertragen der virtuellen Welt auf die reale Welt Schwierigkeiten bereitet. Es sei hier jedoch nochmals betont, dass aufgrund der kleinen Anzahl von Messungen zu diesem Zeitpunkt keine signifikanten Schlussfolgerungen getroffen werden können.

Nicht quantifizierbare Aussagen

Auf der letzten Seite des Fragebogens konnten die Teilnehmer ihre Meinung durch einen kleinen Freitext äussern. Sie wurden aufgefordert, besonders positive und besonders negative Punkte des Versuches zu erwähnen. Daten dieser Art sind leider schwierig zu quantifizieren, geben einem jedoch meistens einen sehr guten Eindruck über die Meinung der Versuchsteilnehmer. So wurde

zum Beispiel die Blickrichtungsanzeige teilweise als besonders negativer Aspekt des Versuches erwähnt. Auf der anderen Seite gab es auch Teilnehmer, die den zusätzlichen Pfeil als „besonders hilfreiches Instrument“ erwähnt haben. Auch hier gilt, dass die kleine Anzahl Messungen von lediglich zwei Voten noch keinen Schluss zulassen.

5.3.2 Aufgezeichnete Sensordaten

Neben bisher erwähnten Erhebungsmethoden wie Videoaufzeichnung, Fragebogen und Gruppendiskussion wurden während des gesamten Spieles auch sämtliche von der Sensorik gemessenen Werte aufgezeichnet. Es konnten pro Sensor 17.5 Datenpunkte pro Sekunde aufgezeichnet werden. Bei einer Spieldauer von 1.5 Stunden entspricht das rund 95’000 Datenpunkten pro Sensor und Spiel. Abbildung 5.3 auf Seite 79 zeigt einen aufgezeichneten Ausschnitt des Magnetfeldsensors[2].

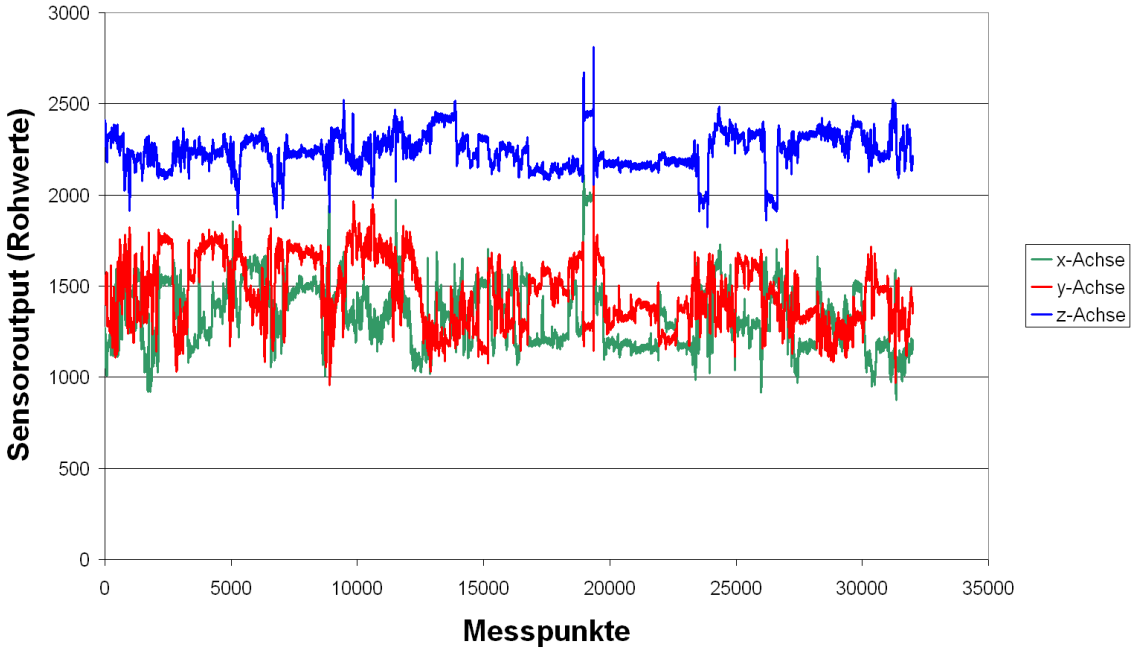

Abbildung 5.3: Während dem Experiment aufgezeichnete Rohwerte: Beispiel für die Magnetfeldsensoren (Aufzeichnungsdauer: 30 Minuten)

Es wird klar, dass die vorgenommene Anbindung des Sensorboards noch eine Vielfalt an wei-

[2]Der gezeigte Ausschnitt wurde während ca. 30 Minuten aufgezeichnet. Für eine Auflösung im Sekundenbereich siehe Abbildung 2.3 auf 13

teren Datenauswertungen ermöglicht. Vor allem, wenn dabei das Instrumentarium des Datamin-ings zur Anwendung gebracht wird, kann das Lernen von Orientierung aus einer weiteren Perspektive analysiert werden. Im Kapitel 6.2 wird eine weitere Auswertung der Sensordaten noch im Detail diskutiert.

5.3.3 Fazit

Nachdem vom MobileGame Verantwortlichen Ch. Göth bis anhin lediglich eine Spielrunde mit nur acht Teilnehmer durchgeführt wurde, können noch keine statistisch relevanten Aussagen gemacht werden. Festgehalten werden kann jedoch, dass das Experiment bereits im Detail vorbereitet wurde und erfolgreich begonnen werden konnte. Ebenfalls kann gesagt werden, dass bereits eine modelhafte erste Auswertung vogenommen wurde, nach welcher sich zukünftige Auswertungen richten können. Auch wurde klar, dass die während dem Experiment aufgezeichneten Sensormessungen zusätzliche Daten liefern. Das Thema wird im Unterkapitel „Weiterführende Arbeiten“ (Seite 83) wieder aufgegrifen.

6

Schlussfolgerung und Ausblicke

Während dieser Diplomarbeit wurde gezeigt, dass die Visualisierung von Sensordaten auf dem verwendeten iPAQ grundsätzlich realisierbar ist. Zusätzlich wurde demonstriert, dass die Sensorfunktionalität in bestehende Navigationssysteme, wie das MobileGame, eingebaut und verwendet werden kann. Eine beispielhafte Implementation dafür wurde vorgenommen und getestet. Um die realisierte Implementation zu evaluieren, wurde ein Experiment vorbereitet und modelhaft durchgeführt. Die Durchführung hat gezeigt, dass der entwickelte Prototyp geeignet ist und fehlerfrei funktioniert. Aufgrund dessen, dass das Experiment vom MobileGame Verantwortlichen Christoph Göth nicht abgeschlossen worden ist, können zu diesem Zeitpunkt noch keine statistisch relevanten Aussagen über den Nutzen des erstellten Prototypes als Navigations- und Lernhilfe gemacht werden. Auch kann noch nicht gesagt werden, ob das Fokusproblem durch dynamische Ansätze reduziert werden kann. Obwohl diese Arbeit zufriedenstellende Ergebnissen erzielt hat, wurden an gewisse Grenzen gestossen. Besonders hinsichtlich der verwendeten Hard- und Software wurden Limiten entdeckt, die im nächsten Unterkapitel beschrieben werden. Ohne diese Limiten hätte auch intuitives Echtzeitkartendrehen implementiert werden können.

6.1 Die Limiten dieser Arbeit

Im Verlauf dieser Arbeit kamen verschiedene Probleme zum Vorschein, die nachfolgend zusammengefasst werden sollen.

Verwendete Entwicklungsplatform

Obwohl die verwendete Entwicklungsplatform (J2ME mit SWT) in Situationen wo keine hardwarenahe Programmierung erforderlich ist, eine interessante Alternative zur Entwicklung mobiler Anwendungen ist, verursachte sie in dieser Arbeit auch einige Probleme. Insbesondere das vorgegebene SWT scheint für fortgeschrittene Grafikprogrammierung auf Windows Mobile nicht sonderlich geeignet zu sein. Die aktuelle Windows Mobile Version beinhaltet Fehler und weist im Vergleich zur PC Version viele Einschränkungen bei der Grafikprogrammierung auf.

Verwendete Hardware

Die Beschränkung des Hauptspeichers auf 64MB (wobei dieser auch als Datenspeicher genutzt wird) ist hinsichtlich der Grafikprogrammierung ein Hindernis. Animierte Grafiken müssen vielfach gepuffert werden. Da es bei der Navigation um Karten geht welche über eine hohe Auflösung verfügen, sind die Ansprüche solcher Software an den Hauptspeicher relativ hoch.

Fehlerhafte Positionsbestimmung mit WLAN

Ein Störfaktor während des Experiments war die ungenaue Positionsbestimmung über WLAN. An Orten, wo kein oder schlechter Empfang der WLAN-Signale herrscht, versagt die Positionsbestimmung von Ekahau teilweise komplett. Das wirkt auf die Teilnehmer störend und verfälscht die Resultate des Experimentes. Die Positionsbestimmung an sich ist jedoch kein Gegenstand dieser Arbeit, hatte aber Einfluss auf das durchgeführte Experiment.

Aussagekraft des Experimentes

Aufgrund der geringen Anzahl durchgeführter Spielen besitzen die Resultate im Unterkapitel 5.3 (siehe Seite 76) keine relevante Aussagekraft. Um dynamische Navigationsansätze zu testen, müssten Experimente mit deutlich grösseren Teilnehmerzahl durchgeführt werden. Zudem müsste allenfalls auch das Setup des Experimentes geändert werden. Die unterschiedlichen Teilaufgaben im Experimenten-Setup betonen nicht die Effizienz beim Orientieren. Die verschiedenen Aufgaben (Huntingmodus, Tasks, Annotations) beeinflussen sich gegenseitig und erschweren eine aussagekräftige Auswertung.

6.2 Weiterführende Arbeiten

Im vorliegenden Unterkapitel werden mögliche zukünftige Themen dieses Forschungsgebietes aufgezeigt, welche auf der vorliegenden Diplomarbeit aufbauen können.

Zusätzliche Auswertung der aufgezeichneten Sensordaten

Die während des Experimentes aufgezeichneten rohen Sensordaten (siehe auch Abschnitt 5.3.2 auf Seite 79) bieten sich an für weitere Auswertungen. Es wäre insbesondere interessant, mittels Techniken des Dataminigs zu versuchen bestimmte Muster zu erkennen. So könnten man zum Beispiel versuchen (wie in der Arbeit von [Ladetto and Merminod, 2002] erwähnt) mit Gyroskop und Magnetfeldsensor Schrittmuster zu erkennen. Es sind jedoch auch andere Muster denkbar. So zum Beispiel könnte man versuchen festzustellen wann eine Person stehenbleibt (Beschleunigungssensoren), wann sie ein Gebäude verlässt bzw. betritt oder wann sie an der Mensa vorbei geht (Gassensoren). Diese zusätzlichen Informationen könnten zum einen für die Navigation verwendet werden, im Sinne eines INS, oder sie können für weitere Auswertungen hinsichtlich des Lernerfolges benutzt werden.

Zusätzliche Experimente

Um den Nutzen von Indoornavigationssystemen zur mobilen Lernunterstützung oder zur Orientierung als solche zu evaluieren, sind weitergehende Experimente mit zusätzlichen Probanden nötig. Ebenfalls denkbar sind zusätzliche Experimente mit alternativen Darstellungstechniken oder geänderten Experimenten-Setups. Besonders interessant wäre Experimente mit einem Belohnungssystem, dass auf die Effizienz der Orientierung fokussiert.

Prüfen von alternativen Entwicklungsplatformen

Das MobileGame wurde mittels der Java2 Micro Edition (J2ME) entwickelt. Eine Neuentwicklung mittels dem .Net Compact Framework wäre eine Alternative, die ebenfalls in Betracht gezogen werden sollte. Der Vorteil davon wäre insbesondere, dass die Möglichkeit besteht, direkt nicht interpretierte Programmiersprachen wie C++ einzubinden und einfacher auf externe Bibliotheken zuzugreifen.

Anwendung alternativer Techniken zur Positionsbestimmung

Um die beschriebene fehlerhafte Positionsbestimmung mittels WLAN zu umgehen, könnten alternative Techniken zur Indoornavigation in Betracht gezogen werden. Eine interessante Lösung wäre die Umsetzung des von [Bachmann, 2006] vorgeschlagenen Interpolationansatzes (Aufgrund des massiven Ressourcenbedarfs liegt dies jedoch noch in weiter Zukunft). Ebenfalls denkbar sind Kombination von WLAN-Positionierung und Positionierung mittels Sensordaten.

6.3 Persönliches Schlusswort

Obwohl bis zum Abgabetermin dieser Arbeit die Experimente noch nicht komplett durchgeführt wurden, konnten viele Erkenntnisse bezüglich der Visualisierung von Navigationsinformationen gemacht werden. Zudem konnte die Praxistauglichkeit des verwendeten Sensorboards demonstriert werden. Weiter wurde aufgezeigt, dass die Anbindung externer Sensorik an das Mobile-Game ein interessantes Anwendungsgebiet ist und Potenzial für weitere Untersuchungen bietet. Für die Zukunft wäre es jedoch auch schön, wenn man ausgereiftere und weitergehende Darstellungstechniken realisieren könnte. Persönlich betrachte ich die Arbeit sehr positiv. Die definierten Ziele konnten erreicht werden. Darüber hinaus durfte ich mich in ein für mich neues Gebiet einarbeiten und habe dabei viele Lektionen von unschätzbarer Wert gelernt. Ich bin der Überzeugung, dass die Navigation mittels derartiger Sensorik ein zukunftsweisendes und spannendes Forschungsfeld ist.

Abkürzungsverzeichnis

ADC	Analog Digital Converter
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BMP	Windows Bitmap
CD-ROM	Compact Disc - Read Only Memory
FPS	Frames per Second
GC	Grafic Context
GIF	Graphics Interchange Format
GND	Ground
GPS	Global Positioning System
GUI	Graphical User Interface
IBM	International Business Machines
INS	Inertiales Navigationssystem
IT	Information Technology
J2ME	Java2 Micro Edition
J2SE	Java2 Standard Edition
JDK	Java Development Kit

JPG	Joint Photographic (Experts) Group, auch JPEG
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KB	Kilo Byte
MB	Mega Byte
MVC	Model View Controller
PC	Personal Computer
PDA	Personal Digital Assistant
PNG	Portable Network Graphics
RAM	Random Access Memory
RGB	Rot Gelb Blau
RMI	Remote Method Invocation
RS232	Radio Sector 232
RXD	Receive
SDRAM	Synchronous Dynamic - Random Access Memory
SWT	Standard Widget Toolkit
TIFF	Tagged Image File Format
TTL	Transistor Transistor Logik
TXD	Transmit
UART	Universal Asynchronous Receiver Transmitter
UML	Unified Modeling Language
WLAN	Wireless Local Area Network
WSDD	Websphere Device Developer

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Analog Devices, 2001] Analog Devices (2001). *ADM202 und ADM203 Specifications*. Analog Devices Inc.

[Bachmann, 2006] Bachmann, A. (2006). Indoornavigation mittels Ortsinterpolation. Diplomarbeit, Universität Zürich, Institut für Informatik.

[Brunner, 2005] Brunner, D. (2005). Architekturentwicklung für mobile Spiele. Diplomarbeit, Universität Zürich, Institut für Informatik.

[Ford et al., 2001] Ford, T., Neumann, J., and Bobye, M. (2001). Oem4 inertial: An inertial/gps navigation system on the oem4 receiver. In *14th International Technical Meeting of the Satellite Division of the Institute of Navigation, Salt Lake City, Utah, USA*.

[Freiesleben, 1978] Freiesleben, H.-C. (1978). *Geschichte der Navigation*. Franz Steiner Verlag, Wiesbaden.

[Glinz, 2004] Glinz, M. (2004). Vorlesungsskript: Software engineering I.

[Göth, 2003] Göth, C. (2003). Prototypische Implementierung einer mobilen Spielumgebung für den PDA. Diplomarbeit, Universität Koblenz-Landau, Institut für Wirtschafts- und Verwaltungsinformatik.

[Göth et al., 2006] Göth, C., Frohberg, D., and Schwabe, G. (2006). The focus problem in mobile learning. In *IEEE 4th International Workshop on Wireless, Mobile and Ubiquitous Technologies in Education, Athens, Greece*.

[Göth et al., 2004] Göth, C., Häss, U.-P., and Schwabe, G. (2004). Requirements for mobile learning games shown on a mobile game prototype. In *MLearn2004 conference, Rome, Italy*.

[Hewlett Packard, 2006] Hewlett Packard (Aufruf vom 1. Dezember 2006). http://www.hp.com.

[Honeywell, 1995] Honeywell (1995). An-203: Compass heading using magnetometers. Application note, Honeywell Sensor Products.

[Honeywell, 1996] Honeywell (1996). An-201: Set/reset pulse circuits for magnetic sensors. Application note, Honeywell Sensor Products.

[Honeywell, 2000] Honeywell (2000). An-209: Magnetic current sensing. Application note, Honeywell Sensor Products.

[Honeywell, 2002a] Honeywell (2002a). An-211: Applications of magnetic position sensors. Application note, Honeywell Sensor Products.

[Honeywell, 2002b] Honeywell (2002b). An-213: Set/reset function for magnetic sensors. Application note, Honeywell Sensor Products.

[Honeywell, 2003] Honeywell (2003). *HMC1051/HMC1052/HMC1053 - 1, 2 and 3-Axis Magnetic Sensors*. Honeywell Sensor Products.

[Honeywell, 2004] Honeywell (2004). An-216: Mounting tips for lcc magnetic sensors. Application note, Honeywell Sensor Products.

[Kägi, 2002] Kägi, U. (2002). Sensor Data Acquisition and Processing on a Pocket Pc. Diplomarbeit, Universität Zürich, Institut für Informatik.

[Ladetto and Merminod, 2002] Ladetto, Q. and Merminod, B. (2002). In step with INS - navigation for the blind, tracking emergency crews. *GPS World*, pages 30–38.

[Ladetto et al., 1999] Ladetto, Q., Merminod, B., Terrier, P., and Schutz, Y. (1999). On foot navigation: When gps alone is not enough. In *Third International Symposium on Global Navigation Satellite Systems*.

[Opdenacker, 2006] Opdenacker, M. (Aufruf vom 24. August 2006). http://opdenacker.org/pda/ipaq/connector/ipaq_connector.png.

[Schwabe and Göth, 2005a] Schwabe, G. and Göth, C. (2005a). Mobile learning with a mobile game: design and motivational effects. *Journal of Computer Assisted Learning, Vol. 21*, pages 204–216.

[Schwabe and Göth, 2005b] Schwabe, G. and Göth, C. (2005b). Navigating and interacting indoors with a mobile learning game. In *International Workshop on Wireless and Mobile Technologies in Education*.

[Schweigert et al., 2005] Schweigert, M., Hübner, T., and Ibach, P. (2005). Wlan basierte Ortung mit MagicMap. Studienarbeit, Humboldt Universität Berlin, Institut für Informatik.

[Tschanz, 2005] Tschanz, S. (2005). Towards a framework for an intertial navigation system: Position and velocity extrapolation of motion-based data. Diplomarbeit, Universität Zürich, Institut für Informatik.

[Wikipedia, 2006] Wikipedia (Aufruf vom 10. August 2006). http://de.wikipedia.org/wiki/echtzeit.

A

Aufgabenstellung

Aufgabenstellung

Das Ziel dieser Diplomarbeit ist, ein Experiment aufzubauen und durchzuführen. Dieses Experiment befindet sich im Kontext von Indoor Navigation / -Orientierung. Dabei geht es um Systeme, welche den Benutzer bei der Orientierung in ihm unbekannten Umgebungen unterstützen sollen. Viele Benutzer bekunden Mühe Karten zur Orientierung in Einklang mit der realen Umwelt zu stellen. Bisherige Lösungsansätze berücksichtigen dieses Problem nicht angemessen, sondern arbeiten mit statischen Karten. In diesem Experiment soll untersucht werden, ob dynamische Unterstützungsansätze (wie Rotieren der Karte oder u.U. das Verzichten auf Karten zugunsten alternativer Orientierungshilfen) die herkömmlichen Ansätze übertreffen.

Im Rahmen dieser Arbeit sollen folgende Tätigkeiten angegangen werden:

1. Einarbeitung

Da diese Arbeit einerseits aus einer Implementation und andererseits aus einem Experiment besteht, muss die Einarbeitung zwei Gebiete umfassen. Zur Implementation sollte eine Übersicht über die verwendeten Komponenten geschaffen werden (Bachmann, 2006). Ausserdem muss sich der Diplomand ¡mit der Implementations-umgebung vertraut machen. Für das durchzuführende Experiment ist eine Einarbeitung in Experimentaltechniken und Statistik erforderlich (Bortz & Döring, 2003). Zusätzlich sollten vorgängige Arbeiten (C. Göth, Hüss, & Schwabe, 2004; G. Göth & Lueg, 2006; Lueg, Goeth, & Bidwell, 2006; Schwabe & Göth, 2005; Schwabe, Göth, & Frohberg, 2005) betrachtet werden.

2. Implementation eines Prototypen

Der in dieser Arbeit verfolgte Ansatz zur dynamischen Orientierungsunterstützung baut auf einer Kompassfunktionalität auf. Auf einem mobilen Gerät (iPAQ) soll ein Prototyp implementiert werden, welcher die Funktionalität eines Kompasses nachweist. Dies beinhaltet die Anbindung der externen Sensorik via serieller Schnittstelle an das Gerät, die Umwandlung, Interpretation und visuelle Aufbereitung des Sensordatenstromes. Dieser Prototyp dient nicht nur zum Nachweisen der Machbarkeit, sondern soll es auch erlauben, Evaluationen durchzuführen, welche die Randbedingungen des folgenden Experiments bestimmen.

3. Implementation des Experimentes

Anhand der oben gewonnenen Kenntnisse werden die Ausprägungen der dynamischen Orientierungsunterstützung bestimmt, das Experiment geplant und die dafür erforderliche Infrastruktur erstellt.

4. Durchführung des Experiments

Falls es die Zeit zulässt, soll das Experiment vollständig durchgeführt/begleitet werden (unter der Leitung von Ch. Göth). Ansonsten ist sicherzustellen, dass das Experiment zuverlässig begonnen werden kann.

5. Analyse und Interpretation

Je nach Stand des Experimentes rundet eine Analyse der gewonnen Daten die Arbeit ab.

B

Technische Details

B.1 Schaltschema für Zwischenstück

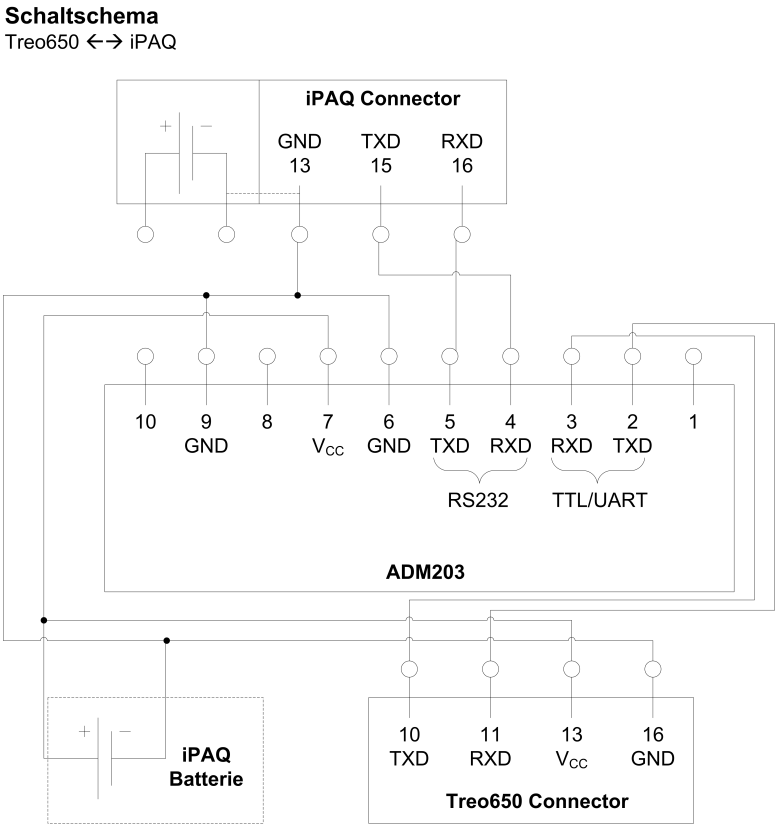

Abbildung B.1: Schaltschema zur Konstruktion eines Zwischenstückes für das Sensorboard

B.2 Spezifikation des iPAQ hx4700

Integrated wireless	WLAN 802.11b, Bluetooth® 1.2, IrDA (FIR), USB and Serial	
Operating system	Microsoft® Windows Mobile™ 2003 Second Edition software for Pocket PC – Premium Edition Pocket versions of Microsoft software are included (Outlook, Word, Excel and Internet Explorer for Pocket PC	
Processor	624 MHz Intel® Bulverde technology-based processor	
Display	4" Transflective type VGA TFT color, 64K colors, LED backlight with power save mode; Portrait and Landscape modes	
Memory	192 MB total memory (128 MB ROM and 64 MB SDRAM) Up to 135 MB user available memory that includes 80 MB iPAQ File Store	
Dimensions	5.17 x 3.03 x .59 in (131 x 77 x 14.9 mm)	
Weight	186.7g (6.6oz) (w/o cover & cards)	
Indicators	Power Button	Charge and Notify indicators Blinking green - notify Blinking amber - charging Solid amber – full charge
	LED – Bluetooth/WLAN	Solid blue LED – wireless radio(s) ON
Power	Battery	Removable/Rechargeable 1800 mAh Lithium-Ion user swappable battery. Optional extended removable/rechargeable 3600 mAh Lithium-Ion battery available for purchase www.hp.com/go/ipaqaccessories
	AC Power	AC Input: 100~240 Vac, 50/60 Hz, AC Input current: 0.3 A max Output Voltage: 5Vdc (typical), Output Current: 2A (typical
	NOTE: Battery run time varies based on the usage pattern of an individual user and the configuration of the handheld. Use of internal wireless capabilities and backlight will significantly decrease battery run time.	
Expansion	SD Slot	Support 1-bit and 4-bit SDSDIO/MMC type memory standard
	CF Slot	1 Type II
Audio	Integrated microphone, speaker and one 3.5 mm stereo headphone/headset jack, MP3 stereo through audio jack and speaker, 5-band equalizer for playback through audio jack	
Design features	4 shortcut programmable buttons, Navigation Touch-pad, Touch-sensitive display for stylus or fingertip, Voice record button, Microphone and Speaker on front-side of unit	
What's in the box	HP iPAQ hx4700 Pocket PC, AC adapter, Plastic Flip Cover, Stylus, Getting Started Guide, and HP iPAQ Pocket PC Companion CD	
Warranty	One-year parts and labor in most regions; 90 days technical support for software in most regions. Additional offers may vary by region.	

Abbildung B.2: Technische Daten des iPAQ hx 4700 [Hewlett Packard, 2006]

C
Experiment

C.1 Konfigurationsdatei

Listing C.1: Beispiel für eine Konfigurationsdatei des MobileGame Client

```
#Client Configuration File
team_name=Team4
team_pass=4444
server_ip=130.60.156.137
server_port=9999
map_path=java\\maps\\
icon_path=java\\icons\\
audio_path=java\\audio\\
skype_name=unizh_game_1
skype_port=11444
tag_name=Tag-7
sensor=1
xoffset=1240
yoffset=1220
```

Tabelle C.1 auf Seite 98 erklärt die Bedeutung der einzelnen Optionen, die in der Konfigurationsdatei verwendet werden.

Eigenschaft	Bedeutung	Akzeptierte Werte
team_name	(Login-)Name des Teams.	String
team_pass	Passwort des Teams.	String
server_ip	IP Adresse des MobileGame Servers.	IPv4 Adresse (String)
server_port	Port auf welchem der MobileGame Server betrieben wird.	Integer mit max. 5 Ziffern
map_path	Verzeichnis in welchem die Karten gespeichert sind.	String
icon_path	Verzeichnis in welchem die Icons gespeichert sind.	String
audio_path	Verzeichnis in welchem die Audiodateien gespeichert sind.	String
skype_name	Benutzername im Skype (diese Funktionalität wurde nicht verwendet in dieser Arbeit)	String
skype_port	Definiert den von Skype verwendeten Port (diese Funktionalität wurde nicht verwendet in dieser Arbeit)	Integer mit max. 5 Ziffern
tag_name	Name des vergebenen Ekahau-Tags.	String
sensor	Bestimmt ob der MobileClient den Sensor einschaltet oder nicht.	0 (aus) oder 1 (ein)
xoffset	Offset zur Kalibrierung des Magnetfeldsensors (X-Achse).	Integer (0-4000)
yoffset	Offset zur Kalibrierung des Magnetfeldsensors (Y-Achse).	Integer (0-4000)

Tabelle C.1: Erklärung zu der Konfigurationsdatei

C.2 Fragebogen

Die Evaluation des durchgeführten Experimentes wurde mit einem Fragebogen vorgenommen. Alle Teilnehmer des Spieles wurden mit dem Bogen befragt. Der Fragebogen befindet sich auf den folgenden vier Seiten.

Fragebogen zum mExplorer

Bitte immer nur EINE Antwort ankreuzen

Persönliche Daten

1	Spielernummer:	
2	Teamnummer:	
3	Geburtsjahr (JJJJ)	
4	Geschlecht	männlich ☐ weiblich ☐
5	Studienfach:	
6	Anzahl Semester	1 = Erstsemestrige ☐ 2 ☐ 3 ☐ 4 ☐ mehr als 5 ☐
7	Wie vertraut sind Sie im Umgang mit Computergeräten?	-- - 0 + ++ gar nicht ☐☐☐☐☐ sehr gut
8	Wie vertraut waren Sie vor dem Spiel im Umgang mit Handhelds/ PDA?	-- - 0 + ++ gar nicht ☐☐☐☐☐ sehr gut

Fragen zum allgemeinen Eindruck

9	Wieviel Spass hat Ihnen die Teilnahme am Spiel insgesamt gemacht?	sehr -- - 0 + ++ sehr wenig ☐☐☐☐☐☐ viel
10	Wie hilfreich war das Spiel für Sie um den Campus kennen zu lernen?	sehr -- - 0 + ++ sehr wenig ☐☐☐☐☐☐ viel
11	Wieviel Neues über Campus und Uni haben Sie durch Spiel und Aufgaben entdeckt?	sehr -- - 0 + ++ sehr wenig ☐☐☐☐☐☐ viel
12	Wie sehr hat das Spiel dazu beigetragen, in Kontakt mit KommilitonInnen zu kommen oder den bestehenden Kontakt auszubauen?	sehr -- - 0 + ++ sehr wenig ☐☐☐☐☐☐ viel
13	Wie gut konnten Sie den PDA bedienen?	sehr -- - 0 + ++ sehr schlecht ☐☐☐☐☐☐ gut

Fragen zur Orientierung

14	Wie gut konnten Sie sich mit Hilfe der Karte und Anzeige der eigenen Position orientieren	sehr -- - 0 + ++ sehr schlecht ☐☐☐☐☐☐ gut
15	Wie hilfreich für die Orientierung war der Pfeil, der die Blickrichtung angezeigt hat (falls vorhanden)?	sehr -- - 0 + ++ sehr wenig ☐☐☐☐☐☐ viel
16	Wie gut konnten Sie sich mit Hilfe der ausgedruckten Karte orientieren (falls vorhanden)?	sehr -- - 0 + ++ sehr schlecht ☐☐☐☐☐☐ gut
17	Wie viel haben Sie während des Spiels die Übersichtsansicht und wie viel haben sie die Detailansicht verwendet. (Summe 100 %)	Übersicht % Detail %

Fragen zur Nutzung des PDAs

18	Wie häufig haben Sie während dem Spiels den PDA-Bildschirm angesehen?	Sehr wenig --- -- - 0 + ++ +++ sehr viel ☐☐☐☐☐☐☐
19	Wie lange haben sie jeweils im Durchschnitt den PDA-Bildschirm angesehen (*sehr kurz*	Sehr kurz --- -- - 0 + ++ +++ sehr lange ☐☐☐☐☐☐☐

	bedeutet wenige Sekunden und *sehr lange* mehrere Minuten)	
20	Das automatische Update der Positionsinformation (falls vorhanden) hat mich stark von der Umgebung abgelenkt.	Stimme --- -- - 0 + ++ +++ stimme nicht zu ☐☐☐☐☐☐☐ zu
21	Dadurch, dass ich immer auf Refresh drücken musste (falls vorhanden), wurde ich stark von der Umgebung abgelenkt.	Stimme --- -- - 0 + ++ +++ stimme nicht zu ☐☐☐☐☐☐☐ zu
22	Ich war, solange **ich** den PDA **selbst** in der Hand hielt, so stark auf den PDA und die dort angezeigten Informationen konzentriert, dass ich nicht viel von der Umgebung wahrgenommen habe	Stimme --- -- - 0 + ++ +++ stimme nicht zu ☐☐☐☐☐☐☐ zu
23	Ich war, solange **mein Mitspieler** den PDA in der Hand hielt, so stark auf den PDA und die dort angezeigten Informationen konzentriert, dass ich nicht viel von der Umgebung wahrgenommen habe	Stimme --- -- - 0 + ++ +++ stimme nicht zu ☐☐☐☐☐☐☐ zu

24. Zeichen Sie skizzenhaft eine Karte des Campus und vermerken Sie die Orte, an die Sie sich noch erinnern können (z. B. Bibliothek, Mensa oder Blaue Kuh).

25. Zeichnen Sie auf der obigen Karte den Weg vom Strickhof zu den Informatikdiensten ein.

26. In der folgenden Tabelle finden Sie verschiedene Orte, die sich auf dem Irchel Campus befinden. Bewerte sie, wie weit diese Orte voneinander entfernt sind. Verwenden Sie dabei eine 1 für sehr nahe Orte und 7 für sehr weit entferne Orte. Bitte raten Sie nicht bei Orten, die Sie nicht kennen, sondern lassen Sie diese Felder einfach frei.

	Strickhof	In-formatik-dienste	Mensa	Blaue Kuh	Kopierer	Lichthof	Info-schalter	Infosäule	Y35-F-51
Strickhof	X								
Informatik-dienste	X	X							
Mensa	X	X	X						
Blaue Kuh	X	X	X	X					
Kopierer	X	X	X	X	X				
Lichthof	X	X	X	X	X	X			
Info-schalter	X	X	X	X	X	X	X		
Infosäule	X	X	X	X	X	X	X	X	
Y35-F-51	X	X	X	X	X	X	X	X	X

27. Nennen Sie drei Punkte die Sie besondert Gut bzw. Hilfreich am mExplorer fanden?

28. . Nennen Sie drei Punkte die Sie besondert Störend bzw. Schlecht am mExplorer fanden?

D

Inhalt der CD-ROM

Die zu dieser Diplomarbeit gehörende CD-ROM beinhaltet neben der Arbeit als PDF-Datei auch das geänderte MobileGame System, dessen Dokumentation und eine Reihe zusätzliche Daten. Eine Übesicht über den Inhalt gibt Tabelle D.1.

Verzeichnis	**Inhalt**
Diplomarbeit	Diplomarbeit als PDF-Datei
Javadoc	Dokumentation des MobielGame Clients
LaTeX	LaTeX-Dateien der Diplomarbeit
Literatur	Für die Arbeit verwendete Literatur im PDF Format
Messresultate	Resultate der durchgeführten Messungen
MobileGame	Abgeändertes MobileGame System
Prototyping	Für die Messungen verwendete Prototypen
Zusammenfassung	Zusammenfassung der Diplomarbeit (Deutsch und Englisch)

Tabelle D.1: Inhalt der CD-ROM

E

CD-ROM