Institut für Informatik
Universität Zürich

Untersuchung und Entwicklung von effizienten History-Funktionalitäten bezüglich Navigation anhand des mobileGame

Diplomarbeit

Ante Nikola Bujas
Schweighofstrasse 262
8055 Zürich
Matrikelnummer: 99-707-404

Betreuer: Prof. Dr. Gerhard Schwabe
Betreuender Assistent: Christoph Göth

Abgabe der Arbeit: 20. Oktober 2006

1 Einleitung

1.1 Hintergrund

Am Institut für Informatik der Universität Zürich wird unter der Leitung von Christoph Göth an einem Spiel namens mobileGame gearbeitet. Dieses Spiel soll neuen Studenten das kennen lernen des Campus der Universität vereinfachen. Die Studenten erhalten einen PDA[1] auf dem die ganze Umgebung und ihre aktuelle Position dargestellt sind. Die Client Software auf dem PDA stellt den Studenten verschiedene Aufgaben wie zum Beispiel das Auffinden eines bestimmten Raumes an der Universität. Mit Hilfe der schon implementierten Funktionalitäten des mobileGame soll es den Studenten möglich sein, sich in ihrer Umgebung zu orientieren, die verschiedenen Orte zu finden und die gestellten Aufgaben zu erfüllen.

Frühere Benutzertests und Analysen haben gezeigt, dass die Darstellung der aktuellen Position auf dem PDA zwar korrekt und nützlich, aber in vielen Fällen nicht ausreichend ist. Auch wenn die aktuelle Position bekannt ist, so ist der Weg dahin schwer zu erkennen bzw. schwer nachvollziehbar, was zu erheblichen Orientierungsschwierigkeiten führen kann. Deswegen soll in dieser Diplomarbeit das Spiel mit einer so genannten zusätzlichen History-Funktionalität erweitert werden. Dabei geht es primär darum, den zurückgelegten Weg auf dem PDA adäquat darzustellen um die Orientierung allgemein zu vereinfachen. Den Benutzern soll aufgezeigt werden, über welchen Weg sie zu ihrer aktuellen Position gekommen sind.

1.2 Aufgabenstellung

In dieser Arbeit werden die Grundlagen und die Literatur zu History-Daten bezüglich Navigation aufgearbeitet und festgehalten. Dabei werden unter anderem vorhandene und etablierte Beispiele aus der Praxis angeschaut und analysiert. Die Grundlagen der Positionsbestimmung werden auch kurz angeschnitten. Weiter wird das mobileGame so erweitert, das die History-Daten bezüglich Navigation adäquat ersichtlich sind. Wie die Darstellung erfolgt, wird im Laufe der Arbeit ermittelt. Die neu entwickelten Funktionalitäten werden von mehreren Benutzern gestestet und evaluiert. Die Benutzertests dienen dazu, eine optimale Lösung für die Darstellung der History-Daten bezüglich der Navigation zu finden, welche mit grosser Wahrscheinlichkeit eine Mischung mehrerer Lösungsansätze ist. Am Schluss wird festgehalten welche Lösungen warum genommen wurden und welche Experimente für welche Zwecke durchgeführt wurden. Zudem wird eine persönliche Bewertung durchgeführt.

[1] Personal Digital Assistant

1.3 Struktur der Diplomarbeit

Die Diplomarbeit ist grob in zwei Teile gegliedert:

Im Literaturteil wird auf die verschiedenen Aspekte der Navigation eingegangen. Dabei werden die verschiedenen Verfahren zur Positionsbestimmung beschrieben. Weiter wird auf den Thematik der History-Funktionalität eingegangen und es werden die verschiedenen Konzepte zur Darstellung von History-Daten untersucht. Bezüglich der Darstellung von History-Daten im Bereich Navigation werden einzelne etablierte Beispiele aus der Praxis erklärt und aufgezeigt.

Im praktischen Teil wird im ersten Schritt die vorhandene Client-Server Software des mobileGame installiert, studiert und analysiert. Dabei werden die verwendeten Java Bibliotheken angeschaut und in die Grundlagen der PDA-Programmierung eingegangen.
Im zweiten Schritt werden die Erkenntnisse aus der Theorie in die Spezifikation der Anforderung einbezogen. Danach wird mit der Entwicklung begonnen. Die neu entwickelten Funktionalitäten werden mehreren Tests unterzogen und die Ergebnisse in Testberichten festgehalten. Bei den Benutzertests wird auch auf die Benutzerfreundlichkeit geachtet. Die Resultate und Auswertung dieser Benutzertests werden ebenfalls festgehalten.
Im dritten und letzten Schritt findet eine persönliche Bewertung der entwickelten und implementierten Funktionalitäten und Komponenten statt, und es wird auf mögliche Verbesserungen wie auch Weiterentwicklungen hingewiesen.

2 Navigation

2.1 Definition und Begriffe

Der Begriff „Navigation" kann folgendermassen definiert werden:

„Navigation ist die Steuermannskunst zu Meer (Nautik), zu Land und in der Luft. Allgemeiner bezeichnet sie das sich Zurechtfinden in einem geografischen Raum, um einen bestimmten Ort zu erreichen[2]."

W. Boehm [BOEHM] umschreibt das Wort „Navigation" als jede Maßnahme (Beobachtung, Messung und Auswertungsmethode), mit welcher der geographische Ort und/oder die Bewegung eines Fahrzeuges ermittelt werden kann.
Der Begriff Navigation kommt von lat. *navis,* Schiff und bezog sich früher quasi ausschließlich auf die Nautik, also die Lehre von der Schifffahrt, und war mit dieser gleichbedeutend. Das wesentliche Merkmal der Navigation generell ist die Bestimmung des momentanen Standortes beziehungsweise der momentanen Position sowie auch von Richtungen, um einen bestimmten festgelegten Ort auf direkten Weg zu erreichen. Die Bestimmung des momentanen Standortes bzw. der aktuellen Position eines Objektes wird auch Ortung genannt. Gemäß Bauer [BAUER] ist die Bestimmung des momentanen Ortes eines beweglichen Objektes auf dem Wasser, in der Luft oder auf dem Land das primäre Ziel der Ortung. Ortung ist also Teil der Navigation. Bei der Ortung werden gemäß Bauer [BAUER] die momentanen Koordinaten eines in Bezug zum Erdkörper sich bewegenden Punktes gesucht. Das Ziel der Ortung ist, die Koordinaten von Punkten zu bestimmen. Das Navigieren, die Tätigkeit der Navigation wird in drei Teilbereiche unterteilt:

- Bestimmung der geographischen Position durch Ortung nach verschiedenen Methoden.

- Berechnen des Weges zum Ziel.
  Ausgehend von der aktuellen Position und des vorgegebenen Zieles wird ein möglichst direkter bzw. kürzester Weg ermittelt.

- Führung zu diesem Ziel.
  Durch die laufende Ermittlung der aktuellen Position und der Kenntnis des Startpunktes und des Zieles wird für jede aktuelle Position der Weg zum Ziel aktualisiert.

2.2 Geschichte der Navigation

Die ersten Verfahren zur Navigation wurden ungefähr vor 6000 Jahren in Indien und zeitnah auch in Ägypten und im Libanon entwickelt. Die ersten überlieferten Karten stammen aus den Hochzivilisationen. Stark wachsende wirtschaftliche Verknüpfungen aber auch verschiedene politische Interessen zwischen den Kulturen förderten die kartographische Beschreibung von Handelsrouten und

[2] http://de.wikipedia.org/wiki/Navigation

Wegstrecken in der Antike. Die Aufkommende Seefahrt motivierte die betroffenen Kulturen neue Navigationshilfen wie Sonnenstand, Gestirne, Koppelnavigation und Küstenmerkmale zu benutzen. Diese Verfahren der Koppelnavigation und der Astronavigation wurden anfangs primär für die Seefahrt entwickelt und benutzt. Später dann auch für Expeditionen. Die Koppelnavigation wurde unter Einbezug weiterer Faktoren wie Strömung, Abdrift, Windrichtung und Beobachtungen von Vögeln ständig weiter entwickelt. Etwa im 11.Jahrhundert wurde der Kompass in China erfunden. Im 12.Jahrhundert wurde der Kompass erstmals auch in Europa gesichtet. Im 13./14. Jahrhundert wurden erstmals detaillierte Seekarten und Seehandbücher erstellt [ZIMMER]. Ab dem 15. Jahrhundert erfolgten weitere Perfektionierungen der Koppelnavigation und ein langsames Aufkommen von komplexen Positionsberechnungen. Es entstanden zahlreiche Weltkarten und es wurden Loggen und Quadranten eingesetzt.

- **Loggen**
  Ein Log (Holzklotz) ist in der Seefahrt und Navigation ein Messgerät zur Bestimmung der Fahrtgeschwindigkeit von Wasserfahrzeugen. Es zeigt die im Wasser zurückgelegte Strecke oder die Geschwindigkeit an.

- **Quadranten**
  Ein Quadrant ist ein astronomisches Instrument, mit dem die Höhen von Gestirnen ermittelt werden.

Im Jahre 1700 erfand Isaac Newton ein Gerät zur Winkelmessung. Bei diesem Gerät, welches mit Hilfe von Spiegeln funktionierte, handelte es sich um den Vorgänger des Sextanten, den 1731 John Hadley erfand.

- **Sextant**
  Ein Sextant ist ein genaues Winkelmessinstrument. Er wird im Wesentlichen dazu gebraucht, um den Winkel zwischen dem Horizont und einem Gestirn zu messen. Bei Sonne und Mond wird dabei zwischen Ober- und Unterrand unterschieden. Bei Planeten und Fixsternen, die uns sowieso nur als kleiner Punkt am Himmel erscheinen ist das unnötig.

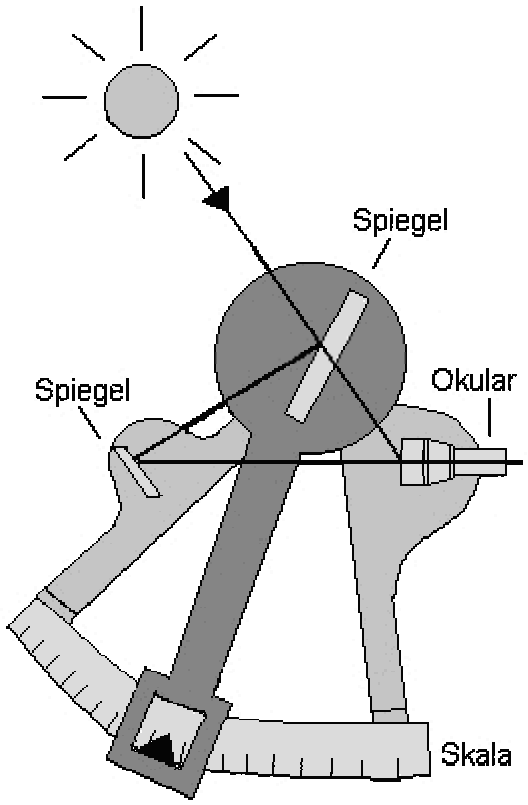

Abbildung 2. 1: Der Sextant[3]

Nachdem 1837 die astronomische Standlinienmethode erfunden wurde, konnte ab 1899 die Funknavigation und ab 1941 die Trägheitsnavigation erfolgreich eingesetzt werden. Im Jahre 1973 wurde das satellitengestützte Navigationssystem erfunden, welches zuerst für militärische Zwecke und später auch für zivile Zwecke eingesetzt wurde.

2.3 Arten der Navigation

Es gibt verschiedenen Arten der Navigation die sich im Wandel der Zeit durch immer neue Technologien und neues Wissen weiterentwickelt haben. Die modernen Navigationsarten basieren größtenteils auf den älteren Navigationsmethoden bzw. Messtechniken, die durch die technologischen Fortschritte weiter verfeinert wurden. Oft kombinieren moderne Navigationsarten Ideen beziehungsweise Konzepte älterer Navigationsmethoden. Folgend werden die wichtigsten Navigationsarten kurz beschrieben, wobei auf die Funknavigation speziell eingegangen wird, da sie im mobileGame primär benutzt wird und heutzutage bei vielen bekannten Anwendungen wie zum Beispiel dem GPS[4] eingesetzt wird.

[3] http://www.phantastik-kalender.de/navigation/navigation.html
[4] General Positioning System

2.3.1 Koppelnavigation

Koppelnavigation ist die laufende Ortsbestimmung (Ortung) eines Schiffs oder Flugzeugs durch Messung von Kurs, Fahrt (Geschwindigkeit) und Zeit.[5] Um die genaue Position zu ermitteln musste der Navigator drei Dinge wissen: Seinen Ausgangspunkt, die Geschwindigkeit und seinen Kurs. Um die Geschwindigkeit zu ermitteln beobachteten die Seefahrer zum Beispiel wie lange ihr Schiff brauchte um einen Gegenstand zu passieren, den sie vom Bug ins Wasser geworfen hatten. War die Geschwindigkeit dem Navigator bekannt, so konnte er die Strecke berechnen die das Schiff an einem Tag zurückgelegt hat. Auf der Seekarte zog er eine Linie, die seinen Kurs entsprach. Die Koppelnavigation oder (engl.) Dead Reckoning wurde mit der Zeit durch verschiedene zusätzliche Messverfahren erweitert. Christoph Kolumbus und weitere Zeitgenossen kombinierten die Koppelnavigation mit dem Einbezug von Strömungen und Winden.

2.3.2 Terrestrische Navigation

Bei der terrestrischen Navigation werden Landmarken (auffallende, markante Punkte bzw. Objekte wie zum Beispiel Bäume, Kirchen, Leuchttürme oder Burgen) zur Positionsbestimmung verwendet. Die verschiedenen Landmarken sind auf den Karten meist speziell gekennzeichnet. Die terrestrische Navigation beinhaltet verschiedene Methoden zur Berechnung der aktuellen Position[6]:

- Optische Peilung
- Berechnung des Höhenwinkels
- Berechnung des Horizontalwinkels
- Radarpeilung
- Doppel Horizontalwinkelmessung

2.3.3 Sichtnavigation

Die Sichtnavigation basiert auf den Vergleich der Karte mit dem sichtbaren Gelände. Die Navigation erfolgt nach dem Auge. Sie ist einer der ältesten Navigationsarten. Die Sichtnavigation wird sehr häufig in der Fliegerei bei Kleinflugzeugen verwendet. Voraussetzung für eine zuverlässige Sichtnavigation sind gute Sichtverhältnisse sowie aktuelle und detaillierte Karten. Bei schlechtem Wetter und daher schlechten Sichtverhältnissen oder ungenauen Karten ist eine Sichtnavigation nur beschränkt möglich. Die Sichtnavigation setzt auch eine gute Vorbereitung der geplanten Reise voraus. So ist die Erstellung einer Kurslinie in der Fliegerei eine Selbstverständlichkeit. Bei einer sorgfältigen Vorbereitung sollte sich die Landschaft im Flug geradezu „kartengerecht" aufrollen und die Kontrollpunkte oder Auffanglinien müssten sich mühelos identifizieren lassen [MAIER].

[5] http://de.wikipedia.org/wiki/Navigation
[6] http://www.astrosail.de/de/static/tutorial/terrstl1.php?cat=42

2.3.4 Astronomische Navigation

Die astronomische Navigation basiert auf Beobachtungen von Gestirnen wie Mond, Sonne, Fixsterne und andere Planeten. Mittels Sextanten, Quadranten und anderen Hilfsmitteln werden die Winkel zu verschiedenen Gestirnen sowie die Horizontal- und Höhenwinkeln zu ausgewählten Gestirnen ermittelt. Zwei Gestirne reichen für eine exakte Ortung aus [KNOPP]. Sonnenaufgang und Sonnenuntergang zeigen Osten und Westen an. Bei Tagesanbruch kann ein Navigator feststellen, wie weit die Sonne im Vergleich mit den langsam verblassenden Sternen gewandert ist. Nachts kann die Position anhand des Polarsterns, der bei Dunkelheit fast genau über dem Nordpol steht, ermittelt werden. Grosses Manko dieser Navigation ist die Abhängigkeit vom Wetter. Ist der Himmel weitgehend bedeckt, so ist keine astronomische Navigation mehr möglich.

2.3.5 Trägheitsnavigation

Die Trägheitsnavigation konzentriert sich auf die Veränderungen von Beschleunigungen. Diese Art der Navigation wird auch als Inertiales Navigationssystem bezeichnet. Die Beschleunigungsveränderungen werden mit hochsensiblen Beschleunigungsmessgeräten gemessen. So können kleinste Positionsveränderungen wahrgenommen werden. Die Trägheitsnavigation wird heutzutage in der Luftfahrt und teilweise auch in U-Booten verwendet.

2.3.6 Satellitennavigation

Die Satellitennavigation ist eine der neusten Navigationsart. Diese Navigationsart benötigt folgende Infrastruktur:

- Satelliten die um die Erde kreisen und Signale zur Erde senden.
- Kontrollzentren welche die Signale empfangen und auf Korrektheit überprüfen.
- Empfänger welche die Signale der Satelliten empfangen und auswerten, und den Benutzern ihre aktuelle Position darstellen.

Die bekannteste Anwendung der Satellitennavigation ist das GPS, welches für verschiedene Zwecke eingesetzt werden kann. Die Satellitennavigation benutzt ähnliche Verfahren wie die Funknavigation. Diese und andere Verfahren werden im folgenden Abschnitt der Funknavigation genauer angeschaut. Die Satellitennavigationssysteme werden üblicherweise von den Funknavigationssystemen abgegrenzt. Da aber bei der Satellitennavigation auch Radiowellen zur Positionsbestimmung eingesetzt werden, kann diese Navigationsart auch zur Funknavigation gezählt werden.

2.3.7 Funknavigation

Bei der Funknavigation wird die aktuelle Position mittels Radiowellen bestimmt. Diese Radiowellen werden von mehreren Sendestationen ausgesandt und von mehreren Empfängern empfangen. Die Radiowellen enthalten verschiedene Informationen wie Sendezeit, Sendeort und Ausbreitungsgeschwindigkeit. Die Empfänger können mit Hilfe dieser Radiowellen bzw. Signale die Position bestimmen. Die in der Praxis häufig verwendeten Verfahren sind:

- Messung der Laufzeit (TOA, Time of Arrival)
- Laufzeitdifferenz (TDOA, Time Difference of Arrival)
- Eingangswinkel des Signals (AOA, Angel of Arrival)
- Berechnung der Signalstärke

Alle genannten Verfahren benutzen als Basis folgende mathematische Konzepte:

- **Trilateration (Distanzmessung)**
  Die Trilateration ist eine Methode zur Bestimmung der relativen Positionen von Punkten mittels Verwendung der geometrischen Eigenschaften von Dreiecken. Dabei müssen mindestens zwei Referenzpunkte eines Dreiecks bekannt sein.

- **Triangulation (Winkelmessung)**
  Bei der Triangulation wird die gegenseitige Lage von Dreieckspunkten durch die Messung der Dreieckswinkel bestimmt. Dabei muss mindestens die Länge einer Dreiecksseite bekannt sein [BAUER].

2.3.7.1 Messungen der Laufzeit (TOA)

Beim TOA-Verfahren (Time of Arrival) wird die Distanz zwischen Sendestation und Empfänger anhand der Laufzeit und der Ausbreitungsgeschwindigkeit des Signals berechnet. Mit mindestens zwei Sendestationen kann somit die aktuelle Position des Empfängers bestimmt werden. Die Positionen der Sendestationen müssen bekannt sein [ZHAO].

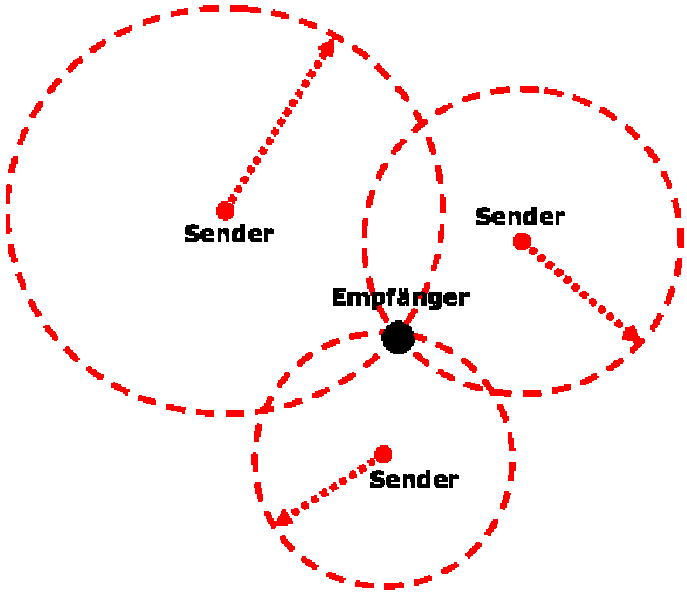

Abbildung 2. 2: Das TOA – Verfahren[7]

Bei diesem Verfahren gibt es eine Einweg und eine Zweiwegmethode:

- Bei der Einweg-Methode müssen die Sender und die Empfänger synchronisiert sein. Zudem müssen die Empfänger die Sendezeitpunkte kennen. Dies kann entweder durch festgelegte Sendezeitpunkte oder durch einen Zeitstempel im Signal erreicht werden.

- Bei der Zweiweg-Methode findet eine Reflexion des Signals beim Empfänger statt (Echo). Deswegen ist hier keine Synchronisation nötig.

Die TOA-Methode wird primär beim GPS [GIBSON] und beim Radar angewendet.

[7] Seminar Anwendung von ubiquitären, mobilen Kleingeräten, Prof. C. Cap, Referenten: Ante Bujas, Christian Breuel, Marco Stadler, Universität Zürich, 2005. Leicht modifiziert

2.3.7.2 Messung der Zeitdifferenz (TDOA)

Die TDOA-Methode (Time Difference of Arrival) ermittelt die Position des Empfängers der Signale anhand von Hyperbeln. Mindestens zwei Sendestationen senden gleichzeitig Signale aus. Der Empfänger misst die Zeitdifferenz dieser Signale. Mit den Positionsangaben der Sendestationen lässt sich eine hyperbolische Kurve berechnen. Mehrere Messungen zu verschiedenen Referenzpunkten ergeben dann einen Schnittpunkt, der die aktuelle Position des Empfängers darstellt.

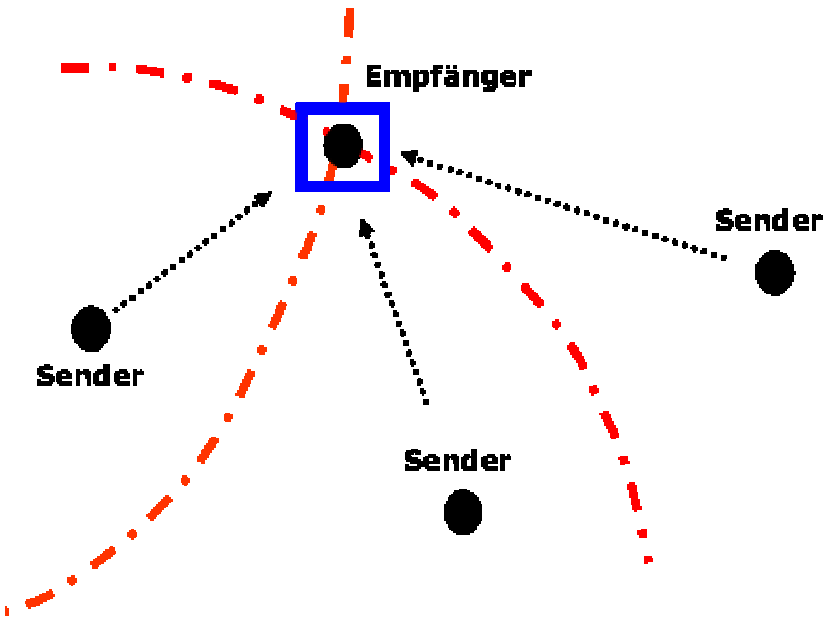

Abbildung 2. 3: Das TDOA – Verfahren[8]

Anwendung findet die TDOA Methode beim GSM[9].

2.3.7.3 Messung des Eingangswinkels des Signals (AOA)

Bei der AOA-Methode (Angle of Arrival) misst der Empfänger den Eingangswinkel der eintreffenden Signale. Die Herkunftsrichtung der Signale wird bestimmt. Sind die Richtungen zweier Sendestationen bekannt, so kann die Position des Empfängers bestimmt werden. Um mehrere Signale aus verschiedenen Richtungen zu empfangen, muss der Empfänger entweder eine drehende Antenne haben oder mehrere Antennen mit verschiedenen Ausrichtungen.
Die Positionsbestimmung mittels Winkelmessung wird in zwei Verfahren unterteilt:

[8] Seminar Anwendung von ubiquitären, mobilen Kleingeräten, Prof. C. Cap, Referenten: Ante Bujas, Christian Breuel, Marco Stadler, Universität Zürich, 2005. Leicht modifiziert
[9] Global System For Mobile Communication

- **Richtempfangsverfahren (Eigenpeilung)**
  Bei der Eigenpeilung peilt der Signalsender selber einen externen Empfänger an und ermittelt damit seine aktuelle Position.

- **Senderichtverfahren (Fremdpeilung)**
  Bei der Fremdpeilung wird ein externes Peilgerät benutzt, mit dem sich die Herkunftsrichtung der vom Sender ausgesandten Signale ermitteln lässt. Daraus lässt sich die aktuelle Position des Senders bestimmen.

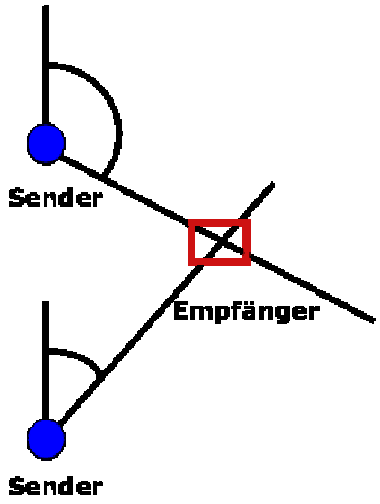

Abbildung 2. 4: Das AOA – Verfahren[10]

Die AOA-Methode wird zum Beispiel beim Radar und beim GSM benutzt.

2.3.7.4 Berechnung der Signalstärke

Dieses Verfahren zur Positionsbestimmung ist dem TOA Ansatz sehr ähnlich. Die Sendestationen senden Signale aus, deren Stärke mit zunehmender Entfernung durch Dämpfung abnimmt. Die Distanz zwischen Sender und Empfänger kann somit über die einzelnen Signalstärken ermittelt werden. Eine exakte mathematische Beschreibung der Signalstärke in Abhängigkeit des Abstands existiert zwar in der Theorie, in der Praxis unterliegt die Signalstärke aber vielen externen Einflüssen [DORNBUSCH].

Zu den externen Einflüssen gehören:

- Elektronische Geräte (Kraftwerke, Mobilfunkantennen, Mikrowellen, Mobiltelefone usw.)

- Einrichtungen im Gebäude (Wände, Fenster, Möbel usw.)

- Hindernisse im Gelände (Gebäude, Berge, Vegetation usw.)

[10] Seminar Anwendung von ubiquitären, mobilen Kleingeräten, Prof. C. Cap, Referenten: Ante Bujas, Christian Breuel, Marco Stadler, Universität Zürich, 2005. Leicht modifiziert

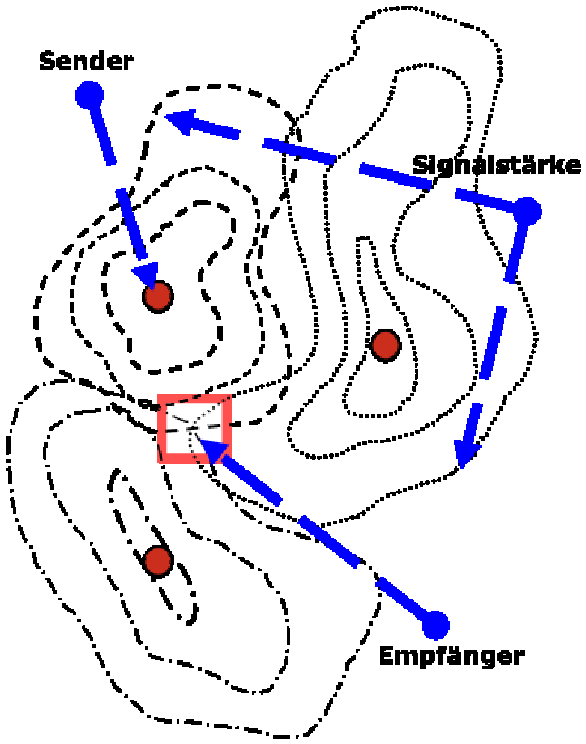

Abbildung 2. 5: Positionsbestimmung mit der Signalstärke[11]

Die Ortung über die Signalstärke wird auch beim mobileGame benutzt. Dort erfolgt die Positionsbestimmung über das WLAN[12].

[11] Seminar Anwendung von ubiquitären, mobilen Kleingeräten, Prof. C. Cap, Referenten: Ante Bujas, Christian Breuel, Marco Stadler, Universität Zürich, 2005. Leicht modifiziert.
[12] Wireless Local Area Network

2.3.7.5 Überblick der verschiedenen Messverfahren

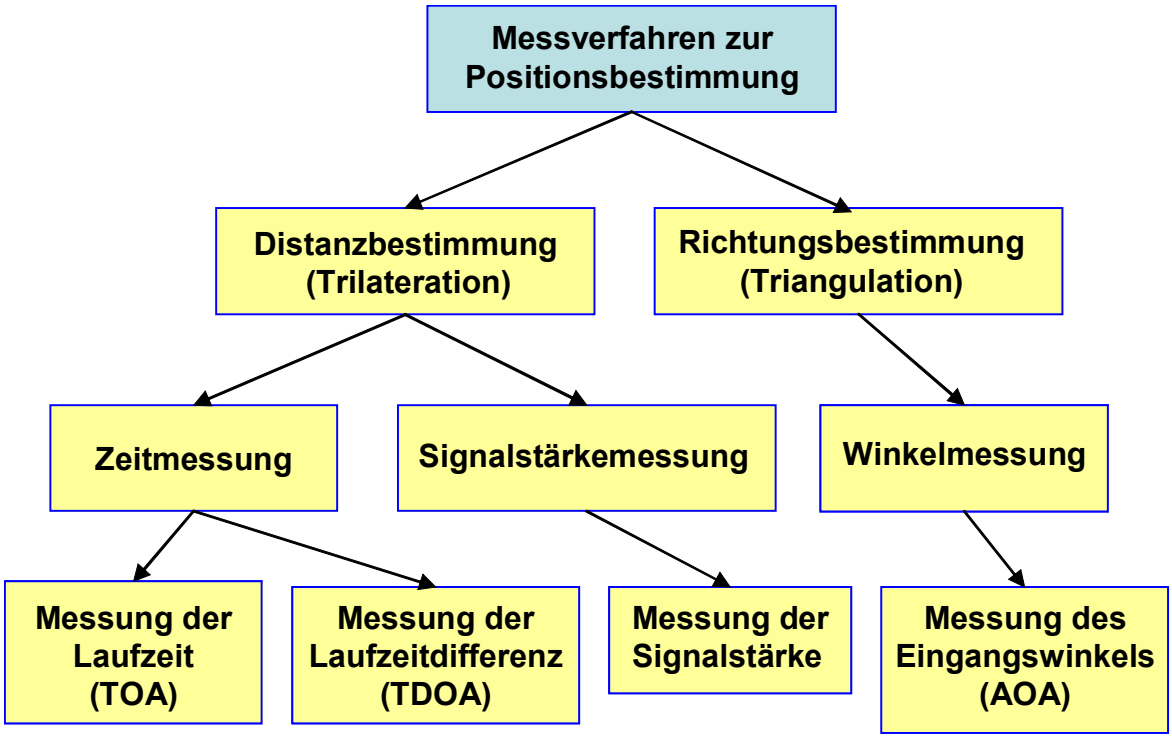

Abbildung 2. 6: Die Messverfahren zur Positionsbestimmung

2.3.7.6 Problematik der funkbasierten Ortungsverfahren

Ein essentielles Problem besteht bei funkbasierten Ortungssystemen darin, dass sich hier die Funksignale notwendigerweise weitgehend ungerichtet im Raum ausbreiten müssen. Dies ist erforderlich, da die relative Lage zwischen mobiler und ortsfester Einheit vorab nicht bekannt ist.

Dies hat zu Folge, dass die ungerichtet ausgesendeten Signalen nicht nur über den direkten und kürzesten Weg von Sender zum Empfänger gelangen, sondern sie erreichen oft über größere Umwege den Empfänger. Diese Umwege führen in einigen Fällen zu großen Ungenauigkeiten der Positionsbestimmung. Nicht nur Hindernisse zwischen Sender und Empfänger führen zu Umwegen, sondern auch Reflexionen der Signale. Diese Mehrwegproblematik ist insofern kritisch, da sie der maßgebliche Faktor bezüglich der erreichbaren Genauigkeit von funkbasierten Ortungssystemen ist.

2.3.8 Überblick der verschiedenen Navigationsarten

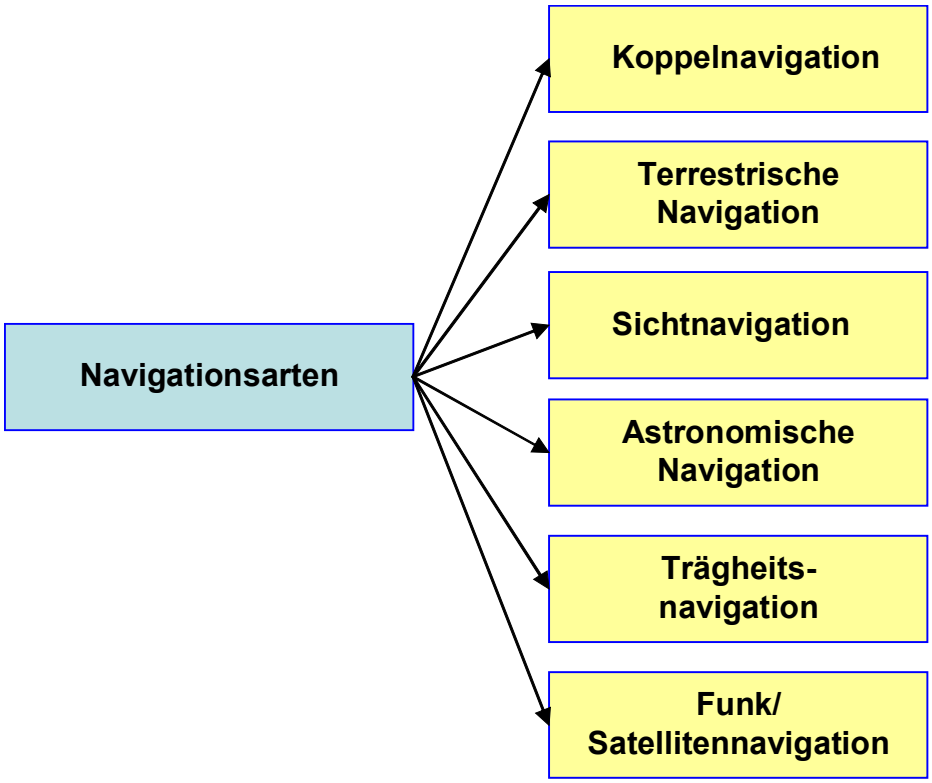

Abbildung 2. 7: Die Navigationsarten

2.4 Weitere Aspekte zur Positionierung

In Abschnitt 2.3 wurden die verschiedenen Navigationsarten und ihre Techniken zur Positionsbestimmung erläutert. Neben den verschiedenen Verfahren zur Positionsbestimmung gibt es noch weitere Aspekte bezüglich der Positionsbestimmung, welche in diesem Abschnitt beschrieben werden.

2.4.1 Absolute und relative Positionierung

Bei der Bestimmung einer Position wird zwischen absoluter und relativer Positionsbestimmung unterschieden.

2.4.1.1 Absolute Positionierung

Bei der absoluten Positionsbestimmung wird die aktuelle Position in einem absoluten Bezugssystem bestimmt und die entsprechenden Koordinaten zur Beschreibung der Position ermittelt Absolute Ortungssysteme sind sich ihrer Position bewusst. Sie ermitteln ihre Position in Echtzeit [WIENHOLZ]. Absolute Ortungssysteme sind unabhängig voneinander, das heißt, um eine Position mittels absoluter Positionierung zu ermitteln, sind vergangene Referenzpositionen nicht nötig. Alle GPS-Systeme basieren zum Beispiel auf dem absoluten Positionierungsverfahren. Alle GPS-Empfänger liefern die

Positionsinformationen in Form von Längen- und Breitengraden und der zugehörigen Höheninformation des Empfängers in einem eindeutig festgelegten Koordinatensystem.

2.4.1.2 Relative Positionierung

Relative Positionierungssysteme erlauben es den zu lokalisierenden Objekten die Verwendung eigener Referenzsysteme. Die Bestimmung der Positionskoordinaten erfolgt dabei relativ zu einem vorgegebenen Bezugspunkt. Relative Positionierungssysteme werden in verschiedenen Navigationssystemen in der Luft- und Schifffahrt verwendet. Bei Trägheitsnavigationssystemen werden mittels eines Kreiselkompasses und Beschleunigungssensoren die relative Lage im Raum und die Bewegung erfasst. In Kombination mit der Kenntnis der Ausgangsposition kann dann die aktuelle Position rechnerisch bestimmt werden.

Die Kombination zwischen absoluter und relativer Positionierung ist möglich. Absolute Positionsangaben werden in relative Positionsangaben umgewandelt. Dabei wird die Position relativ zu einem anderem Referenzpunkt, dessen absolute Position bekannt ist, beschrieben. Auch die Umwandlung von relativen zu absoluten Positionsangaben ist möglich. Dabei müssen mehrere relative Positionen zu mehreren Referenzpunkten bekannt sein, deren absolute Positionen bekannt sind.

2.4.2 Selbst- und fernortende Positionssysteme

Die selbstortende Positionsbestimmung basiert auf der eigenständigen Positionsberechnung der Ortungssysteme. Selbstortende Systeme können demnach die Positionierung selber vornehmen. Vorraussetzungen für eine Selbstortung sind genügend Energiereserven und allgemein eine leistungsstarke Recheneinheit (Prozessor). Ein großer Vorteil selbstortender Systeme ist die gewährleistete Sicherheit. Da sie die Positionsbestimmung selber vornehmen, sind keine externen Hilfssysteme notwendig, und die ermittelten Informationen sind nur dem eigentlichem Ortungssystem bekannt.

Dem gegenüber steht die fernortende Positionsermittlung. In fernortenden Systemen nehmen externe Systeme die Ortung vor. Somit werden die einzelnen Ortungsknoten bezüglich Rechenleistung, Speicherbedarf und Energiebedarf massiv entlastet. Auf der anderen Seite werden die berechneten Informationen in einem Netzwerk von äußeren Systemen gespeichert. Dieser Umstand bringt erhebliche Nachteile bezüglich des Datenschutzes und allgemeinen Sicherheitsfragen mit sich. Fernortende Systeme sind oft kostengünstiger als selbstortende Systeme.

2.4.3 Globale Lokalisierung und lokale, kleinräumige Lokalisierung

Die globale Lokalisierung („Wide Area Location“) bestimmt die Positionen von Objekten die sich überall auf der Welt befinden.

Beispiel: GPS

Bei der lokalen, kleinräumigen Lokalisierung („Local Area Location“) kann die Position nur in einem vordefinierten, bestimmten Raum ermittelt werden, in dem auch eine entsprechende Infrastruktur vorhanden ist.

Beispiel: Active Badges

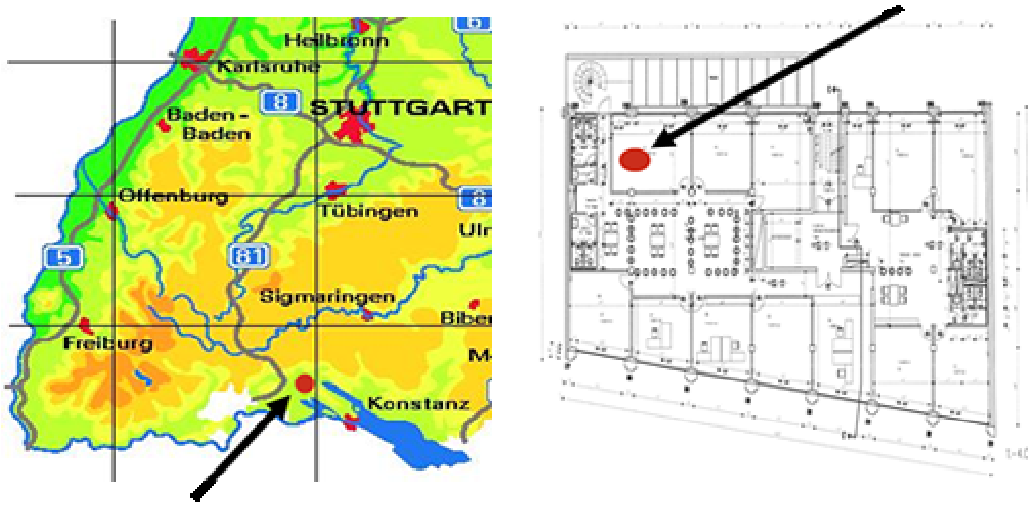

Abbildung 2. 8: Wide Area und Local Area Location[13]

2.5 Führung zum Ziel

Neben der eigentlichen Bestimmung der aktuellen Position beinhaltet die Navigation auch die Führung eines Objektes zu einem Ziel bzw. zum nächsten Wegpunkt. Primär geht es darum, einen möglichst optimalen Weg von einem Punkt A zu Punkt B zu finden und diesen verständlich aufzuzeigen. Eine optimale und verständliche Wegführung hängt von vielen Faktoren ab, welche sich untereinander ergänzen. Ein wichtiger Faktor stellt die beschriebene Positionsbestimmung dar. Die genaue Kenntnis der aktuellen Position wie auch die Position des Zieles, bzw. des nächsten Zwischenzieles, erleichtert die Wegführung enorm. Neben einer genauen Positionsbestimmung können noch andere Faktoren die Wegführung unterstützen. Diese beziehen sich auf die Art der Beschreibung des zum Ziel führenden Weges. So können verschiedene Annotationen wie Texte, Bilder, Symbole und Klänge eine Wegführung stark erleichtern. Pfeile auf einer Karte können zum Beispiel sehr einfach die Richtung zu einem Ziel zeigen. Ein Text kann den zum Ziel führenden Weg sehr detailliert beschreiben. Eine Verbindungslinie zwischen zwei Symbolen, welche je einen wichtigen Wegpunkt darstellen, kann sehr einfach den Weg aufzeigen. Oft werden die verschiedenen Annotationen miteinander kombiniert. So werden zum Beispiel Karten oft mit unterschiedlichsten Beschreibungen und Hinweisen versehen. Navigationsgeräte für den Straßenverkehr kombinieren Klänge mit Visualisierungen. So wird eine Richtungsänderung nicht nur mit Pfeilen angezeigt, sondern zusätzlich durch eine akustische Ansage angekündigt. Solche Sprachführungen werden häufig verwendet um den Fahrer nicht durch Visualisierungen auf dem GPS-Display abzulenken. In der Praxis werden die zu einem Ziel führenden Hilfsmittel oft nicht alleine verwendet, sondern mit

[13] Seminar Anwendung von ubiquitären, mobilen Kleingeräten, Prof. C. Cap, Referenten: Ante Bujas, Christian Breuel, Marco Stadler, Universität Zürich, 2005.

weiteren Informationsquellen kombiniert. So achtet ein Autofahrer nicht nur auf sein Navigationsgerät, sondern er betrachtet auch die verschiedenen Verkehrsschilder. Personen die sich in einer für sie unbekannten Stadt bewegen orientieren sich nicht nur mit Hilfe von Stadtkarten, sondern auch durch Befragung von ortsansässigen Personen, durch Hinweisschilder usw.

Die Wegführung zu einem Ziel geschieht nicht durch ein Hilfsmittel allein, sondern mit Hilfe einer Kombination von verschiedenen Navigationshilfen. Entscheidend ist welche Hilfsmittel zur Wegfindung eingesetzt und miteinander kombiniert werden. Wichtig ist vor allem, dass die Art und Weise der Beschreibung des Weges an die jeweilige Situation angepasst ist.

3 History-Funktionalität bezüglich Navigation

Im vorhergehenden Abschnitt wurde auf die verschiedenen Aspekte der Navigation beziehungsweise der Positionsbestimmung eingegangen. In diesem Kapitel geht es nun primär um die Thematik der so genannten History-Funktionalität. Es wir erklärt was History-Funktionalitäten sind, wie diese zusammengesetzt sind, wo sie eingesetzt werden und vor allem inwiefern sie bezüglich der Navigation ihre Verwendung finden. Dazu werden größere etablierte Beispiele aus der Praxis genauer angeschaut. Diese Beispiele dienen dazu, die Verwendung von History-Funktionalitäten adäquat aufzuzeigen, und auch auf die verschiedenen Darstellungsarten der History-Daten bezüglich Navigation hinzuweisen.

3.1 History

Der Begriff „History", auf Deutsch Geschichte, ist in der Praxis sehr oft zu hören. Meistens spricht man nur von der History einer gegebenen Problematik. Aber genau dieser Begriff wird sehr vielseitig umschrieben. Die Bereiche, in denen der Begriff History vorkommt, sind sehr breit gefächert. Einige Beispiele die das Wort History umschreiben, sollen dies verdeutlichen:

- Eine Aufzählung von Ereignissen[14].
- Eine Erzählung.
- Eine chronologische Speicherung von Ereignissen wie vom Leben, von der Weiterentwicklung der Menschen oder einer Institution, welche meistens eine Erklärung oder einen Kommentar enthält[15].
- Eine ständige Entwicklung und Anpassung gesellschaftlich organisierter Lebewesen an ihre Umwelt[16].
- Geschichte als Vergangenes, Geschehenes, als eine Summe von Ereignissen, Handlungsabläufen, Taten und Errungenschaften der Menschheit, die, in Bezugsgeflechten zueinander stehend, sich abspielten und sukzessiv aneinanderreihten [PASCKE].
- Der Begriff History ist eine mögliche Bezeichnung für die Liste der URL's die ein Benutzer bis zum einem Zeitpunkt in der Vergangenheit besuchte[17].

Wie zu erkennen ist, gibt es viele verschiedene Umschreibungen des Begriffs der „History". Die für diese Arbeit und daher für das mobileGame passende Beschreibung beinhaltet eine Mischung der aufgezeigten Umschreibungen. Nicht zuletzt bezieht sich der History Begriff dieser Diplomarbeit hauptsächlich auf die Navigation bzw. der ermittelten Positionen. Die History bezüglich Navigation und

[14] http://www.thefreedictionary.com/history
[15] http://www.thefreedictionary.com/history
[16] http://de.wikipedia.org/wiki/Geschichte
[17] http://www.internet-kompetenz.ch/sicherheit/glossar/h.html

des mobileGames beinhaltet primär die Geschichte von Positionen eines sich in einem Raum bewegenden Objektes. Dabei handelt es sich auch um die aktuelle aber vor allem um Positionen der unmittelbaren Vergangenheit. Die vorherigen Standorte stehen primär im Vordergrund. Die History bezüglich dem mobileGame beschreibt wo sich ein Anwender überall befunden hat. In einer History bezüglich Navigation werden die Positionen eines sich in einem Raum bewegenden Objektes adäquat abgespeichert. Beim GPS und allgemein in der Navigation wird diese History auch als (engl.) Track bezeichnet.

3.1.1 Track bzw. Track Log

Ein Track, häufig auch Track Log genannt, ist eine sehr genaue Aufzeichnung eines zurückgelegten Weges. Unter einem Track versteht man auch eine automatisch generierte kontinuierliche Abfolge aufgezeichneter geografischer Koordinaten [HABERLAH]. Die einzelnen Punkte eines Tracks bzw. einer Wegaufzeichnung werden als Wegpunkte bezeichnet[18].

Wegpunkte sind Positionen, an denen sich ein Objekt befunden hat. Wegpunkte können auch spezielle geographische Eckpunkte sein[19].

Neben dem Track gibt es noch den Begriff „Route". Eine Route besteht aus einer Aneinanderreihung von bestehenden Wegpunkten in einer bestimmten, selbst festgelegten Reihenfolge. Sie ist eine Kette von Wegpunkten, in der Anfangs- und Endpunkt ausgezeichnet sind[20].

Eine Route enthält demnach nicht alle Wegpunkte wie ein Track. Eine Route zeigt nur den Weg zwischen einzelnen festgelegten Wegpunkten an, und nicht den exakt zurückgelegten Weg[21]. Die festgelegten Wegpunkte können spezielle, wichtige Positionen sein oder auch nur auffallende Geländemarken.

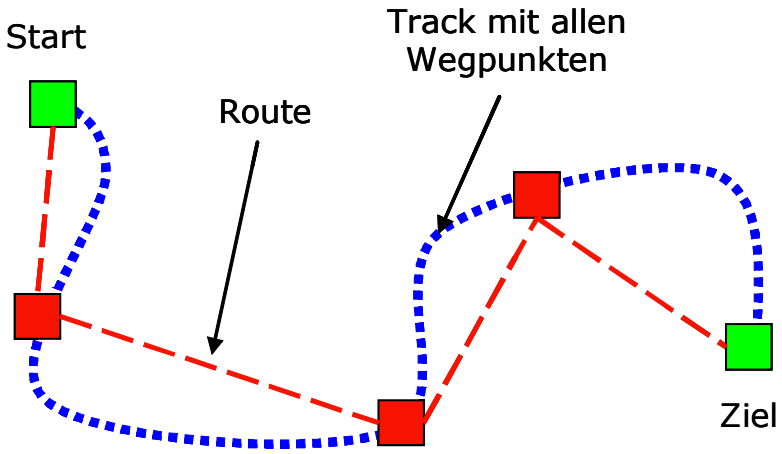

Abbildung 3. 1: Wegpunkte, Track und Route

[18] http://de.wikipedia.org/wiki/Track_Log
[19] http://service.alan-germany.de/GPS/Map500/FAQ.htm#_Toc75850528
[20] http://www.gps24.de/softtraxx_handbuch.doc
[21] http://www.kowoma.de/gps/Navigation.htm

Ein Track ist viel genauer als eine Route. Tracks werden zum Beispiel von GPS Geräten automatisch erstellt und abgespeichert. Dabei werden die einzelnen Positionen in festgelegten Zeit- und Entfernungsabständen erfasst. Diese Zeit und Entfernungsabstände stellen das primäre Kriterium der Trackerstellung dar.

3.1.1.1 Kriterium Zeit zur Trackerstellung

Wird das Kriterium Zeit benutzt, so werden nur Wegpunkte in einem gewissen festgelegten Zeitintervall im Track Log abgespeichert. Dies kann beim GPS je nach Voreinstellung des Benutzers zum Beispiel sekündlich geschehen [BURKHARDT]. Je nach Art der Benutzung und der gegeben Situation, ist ein entsprechendes Intervall zu wählen. Kleinere Zeitintervalle erzeugen einen genaueren Track, werden aber bei vielen Überschneidungen bzw. vielen nahe liegenden Wegpunkten sehr schnell unübersichtlich. Kurze Zeitintervalle eignen sich daher für sich schnell und eher gradlinig bewegende Objekte. Längere Intervalle eignen sich für Objekte die sich langsam bewegen und ihre Richtung regelmäßig ändern.

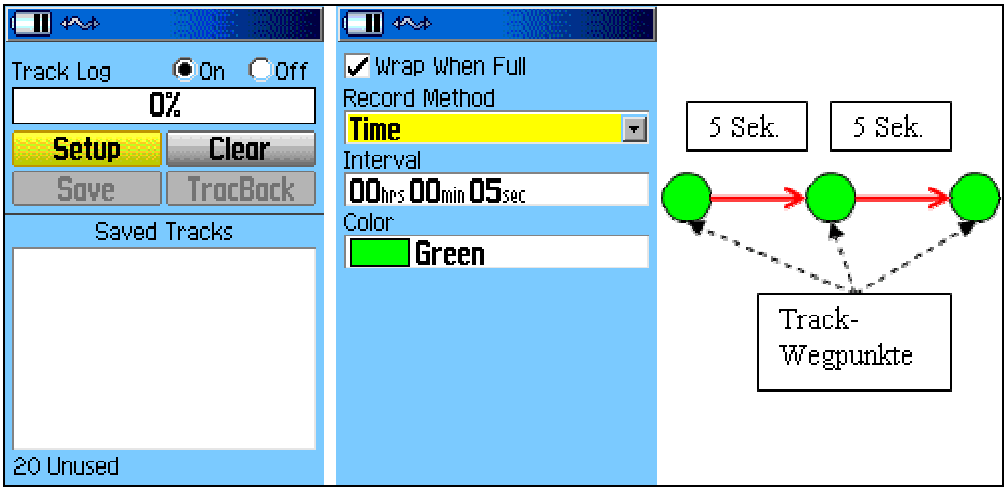

Abbildung 3. 2: Einstellung des Zeitintervalls beim GPS auf 5 Sekunden

3.1.1.2 Das Distanz-Kriterium zur Trackerstellung

Neben dem Kriterium der Zeit werden in der Praxis die Wegpunkte auch nach Abstandsintervallen gefiltert und im Track Log abgespeichert. Die Abstandsintervalle können beim GPS sehr stark variieren. Bei einer „Local Area Location", (siehe Kap. 2.4.3) können auch andere Maßeinheiten wie zum Beispiel Anzahl Pixel verwendet werden. Das Abstandskriterium eignet sich für langsam und schnell bewegende Objekte, ist aber wie das Verfahren nach Zeit anfällig bezüglich Überschneidungen.

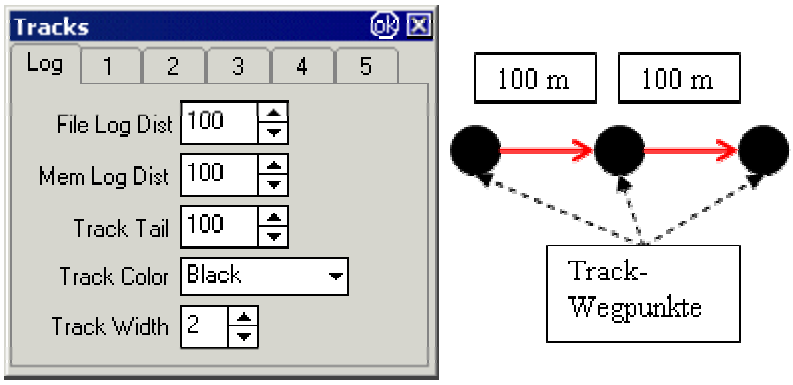

Abbildung 3. 3: Einstellung des Abstandes bezüglich den Wegpunkten[22]

3.2 History-Daten

Im letzten Kapitel wurde der Begriff der „History" bezüglich der Navigation erklärt. Eine derartige History beinhaltet Daten in Form von Wegpunkten, welche selber mehrere Informationen enthalten. Wegpunkte von Tracks (engl. Trackpoints) enthalten Zeit- und Datumsangaben, Positionskoordinaten und die Richtung. In einigen Fällen auch Höhenangaben[23] und auch einfache Notizen[24]. Bei Tracks werden die Notizen und sonstige zusätzliche Informationen zur optimalen Speicherbenutzung des Ortungssystems (z.B. GPS-Gerät) oft weggelassen. Ein Track besteht daher im Verglich zur Route mehrheitlich aus anonymen Wegpunkten[25]. Die in den History-Daten enthaltenen Koordinaten legen die Position fest, die Zeit- und Datumsangaben den genauen Zeitpunkt der Erfassung bzw. den Zeitpunkt bei welchem das Objekt an der entsprechenden Position war. Die Höhenangaben dienen bei Flugobjekten dazu, einen Wegpunkt im dreidimensionalen Raum eindeutig festzulegen. Sehr oft werden neben diesen Informationen auch noch zusätzliche Informationen gespeichert, wie die aktuelle Bewegungsrichtung und Geschwindigkeit[26]. Je nach Notwendigkeit können auch noch weitere Informationen wie der Name eines Wegpunktes oder andere Merkmale festgehalten werden.

[22] http://www.fermoll.de/ozi_ce/ce_handbuch/konfiguration/trackkonf.htm
[23] http://home.wtal.de/noegs/gps-lexikon.htm#trackpoint
[24] http://www.gps24.de/softtraxx_handbuch.doc
[25] http://www.gps24.de/softtraxx_handbuch.doc
[26] http://service.alan-germany.de/GPS/Map500/FAQ.htm#_Toc75850528

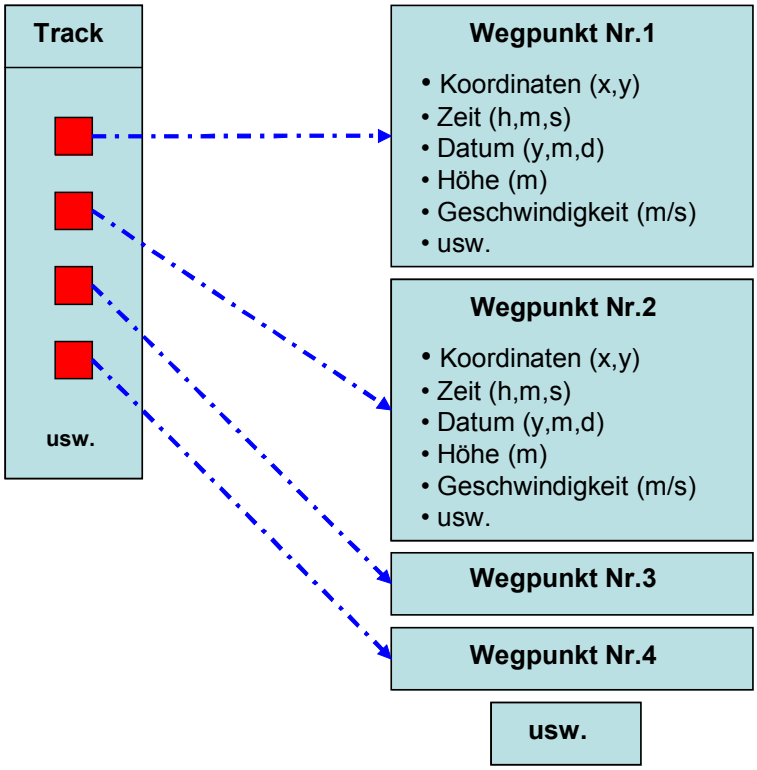

Abbildung 3. 4: Die History-Daten bezüglich der Navigation

3.3 Die Darstellung von History-Daten

Ein weiterer wichtiger Bestandteil von History-Funktionalitäten ist neben der Erstellung von History-Daten und deren Speicherung in Tracks, die angemessene graphische Darstellung einer entsprechenden History bezüglich Navigation. Wie in Abschnitt 3.2 gezeigt wurde, sind in den einzelnen History-Daten bzw. den einzelnen Wegpunkten eines Tracks, verschiedene Informationen enthalten. Der nächste Schritt besteht nun darin, diese Informationen optimal zu verarbeiten, und daraus eine für den Benutzer bzw. für den Anwender informative Darstellung zu erreichen. Die Darstellung soll dem Benutzer die Daten in einer anderen und vor allem einer zugänglicheren Form aufzeigen. Die Interpretation und das Verständnis von Daten werden durch eine situationsgerechte und übersichtliche Darstellung stark vereinfacht. Genau so wie die Daten je nach Anwendung unterschiedlich gesammelt und gespeichert werden, so müssen sie auch abhängig vom erwarteten Ziel der Auswertung unterschiedlich dargestellt werden. Möchte der Benutzer zum Beispiel nur die zurückgelegte Bodendistanz festhalten, so wird er kaum eine Darstellung bevorzugen, welche die Höhen und die Beschleunigungen berücksichtigt. Visualisierungen können also unterschiedliche Ziele haben. Folgende Ziele lassen sich unterscheiden:

- **Explorative Analyse**
  Bei der explorativen Analyse wird versucht, die Struktur, die Zusammenhänge und eventuelle Trends von Daten zu erkennen. Zudem soll eine Identifikation und Klassifikation der vorhandenen Daten erfolgen.

- **Bestätigende Analyse**
  Hier wird versucht eine Hypothese zielgerichtet zu überprüfen. Mittels einer Verdachtsdiagnose werden Werte verglichen und die Hypothese bestätigt oder abgelehnt. Es kann auch eine Konkretisierung erfolgen.

- **Ergebnisdarstellung**
  Die ausgewählten Ergebnisse einer Auswertung von Daten werden präsentiert. Die Ergebnisfindung und die Ausgangslage werden explizit hervorgehoben.

- **Präsentation als Kommunikationsgrundlage und Entscheidungsunterstützung**
  Die zuvor überprüften Daten werden durch eine konkrete Auswahl von Visualisierungsparametern dargestellt. Die Visualisierungsparameter sind so gewählt, dass sie die Visualisierungsziele effektiv erreichen. Strukturen und Trends werden hervorgehoben.

Eine Visualisierung, die alle gesammelten Daten situationsgerecht und daher effizient gefiltert darstellt, überträgt die Informationen zielgerichteter an den Betrachter. Visualisierungen bzw. Bilder wirken generell schneller, unmittelbarer sowie mit höherem Wiedererkennungs- und Erinnerungswert auf die menschliche Reizverarbeitung[27].

Auch die Darstellungsart der History-Daten hängt stark von dem erwarteten Informationsgehalt ab. Darstellungsarten die alle verfügbaren Informationen mit einbeziehen können sehr schell unübersichtlich werden. Begrenzte Bildschirmgrößen und deren mögliche Auflösung limitieren auch die Darstellung von großen und komplexen Datensätzen bzw. Tracks. Entweder verzichtet man bei der Darstellung des gesamten Datensatzes auf wesentlich feinere Detailinformation oder durch Darstellung der Detailinformation auf die Darstellung des Gesamtkontextes. Bei großen Tracks besteht zudem die Gefahr, dass die Informationszusammengehörigkeit sehr schnell in der Unübersichtlichkeit verloren geht[28].

Bei der Darstellung von History-Daten bezüglich Navigation kann es sich um Pfeile, Linien, Kreise, generelle Grafiken und geometrische Figuren oder um Abbildungen jeglicher Art handeln. Aber ebenso kann der Einsatz verschiedener Farben zu einem höheren und leichter verständlicheren Informationsgehalt führen. Farbe ist Information. Sie hat eine unterstützende Funktion, um in einem Kommunikationsprozess eine Botschaft zum Empfänger bzw. Betrachter zu transportieren. Dies umso mehr, je gelungener die Farbgestaltung ist. Desto wirkungsvoller der Farbeinsatz ist, desto leichter und bereitwilliger werden die Informationen aufgenommen[29]. Die Interaktion von Farben untereinander ist sehr interessant. Gewisse Farbenkombinationen die nebeneinander platziert werden, generieren auffallende Effekte, wie Verschwommenheit an den Kanten wo sie sich treffen. Farben können aber auch sehr täuschend sein. Zwei verschiedene Farben können auf unterschiedlichen Hintergründen sehr ähnlich oder sogar gleich aussehen. Auf der anderen Seite kann eine Farbe auch unterschiedlich aussehen, abhängig von der Hintergrundfarbe, den adjazenten Farben oder der Größe der Farbfläche [EARNSHAW]. Die Menge von Farben kann

[27] http://www.doku.net/artikel/visualisie.htm
[28] http://gio.uni-muenster.de/beitraege/ausg97_1/pb/bbbpaper.htm#5_gene
[29] http://www.ipsi.fraunhofer.de/~crueger/farbe/farb-ein.html

je nach Einsatz stark variieren. Dreidimensionale Anwendungen brauchen zum Beispiel entscheidend mehr Farben für realistische Darstellungen als zweidimensionale Graphen oder Tabellen.

Die Visualisierung von großen Datenmengen stellt eine große Herausforderung dar. Dabei ist die Beibehaltung der Übersichtlichkeit eine zentrale Problematik von umfangreichen Visualisierungen. Folgend werden zwei Verfahren gezeigt, die diese Problematik zu lösen versuchen.

3.3.1 „Focusing and Linking"

Andreas Buja [BUJA] beschreibt in seiner Publikation „Interactive Data Visualization using Focusing and Linking" die Darstellungsphilosophie des ,,Focussing und Linking". Diese Darstellungsphilosophie verzichtet auf eine direkte Einbettung der Detailinformation in den Gesamtkontext. Die Informationen werden nicht direkt in einem Bild dargestellt, sondern sie werden sinngemäß auf mehrere Fenster (in diesem Zusammenhang auch engl. „Views" genannt) auf dem Rechner verteilt. Koordinaten einer Position können zum Beispiel in einem anderen Fenster sein und nicht im Fenster welches die eigentlichen Wegpunkte darstellt. So sind die Informationen nun übersichtlicher und leichter zugänglich. Jedes zusätzliche Fenster setzt seinen Fokus klar auf einen eindeutigen Aspekt der verfügbaren Daten („Focusing"). Eine Konsequenz dieser Fokussierung ist das jedes Fenster nur einen bestimmten Teil der gesamthaft vorhandenen Informationen und Daten beinhaltet. Die verschiedenen Fenster müssen also verbunden werden („Linking"), damit die in den einzelnen Fenstern enthaltenen Informationen in einem einheitlichen Kontext integriert werden können. Die Art und Weise wie die einzelnen Fenster verbunden werden hängt davon ab ob sie nacheinander zeitabhängig oder gleichzeitig angezeigt werden.

Gemäß Andreas Buja [BUJA] gibt es verschiedene Methoden des „verbinden" bzw. des „verknüpfen" von Fenstern. Ein generelles Verfahren zur Verbindung von nacheinander zeitabhängigen Fenstern ist die leichte Veränderung der Position eines Objekts im entsprechenden Fenster. Auch sind leichte Farb- und Größenveränderungen möglich, wie auch feine Animationen. Dazu ein Beispiel: In der unteren Abbildung sind zwei Fenster zu sehen. In diesen zwei Fenstern sind zwei Streudiagramme bezüglich Ortschaften abgebildet. Die Punkte repräsentieren Ortschaften. Das linke Streudiagramm stellt Ortschaften bezüglich dem Klima und den Wohnkosten dar. Das rechte Diagramm enthält eine Karte mit 300 Ortschaften. Möchte der Benutzer nun wissen wo er mit möglichst niedrigen Kosten in einem möglichst milden Klima eine Wohnung finden kann, so selektiert er im Fenster links die entsprechenden Punkte. Nachdem mit der Maus eine Selektion vorgenommen wurde, werden die im verbundenen rechten Fenster zugehörigen Messpunkte sichtbar. Dies durch Veränderungen ihrer Repräsentation im Diagramm.

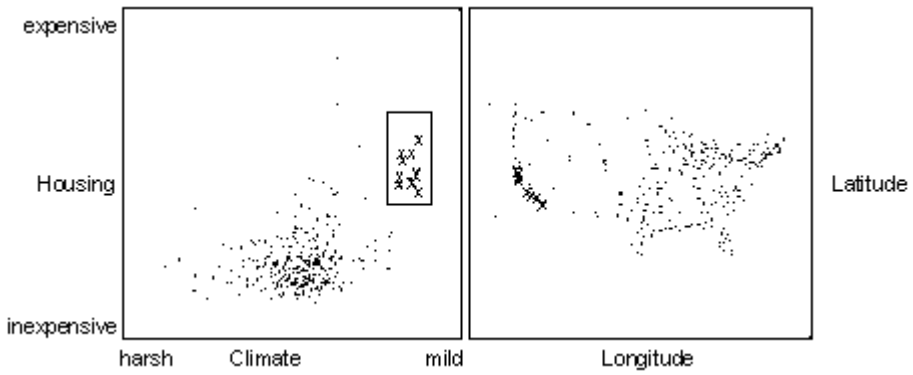

Abbildung 3. 5: Streudiagramm von Ortschaften [BUJA]

Das Prinzip des „Focusing and Linking" bietet eine Lösung des Problems bezüglich überladenen Visualisierungen an. Es fügt nicht alle vorhandenen Informationen in eine Darstellung ein, sondern teilt die Darstellung in mehrere Fenster auf, welche sich auf bestimmte Aspekte der Daten fokussieren. Werden diese sinngemäß durch situationsgerechte Verfahren verbunden, so entsteht eine Visualisierung die alle Daten und Informationen integriert ohne dass die Übersicht verloren geht.

3.3.2 DOI – „Degree of Interest"

Das Prinzip des „Focusing and Linking" versucht große Datenmengen übersichtlich in mehrere verbundene Fenster zu visualisieren. Trotz diesen Verfahren besteht immer noch die Möglichkeit, dass die generierten Fenster sehr viele Detailinformationen enthalten was zu einer verminderten Übersichtlichkeit führt. Ein weiterführender Ansatz zur Reduktion der Mengen von Informationen ist die Bestimmung der Wichtigkeit einer Detailinformation. Ist bekannt welche Informationen für den Benutzer wichtig sind, so kann sich eine Visualisierung ausschließlich auf diese Informationen fokussieren. Durch die Bestimmung des DOI einer Information bzw. eines Objekts, kann eine Filterung der Datenmengen vorgenommen werden, was wiederum zu einer Verminderung der dargestellten Informationen führt. Ziel ist es also, eine Visualisierung zu generieren, die selbständig die für den aktuellen Benutzer wichtigen Informationen darstellt. Gemäß d'Entremont und Storey [ENTREMONT] enthalten Interfaces, welche auf die Aufmerksamkeit der Benutzers reagieren, zwei Komponenten:

- Einen Mechanismus der kontinuierlich den Interessensgrad (Degree of Interest) berechnet.

- Eine dynamische Anzeige der selektierten Informationen bezüglich des DOI.

Doch welche Objekte bzw. Informationen sind wichtig? Dazu können zum Beispiel die verschiedenen Interessensgrade berücksichtigt werden, also der „Degree of Interest" (DOI) eines Objekts bzw. einer Information. Dabei spielen laut Fairchild [FAIRCHILD] zwei Werte eine wesentliche Rolle:

- Die „Semantische Distanz" (Die Entfernung zwischen Objekt und Beobachtungspunkt).

- Die „A Priori Importance" (Die Wichtigkeit eines Objekts für den Beobachter).

George W. Furnas [FURNAS] zeigt in seiner Publikation „Generalized Fisheye Views" wie mit diesen zwei Werten der Interessensgrad (DOI) berechnet werden kann.

SD = Semantische Distanz, API = A Priori Distanz

$$DOI\ (x|.=y) = API(x) - SD(x,y)$$

Mit dieser Formel wird das Interesse des Benutzers an dem Punkt X vom (Stand)Punkt y aus beschrieben. Abhängig ist die Wichtigkeit des Objektes von der vorher bestimmten API abzüglich der semantischen Distanz SD. Als SD wird in der Regel die euklidische Distanz angenommen.

Neben dieser Formalisierung des DOI gibt es auch andere zugänglichere Kriterien zur Bestimmung der Wichtigkeit einer Information bzw. eines Objektes:

- Mausbewegungen, Häufigkeit des Anklickens und des Betretens eines Bereiches oder einer Zone des Bildschirms.

- Augenbewegungen. Wo schaut der Anwender am häufigsten hin? Was schaut er sich am längsten an?

- Allgemeine Gestiken wie Handbewegungen und Mimikveränderungen.

3.3.3 Fazit

Darstellungen von History-Daten bezüglich Navigation können sehr verschieden sein. Es gibt wohl unzählige Möglichkeiten eine History bzw. einen Track darzustellen. Das Konzept des „Focusing and Linking" stellt eine Möglichkeit dar, komplexe, umfangreiche und schwer zugängliche Informationen in einer für den Anwender übersichtlichen Form aufzuzeigen. Die Verfahren zur Bestimmung des Interessengrades (DOI) oder generell der Wichtigkeit von Informationen verstärkt dabei die Idee der Fokussierung. Eine Visualisierung, welche diese beiden Konzepte integriert und zusammen kombiniert, oder sogar erweitert, könnte eine Visualisierung der Zukunft sein. Nicht zu letzt hängt die Effizienz der Informationsübertragung von der Visualisierung zum Anwender schlussendlich auch vom Anwender selber ab. Berücksichtigt man dabei die Theorie des „Information Foraging" [PIROLLI] welche besagt, dass der Mensch seine Umgebung und seine Strategien dahingehend strukturiert, so viele Informationen wie möglich zu sammeln, so besteht eine große Chance einen möglichst optimalen Informationsfluss zwischen der Visualisierung und dem Anwender zu erzeugen.

3.4 Übersicht einer History-Funktionalität

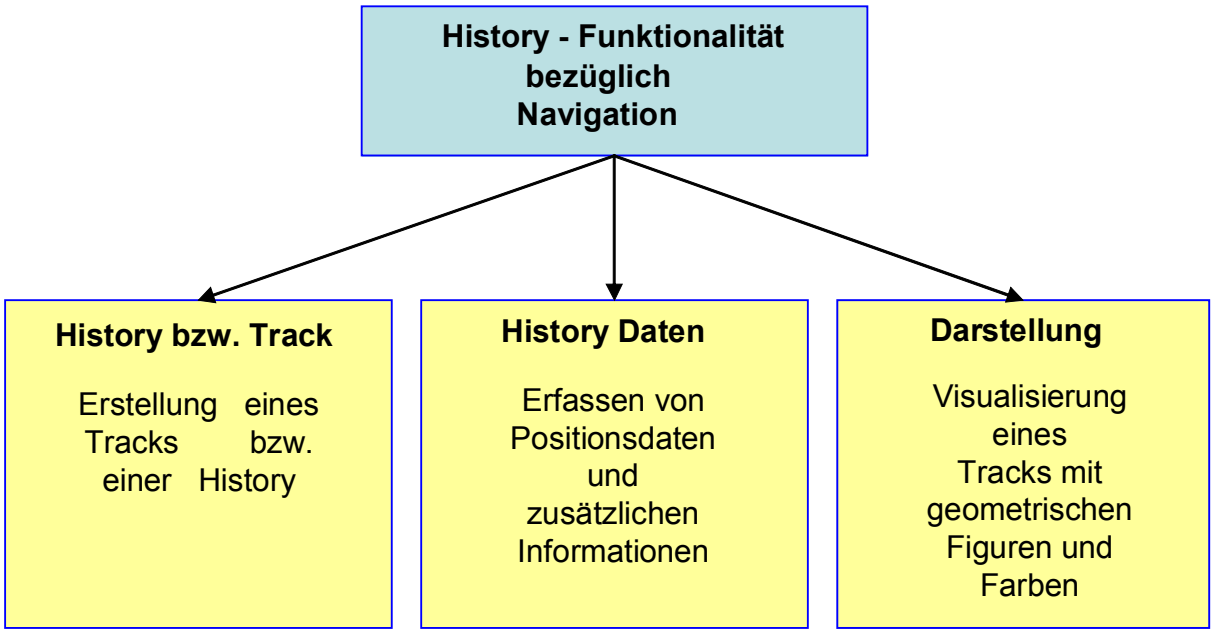

Abbildung 3. 6: Übersicht der History-Funktionalität

3.5 Beispiele aus der Praxis

History-Funktionalitäten bezüglich Navigation werden heutzutage in der Praxis sehr oft verwendet. Generell nimmt die Benutzung von qualitativ hervorragenden Navigationsgeräten stetig zu. Das GPS hat sich in den letzten Jahren auch in zivilen Bereichen etabliert. Wurde früher diese Satellitennavigation ausschließlich für militärische Zwecke benutzt, so wird sie heutzutage häufig auch für verschiedene Freizeitbeschäftigungen erfolgreich eingesetzt. Dabei halten sich die Kosten für die Beschaffung und Wartung der Navigationsgeräte in Grenzen. Auch die Bedienung ist sehr einfach gehalten und wird immer einfacher. Kein Wunder denken nicht wenige Menschen an eine Beschaffung dieser Navigationsgeräte nach.
Folgend wird ein Beispiel aus der Praxis gezeigt, welches die Benutzung von History-Funktionalitäten aufzeigen sollen. Dabei wird speziell auf die verschiedenen Darstellungsarten von Wegaufzeichnungen bzw. Tracks eingegangen. Die Darstellung des Weges erfolgt in diesem Beispiel nicht parallel bzw. gleichzeitig neben der Erfassung der Wegdaten, wie das beim mobileGame der Fall ist. Trotzdem ist sehr gut zu sehen, was für unterschiedliche Möglichkeiten zur Visualisierung von History-Daten bezüglich Navigation vorhanden sind.

3.5.1 Beispiel - MaxPunkte[30]

Der Entwickler dieser Software ist Dr. Ing. Dietrich Münchmeyer. MaxPunkte ist eine Software zur Optimierung von Flugdaten. Konkret können in diesem

[30] http://www.flugplatz-beilrode.de/maxpunkte/

Programm Wegaufzeichnungen bzw. Tracks, die mit einem GPS-Gerät erfasst wurden, importiert und durch verschiedene Darstellungsarten analysiert und allgemein eingesehen werden. Es ist auch möglich entsprechende digitale Karten zu importieren, die mit den Wegaufzeichnungen eine optimale Darstellung ergeben. Weiter besteht die Möglichkeit, die Wegaufzeichnungen nach Google Earth[31] zu exportieren. Wodurch der Import von digitalen Karten nicht mehr notwendig ist. Die Software bietet mehrere Darstellungsarten an die unter anderem in der Semantik der Farben, generell der Auswertung der Daten und im Umfang der Darstellungen variieren.

Wie erwähnt müssen Wegaufzeichnungen importiert werden. Diese Wegaufzeichnungen beinhalten verschiedene Informationen. Welche Informationen pro Wegpunkt gespeichert werden hängt primär von der Art des Gebrauchs und vom erwarteten Nutzen der Auswertung ab. Diese Software wird ausschließlich für Drachenflieger benutzt, und dem entsprechend werden neben den Positionskoordinaten auch Daten bezüglich Höhe, Geschwindigkeit und Kursrichtung erfasst. Folgende Abbildung zeigt einen Ausschnitt eines Tracks eines Drachenfliegers:

Header	Name	Start Time	Elapsed Time	Length	Average Speed			
Track	Tour 04-03-30	30.03.2004 15:09	04:42:28	56.6 km	12 kph			
Header	**Position**	**Time**	**Altitude**	**Depth**	**Leg Length**	**Leg Time**	**Leg Speed**	**Leg Course**
Trackpoint	N49.76682 E8.68628	30.03.2004 15:09	326 m					
Trackpoint	N49.76695 E8.68654	30.03.2004 15:09	327 m		23.4 m	00:00:16	5.3 kph	52° true
Trackpoint	N49.76710 E8.68658	30.03.2004 15:09	326 m		17.0 m	00:00:03	20 kph	10° true
Trackpoint	N49.76774 E8.68686	30.03.2004 15:09	322 m		74.4 m	00:00:12	22 kph	16° true
Trackpoint	N49.76785 E8.68690	30.03.2004 15:09	318 m		12.3 m	00:00:02	22 kph	14° true
Trackpoint	N49.76789 E8.68692	30.03.2004 15:09	317 m		5.02 m	00:00:01	18 kph	18° true
Trackpoint	N49.76807 E8.68716	30.03.2004 15:09	321 m		25.6 m	00:00:05	18 kph	42° true
Trackpoint	N49.76809 E8.68720	30.03.2004 15:09	321 m		3.90 m	00:00:01	14 kph	52° true
Trackpoint	N49.76809 E8.68731	30.03.2004 15:10	320 m		7.71 m	00:00:02	14 kph	90° true
Trackpoint	N49.76804 E8.68767	30.03.2004 15:10	324 m		26.7 m	00:00:08	12 kph	100° true
Trackpoint	N49.76800 E8.68791	30.03.2004 15:10	326 m		17.6 m	00:00:06	11 kph	106° true
Trackpoint	N49.76794 E8.68849	30.03.2004 15:10	328 m		42.3 m	00:00:16	9.5 kph	100° true
Trackpoint	N49.76796 E8.68868	30.03.2004 15:10	331 m		14.1 m	00:00:07	7.2 kph	80° true
Trackpoint	N49.76798 E8.68873	30.03.2004 15:10	332 m		3.90 m	00:00:02	7.0 kph	52° true
Trackpoint	N49.76817 E8.68898	30.03.2004 15:10	335 m		28.4 m	00:00:11	9.3 kph	41° true
Trackpoint	N49.76852 E8.68920	30.03.2004 15:11	336 m		41.2 m	00:00:14	11 kph	22° true
Trackpoint	N49.76875 E8.68920	30.03.2004 15:11	338 m		26.3 m	00:00:08	12 kph	0° true
Trackpoint	N49.76897 E8.68896	30.03.2004 15:11	339 m		29.3 m	00:00:11	9.6 kph	325° true
Trackpoint	N49.76918 E8.68858	30.03.2004 15:11	342 m		36.6 m	00:00:14	9.4 kph	311° true
Trackpoint	N49.76931 E8.68866	30.03.2004 15:11	341 m		15.6 m	00:00:07	8.0 kph	23° true
Trackpoint	N49.76935 E8.68888	30.03.2004 15:11	343 m		16.2 m	00:00:09	6.5 kph	73° true
Trackpoint	N49.76942 E8.68909	30.03.2004 15:12	347 m		17.0 m	00:00:10	6.1 kph	65° true
Trackpoint	N49.76948 E8.68941	30.03.2004 15:12	349 m		24.2 m	00:00:13	6.7 kph	73° true

Abbildung 3. 7: History-Daten einer Wegaufzeichnung

Tracks können als einfache Skizze, als Barogramm (Fokus auf die Höhe), Polardigramm (Fokus auf Geschwindigkeit und Richtung zum nächsten Wegpunkt) und als dreidimensionaler Plot angezeigt werden. Falls entsprechende Dateien vorhanden sind, können zudem das Relief bzw. das digitale Höhenmodell, Landkarten und Lufträume dargestellt werden.

[31] http://earth.google.com/

Die Farbcodierung können auch verändert werden. Je nach Einstellung stellt die Farbe die Höhe, das Steigen oder Sinken oder die Bewegungsgeschwindigkeit dar. Es können auch verschiedene Farbmodelle für die Reliefdarstellung gewählt werden.

Mit dieser Software lassen sich auch Distanzen zwischen zwei Wegpunkten sowie auch die Gesamtdistanz berechnen. Zoom Funktionen sind ebenfalls enthalten.

Folgende Abbildungen zeigen die verschiedenen Darstellungsarten von Wegaufzeichnungen, welche die Software „MaxPunkte" zur Verfügung stellt.

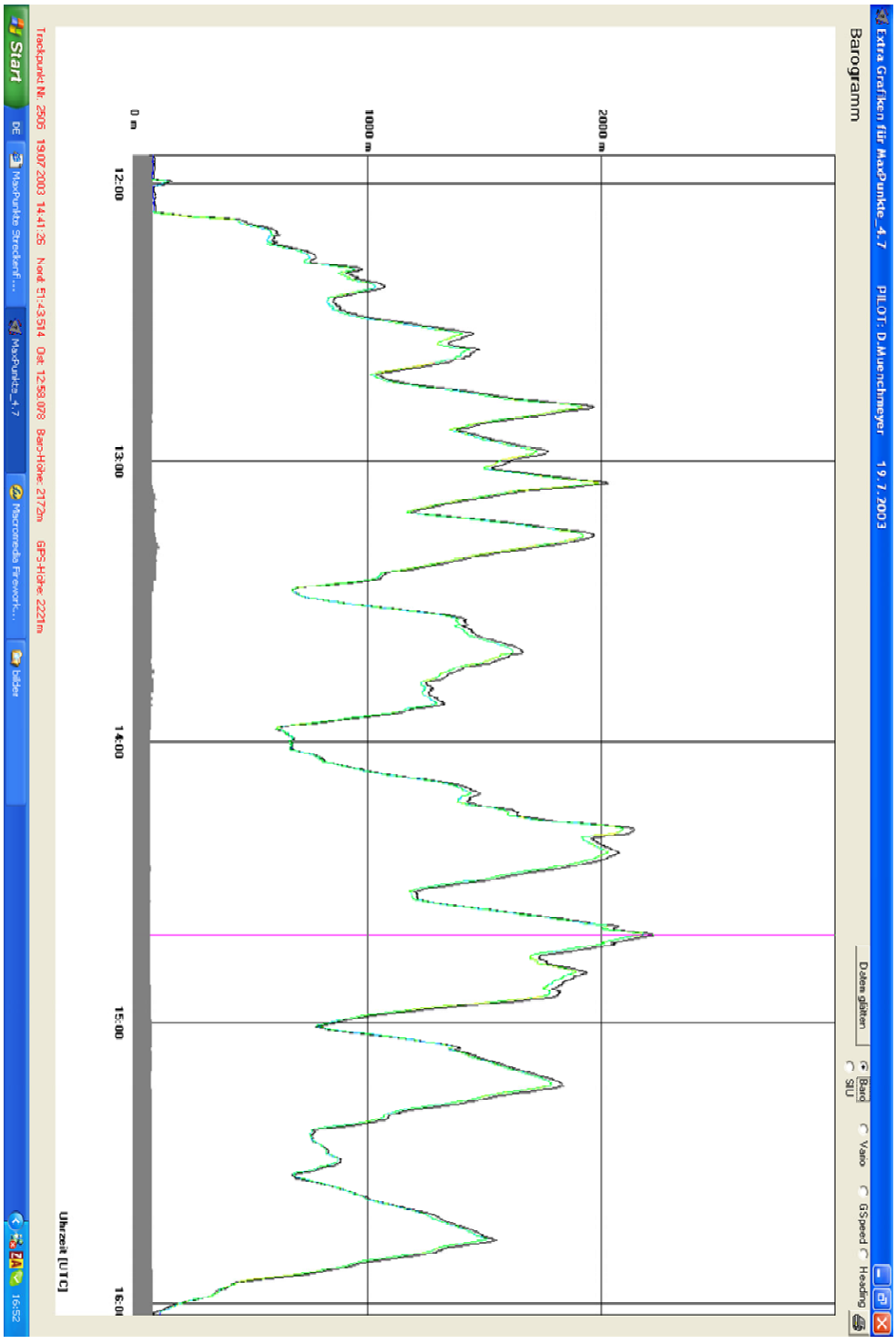

Abbildung 3. 8: Das Barodiagramm bzw. der Höhengraph

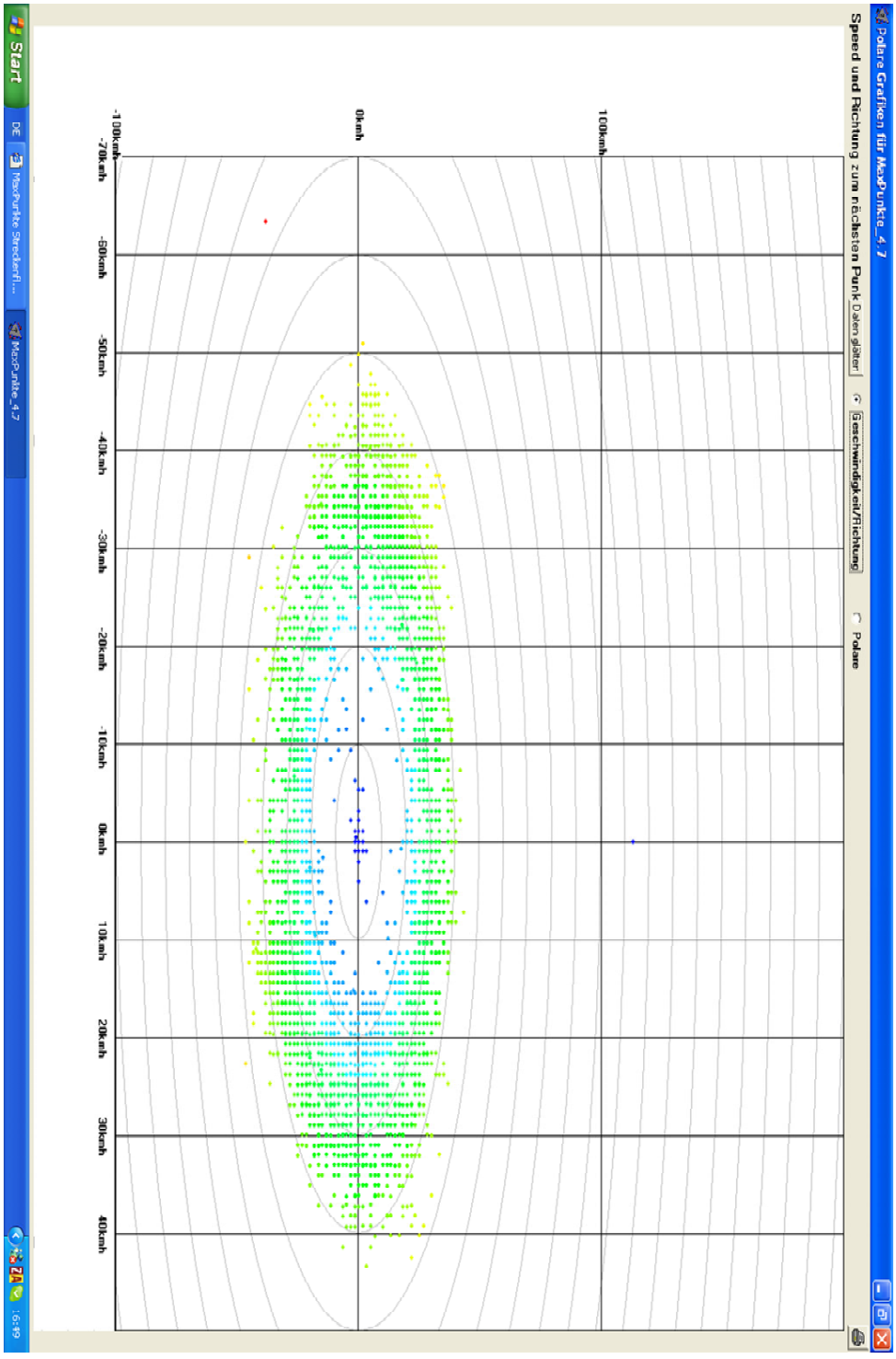

Abbildung 3. 9: Das Polardiagramm

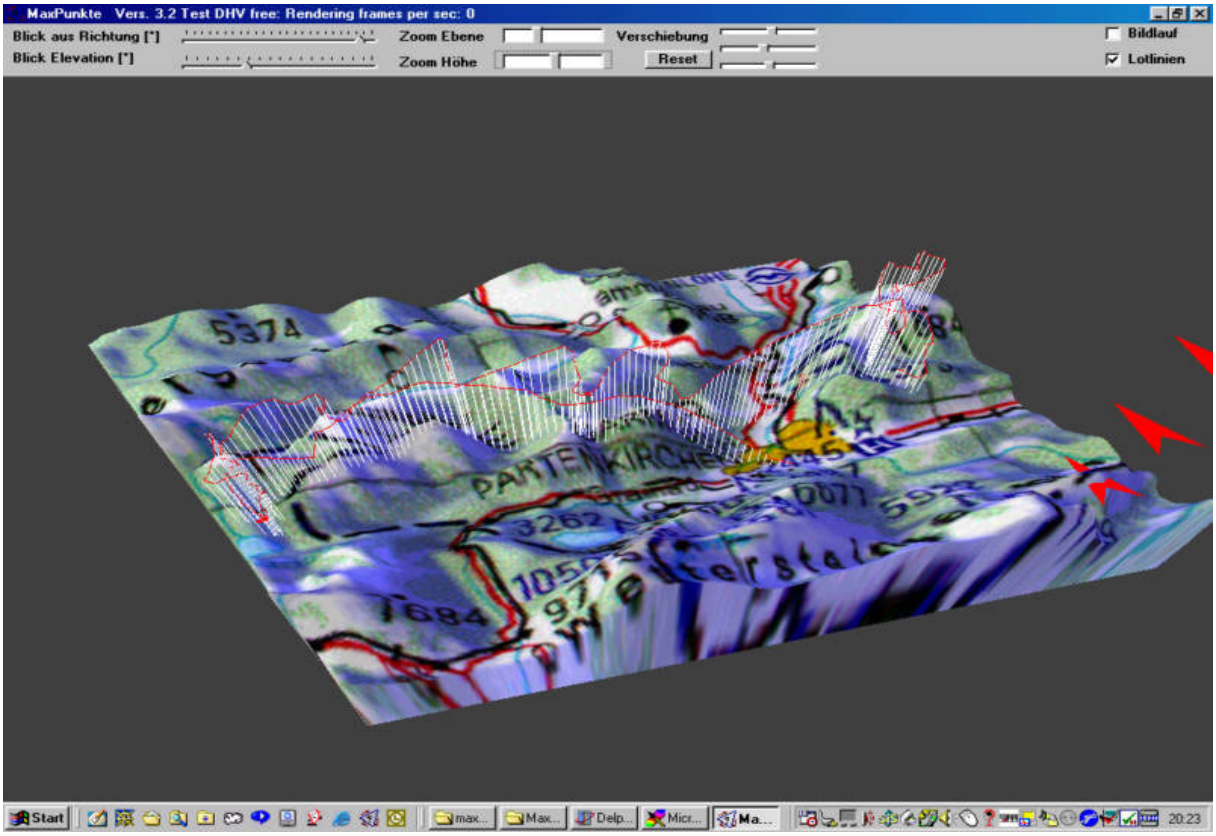

Abbildung 3. 10: 3-Dimensionale Darstellung mit Relief[32]

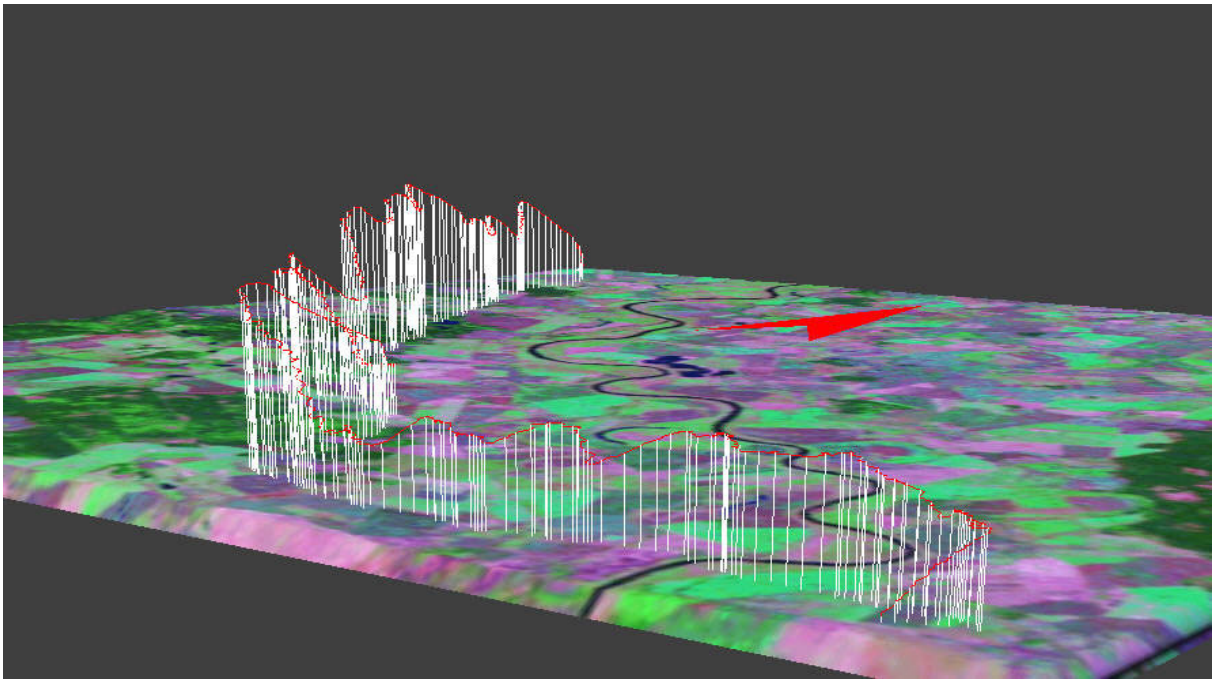

Abbildung 3. 11: 3-Dimensionale Darstellung ohne Relief[33]

[32] http://www.flugplatz-beilrode.de/maxpunkte/
[33] http://www.flugplatz-beilrode.de/maxpunkte/

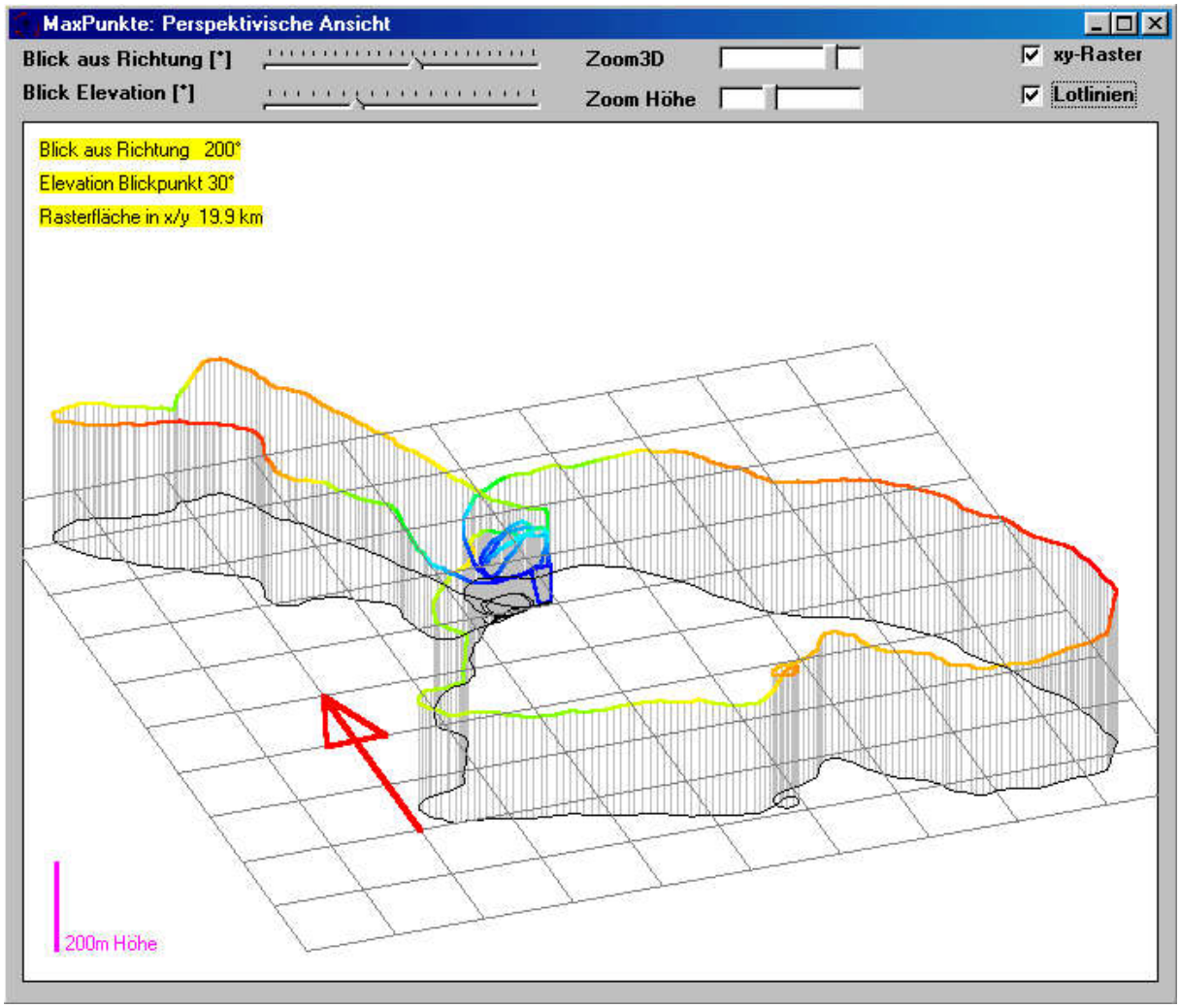

Abbildung 3. 12: 3-Dimensionale Darstellung ohne Relief und ohne Landkarte[34]

3.5.2 Weitere Beispiele

Neben der im letzten Abschnitt gezeigten Software gibt es natürlich noch andere Programme die ähnliche Funktionalitäten bieten. Nicht alle sind so umfangreich wie die eben vorgestellte Software. Sie unterscheiden sich nicht grundlegend voneinander. Die Unterschiede sind meist im Detail zu finden. Für den Benutzer steht somit eine große Auswahl an Programmen zur verfügen, mit Hilfe dieser er sich seine Wegaufzeichnung analysieren und darstellen lassen kann. Die folgenden Abbildungen zeigen weitere Darstellungsarten, wobei immer auch die dazu gehörende Software genannt wird.

[34] http://www.flugplatz-beilrode.de/maxpunkte/

3.5.2.1 Trackdarstellung mit dem GPS-Visualizer[35]

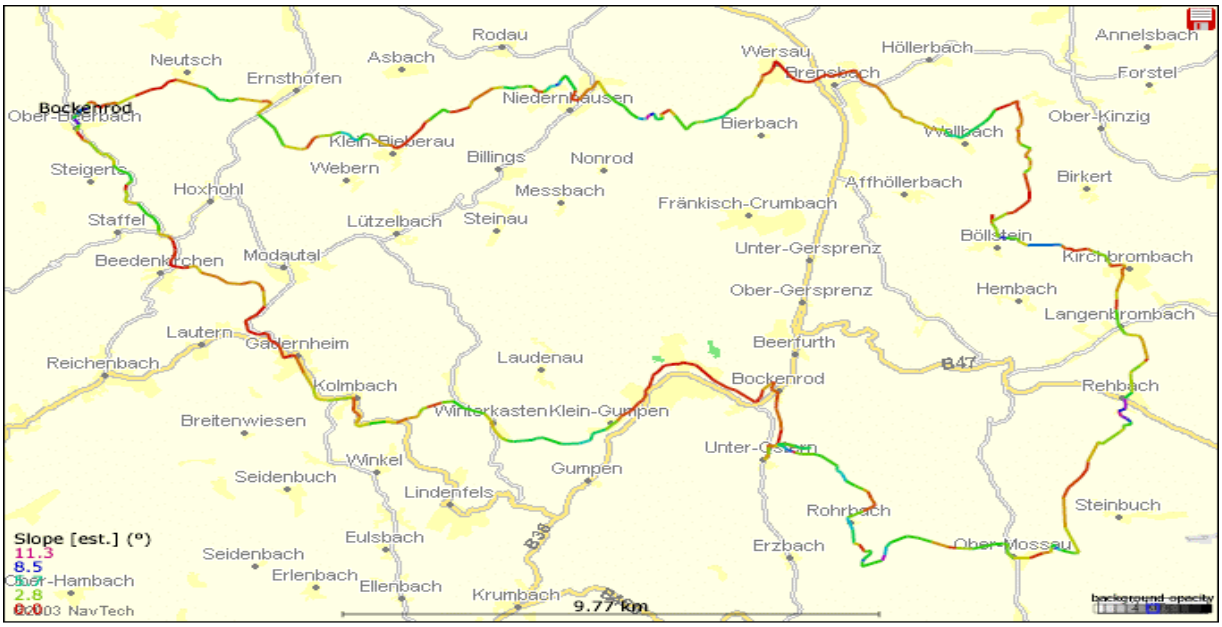

Abbildung 3. 13: Wegdarstellung mit dem GPS-Visulizer

3.5.2.2 Trackdarstellung mit Garmin MapSource[36]

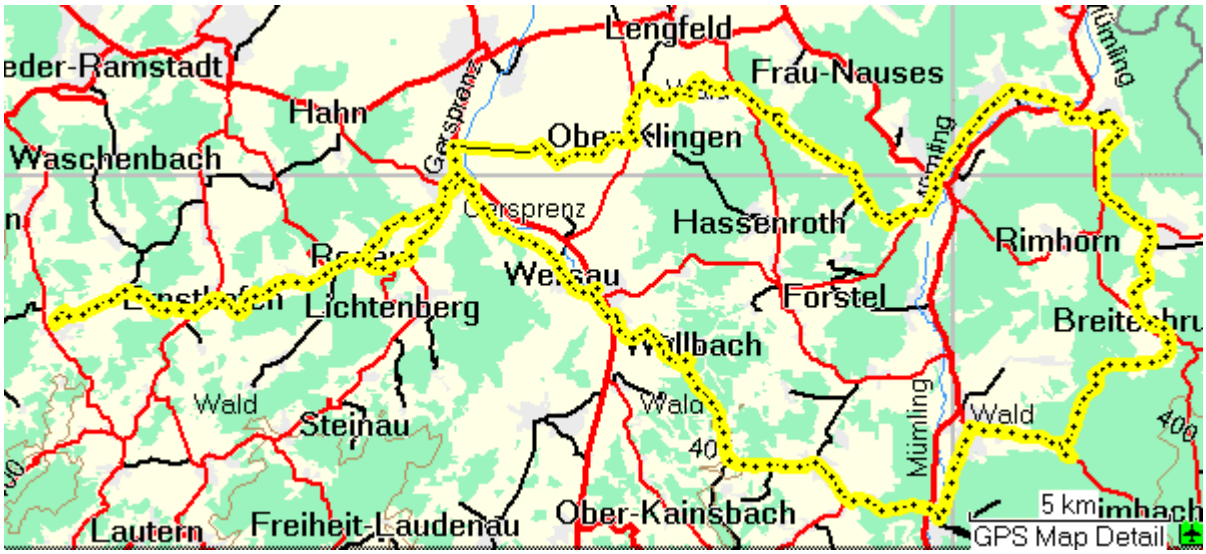

Abbildung 3. 14: Wegdarstellung mit MapSource

[35] http://www.gpsvisualizer.com/
[36] http://www.garmin.com/cartography/mapsourceTutorial.html

3.5.2.3 Trackdarstellung mit dem GPS TrackMaker[37]

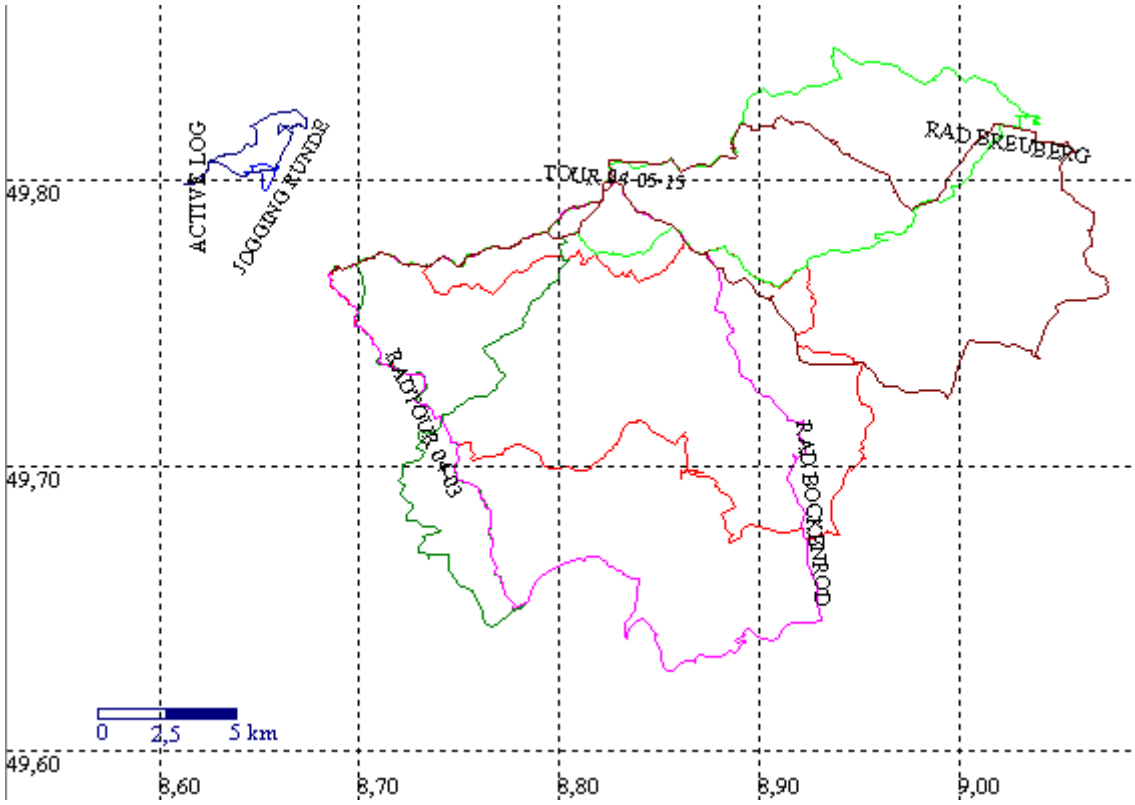

Abbildung 3. 15: Wegdarstellung mit dem GPS TrackMaker

[37] http://www.gpstm.com/

4 Entwurf und Entwicklung der History-Funktionalität im mobileGame

Nachdem die verschiedenen Bestandteile einer History-Funktionalität bezüglich Navigation aufgezeigt wurden, wird in diesem Abschnitt die Entwicklung dieser neuen Funktionalität im mobileGame dokumentiert. Zuerst werden die Anforderungen an die neue Funktionalität bzw. der neuen mobileGame Erweiterung spezifiziert, anschließend werden diese Schritt für Schritt im System implementiert. Schlussendlich werden die neuen Funktionalitäten an verschiedenen Orten mit verschiedenen Personen und in diversen Versuchen geprüft und getestet.

4.1 Anforderungsspezifikation

Die Anforderungsspezifikation legt detailliert fest, was von dem zu entwickelnden System verlangt wird. Fehler in der Anforderungsanalyse werden oft sehr spät erkannt. Deswegen ist es wichtig genau zu eruieren, wie die Anforderungen sind bzw. was das neu zu entwickelnde System für Fähigkeiten und Eigenschaften haben muss. Auch wenn eine Evolution der Anforderungen stattfinden kann, so ist eine gründliche Anforderungsanalyse während der Anfangsphase als wichtig zu betrachten. Eine gute Spezifikation der Anforderung hat folgende Merkmale [GLINZ_SE]:

- **Vollständigkeit**
  Die Spezifikation beschreibt alles was die Software enthalten soll bzw. was der Auftraggeber, in diesem Falle mein Betreuer Christoph Göth, vom Endsystem erwartet.

- **Widerspruchsfreiheit**
  Die Spezifikation ist nicht realisierbar wenn Widersprüche darin enthalten sind. Deswegen muss sie widerspruchsfrei sein.

- **Eindeutigkeit**
  Die Spezifikation soll klar und eindeutig sein, damit Fehlinterpretationen vermieden werden können.

- **Prüfbarkeit**
  Die Spezifikation soll prüfbar sein, damit am Schluss der Entwicklung feststellbar ist, ob das System die geforderten Anforderungen erfüllt.

- **Verständlichkeit**
  Die Anforderungen müssen für alle Beteiligten verständlich und gut nachvollziehbar sein.

- **Adäquatheit**
  Die Spezifikation soll nur genau das Beschreiben, was der Kunde will. Eine Überladung der Spezifikation kann zu Missverständnissen führen.

Der Spezifikationsprozess beinhaltet die Gewinnung, die Darstellung und die Prüfung der Anforderungen. Die Gewinnung der Anforderungen bezüglich der History-Funktionalität erfolgte nicht an einem bestimmten Zeitpunkt, sondern

iterativ in mehreren zeitlich verstreuten Interaktionen zwischen mir und meinem Betreuer, der bezüglich des Spezifikationsprozesses auch mein Kunde widerspiegelte. Um die Anforderungen zu erfassen wurden am Anfang und auch während der Diplomarbeit Interviews bzw. Experteninterviews geführt.

4.1.1 Interviews

Um die verschiedenen Anforderungen an die neue Funktionalität erfassen zu können, wurden mehrere kleinere Interviews ausschließlich mit meinen Betreuer Christoph Göth durchgeführt. Diese Interviews sollten aufzeigen, was für konkrete Vorstellungen Christoph Göth bezüglich einer History-Funktionalität beim mobileGame hat. Die verschiedenen Interviews stellten einen längeren Prozess dar, der sich aus mehreren abhängigen Teilprozessen zusammensetzte. Zuerst habe ich die grundlegenden Ideen und Vorstellungen von Christoph Göth kennen gelernt. Durch Fragen meinerseits und den Antworten die ich bekommen habe konnten die Ideen und Vorstellungen verfeinert und konkretisiert werden. Diese verfeinerten Ideen wurden von mir bezüglich verschiedener Faktoren analysiert und die gefilterten Erkenntnisse mit zusätzlichen Fragen meinem Betreuer mitgeteilt. Durch diese Erkenntnisse wurden verschiedene Anwendungsszenarien von mir gebildet und durchgespielt, welche vor allem die wichtigsten Interaktionen des zu spezifizierenden Systems mit der Umgebung umfassten. Diese Anwendungsszenarien eigneten sich gut zur interaktiven Gewinnung der Anforderungen. Die ermittelten Anforderungen wurden dann nochmals mit meinem Betreuer besprochen und schlussendlich auch von ihm bestätigt.

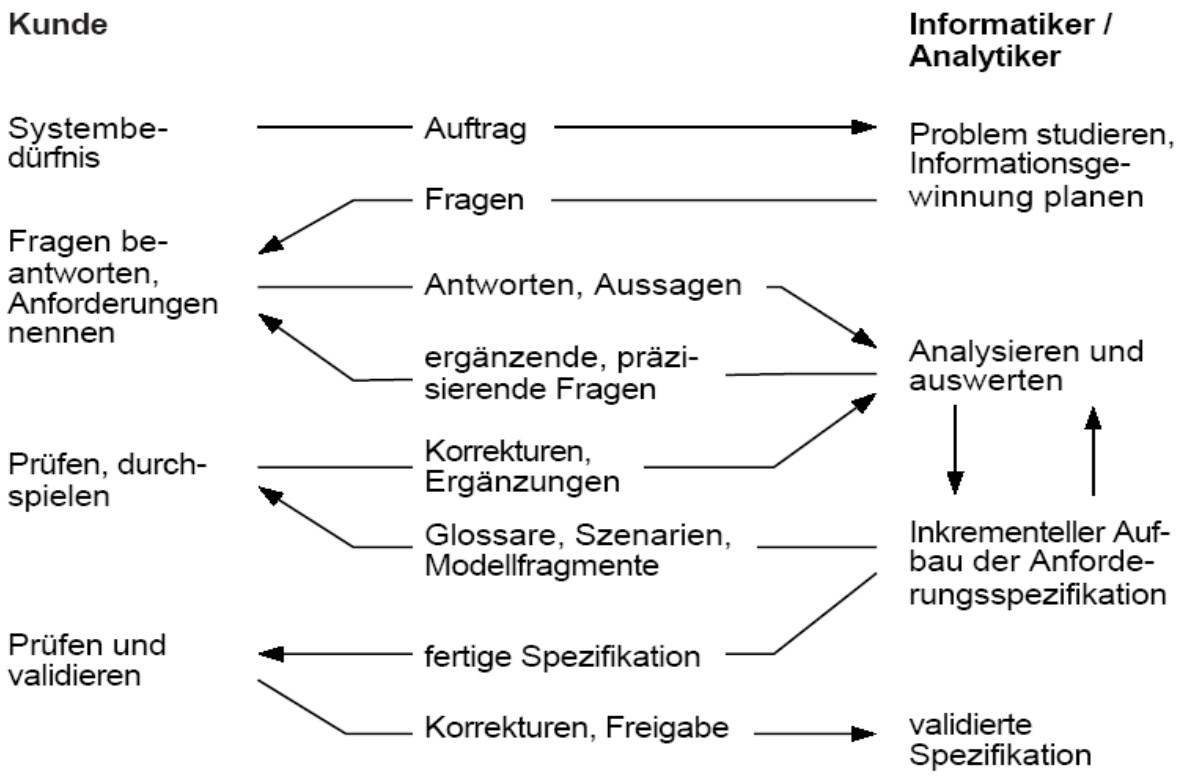

Abbildung 4. 1: Ablauf des Spezifikationsprozesses [GLINZ_SP]

Um eine besseres Verständnis bezüglich den gewünschten Anforderungen zu bekommen, habe ich zwischen den verschiedenen Teilprozessen des Spezifikationsprozesses schrittweise eine Ist-Anaylse des Systems bzw. des mobileGames vorgenommen. Schrittweise deshalb, weil einen einmalige Ist-Analyse nicht sinnvoll und nur schwer realisierbar gewesen wäre. Dabei habe ich mich anfangs primär auf die schon vorhandenen Funktionen bzw. Fähigkeiten des Systems konzentriert, weniger auf die technischen Aspekte der Implementierung. Dieser Fokus vergrößerte sich im Laufe der Zeit jedoch auch auf die Architektur des Systems und umfasste schlussendlich das komplette mobileGame. Die Ist-Analyse beinhaltete folgende Schritte:

- Verschiedene Modelle der schon vorhandenen Implementierung wurden angeschaut und analysiert. Diese Modelle wurden in den meisten Fällen nicht von mir entworfen sondern standen mir aus vergangenen Diplomarbeiten bezüglich des mobileGames zur Verfügung.

- Generelles Studium der vorhandenen Dokumentationen. Dabei handelte es sich um vergangene Arbeiten so wie auch um die im Code eingebettete Dokumentation.

- Befragung von Personen die bei des Entwicklung und Weiterentwicklung des mobileGame beteiligt waren oder sind.

- Eigenständige Installation des mobileGames mit anschließendem kennen lernen aller implementierten Funktionen und durchspielen von Szenarien. Diese Szenarien dienten dazu zu sehen, was das System kann und nicht kann.

- Schlussendlich hab ich mir den eigentlichen Programmcode des Clients und des Server des mobileGames angeschaut, um die Struktur und vor allem die Ausbaufähigkeit des Systems zu eruieren.

Die Ist-Analyse diente auch dazu, Stärken und Schwächen zu erkennen und die Teile des Programms zu finden, welche übernommen und erweitert werden sollen. Prinzipiell stellte die Ist-Analyse von Anfang an die notwendige Grundlage dar, effiziente und zielgerichtete Gespräche mit meinem Betreuer bezüglich der Spezifikation der Anforderung zu führen. In den verschiedenen Interviews mit Christoph Göth wurden die einzelnen Anforderungen an die History-Funktionalität festgelegt. Dabei gab mir der Betreuer zu erkennen, dass ich bezüglich einzelnen Aspekten große Freiheiten habe. Diese Freiheiten sollten eine möglichst breite Ausgestaltung einzelner Faktoren fördern und im Endeffekt eine möglichst optimale Lösung, die mehrere Lösungen miteinander vereint, hervor bringen.

In den verschiedenen Interviews mit Christoph Göth wurden zusammengefasst folgende Anforderungen und Vorstellungen festgehalten:

- Zur Orientierungshilfe soll der Benutzer die Möglichkeit haben seinen zurückgelegten Weg auf dem PDA zu sehen. Der zurückgelegte Weg soll direkt auf der Karte ersichtlich sein.

- Die Speicherung der Wegpunkte geschieht lokal auf dem Client bzw. nicht beim Server.

- Die Wegpunkte müssen nicht persistent gespeichert werden. Sie werden temporär auf dem Client zwischengespeichert und bei Ende des Spiels wieder gelöscht.

- Mehrere Darstellungsarten sollen verfügbar sein.

- Die Darstellungsarten können auch miteinander kombiniert werden.

- Unter den verschiedenen Darstellungsarten sollte eine „nebelartige" Darstellung zumindest getestet werden.

- Der Benutzer soll spezielle Wegpunkte setzen können. Diese sind Teil des zurückgelegten Weges.

- Die Darstellungsart von diesen speziellen Wegpunkten soll sich von derjenigen der normalen Wegpunkte unterscheiden.

- Die Wegdarstellung muss über eine einfache Taste auf dem Bildschirm ein- und ausschaltbar sein.

- Die Wegdarstellung muss Rücksicht auf die Rechenleistung und dem Speichervolumen des PDA's nehmen.

- Die Entwicklung sollte, wie die schon vorhandene Software, auf der Programmiersprache Java basieren.

- Die Wegdarstellung sollte auch bei Störungen wie geringe Bandbreite oder bei kurzzeitigen Verbindungsabbrüchen funktionieren. Sie sollte auch bei Schwierigkeiten mit der eigentlichen Positionierung den zurückgelegten Weg adäquat aufzeigen können.

- Die History-Funktionalität soll den Umstand, dass die Wegdarstellung bei kreisförmigen Bewegungen mit vielen Überschneidungen und daher hohen Redundanzen unübersichtlich werden kann, angemessen berücksichtigen.

- Die Wegdarstellung muss über mehrere Stockwerke funktionieren.

- Welche Darstellungsart benutzt wird, wird in der schon vorhandenen Konfigurationsdatei festgelegt.

Nachdem die Anforderungen festgehalten worden sind, wurden diese zusätzlich nach ihrer Wichtigkeit klassifiziert. Eine Klassifikation der Anforderungen diente in diesem Falle hauptsächlich zur Priorisierung der gewünschten Fähigkeiten und Eigenschaften die das zu entwickelnde System haben soll. Die Anforderungen wurden in Muss- und Wunsch- Anforderungen klassifiziert. Muss- Anforderungen sind Anforderungen, die unverzichtbar und auf jeden Fall realisiert werden müssen [GLINZ_SE]. Wunsch- Anforderungen dagegen sind Fähigkeiten und Anforderungen, die realisiert werden, wenn es gewisse kritische Faktoren zulassen. Die kritischen Faktoren variieren je nach Gegebenheit stark. In diesem Falle zählte primär die Zeit als kritischer Faktor. Dies umso mehr, da ich nur schwer einschätzen konnte, wie lange die vollständige Realisierung der festgelegten Muss-Anforderungen dauern wird.

Muss-Anforderungen
Zur Orientierungshilfe soll der Benutzer die Möglichkeit haben, seinen zurückgelegten Weg auf dem PDA zu sehen. Der zurückgelegte Weg soll direkt auf der Karte ersichtlich sein
Mehrere Darstellungsarten sollen verfügbar sein
Die Darstellungsarten können auch miteinander kombiniert werden
Die Speicherung der Wegpunkte geschieht lokal auf dem Client bzw. nicht beim Server
Die Wegpunkte müssen nicht persistent gespeichert werden. Sie werden temporär auf dem Client zwischengespeichert und bei Spielende wieder gelöscht
Die Wegdarstellung muss über eine einfache Taste auf dem Bildschirm ein- und ausschaltbar sein
Die Wegdarstellung muss Rücksicht auf die Rechenleistung und dem Speichervolumen des PDA‘s nehmen
Die Visualisierung soll den Umstand, dass die Wegdarstellung bei kreisförmigen Bewegungen mit vielen Überschneidungen und daher hohen Redundanzen unübersichtlich werden kann, angemessen berücksichtigen
Die Wegdarstellung muss über mehrere Stockwerke funktionieren

Tabelle 1: Die Muss-Anforderungen

Wunsch-Anforderungen
Unter den verschiedenen Darstellungsarten sollte eine „nebelartige“ Darstellung zumindest getestet werden. Der Aspekt einer optimalen Transparenz bei der Darstellung ist zu berücksichtigen
Der Benutzer soll spezielle Wegpunkte setzen können. Diese sind Teil des zurückgelegten Weges
Die Darstellungsart von diesen speziellen Wegpunkten soll sich von derjenigen der normalen Wegpunkte unterscheiden
Die Entwicklung sollte, wie die schon vorhandene Software, auf der Programmiersprache Java basieren
Welche Darstellungsart benutzt wird, wird in der schon vorhandenen Config Datei festgelegt

Tabelle 2: Die Soll-Anforderungen

4.1.2 Darstellung der Anforderungen

Zur Darstellung wird die konstruktive Darstellungsart gewählt. Eine konstruktive Darstellungsart modelliert das zu spezifizierende System als eine Menge interagierenden Komponenten [GLINZ_SE]. Einge Anforderungen werden nun durch entsprechende Anwendungsfälle beschrieben. Die Menge der Anwendungsfälle wird in einem Anwendungsfalldiagramm dargestellt. Das Anwendungsfalldiagramm zeigt den gesamten Systemkontext des zu weiterentwickelnden mobileGames.

4.1.2.1 Das Anwendungsfalldiagramm

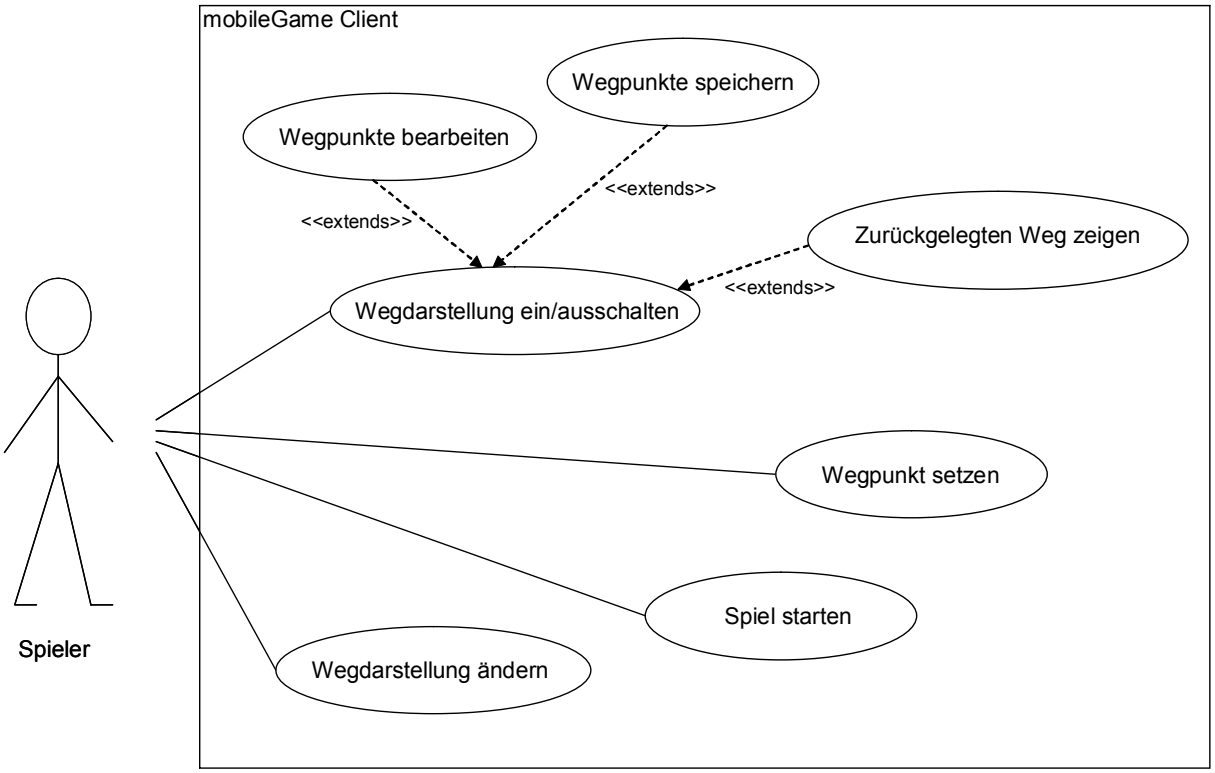

Abbildung 4. 2: Anwendungsfalldiagramm

4.1.2.2 Anwendungsfälle

Die im Anwendungsfalldiagramm enthaltenen Anwendungsfälle werden nun genauer beschrieben.

- **Spiel starten:**
  Der Client meldet sich beim Server an. Der Server übermittelt die nötigen Daten und das Spiel beginnt. Der Benutzer sieht die Karte und seine aktuelle Position.

- **Wegdarstellung ein/ausschalten:**
  Der Spieler aktiviert die Wegdarstellung über ein Bedienelement auf der

Benutzeroberfläche. Die schon vorhandene Benutzeroberfläche muss entsprechend erweitert werden. Bei Spielstart ist die Wegdarstellung standardmäßig deaktiviert.

- **Wegdarstellung ändern:**
  Die Wegdarstellungsart wird in der schon vorhandenen Konfigurationsdatei festgelegt. Will der Spieler diese ändern, muss er den entsprechenden Eintrag in der Konfigurationsdatei anpassen. Eine Änderung in der Konfigurationsdatei hat einen Neustart des Spiel zur folge, damit diese Änderungen aktiv werden.

- **Wegpunkte bearbeiten:**
  Die vom Server empfangenen Positionen werden durch den Client nach gewissen Faktoren gefiltert. Diese Filterung hat das Ziel, einen möglichst optimalen Datensatz an Wegpunkten zu generieren.

- **Wegpunkte speichern:**
  Die gefilterten Wegpunkte werden im Client transistent zwischengespeichert um später für die Darstellung zur Verfügung zu stehen. Neben den eigentlichen Positionsangaben werden noch weitere Informationen wie Zeit, Datum, zugehörige Karte usw. in den Datensätzen festgehalten.

- **Zurückgelegten Weg zeigen:**
  Die verarbeiteten und gespeicherten Wegpunkte werden je nach gewählter Darstellungsart auf dem PDA bzw. direkt auf der Übersichtskarte dargestellt. Sind mehrere Ebenen bzw. Stockwerke vorhanden wird nur der Weg im aktuellen Stockwerk gezeigt.

- **Wegpunkt setzten**:
  Über ein Bedienelement auf der Benutzeroberfläche setzt der Spieler einen speziellen Wegpunkt. Dieser entspricht dem letzten gespeicherten Wegpunkt in Moment der Betätigung des Bedienelementes.

Anwendungsfall: Spiel starten	
Akteur	Spieler
Systeme	Clientprogramm
Referenzen	-
Bemerkungen	Der Server der mobileGames muss gestartet worden sein
Ablauf	1. Die Konfiguration wird beim Client ausgelesen 2. Diese Daten werden dem Server übermittelt wo sie geprüft werden 3. Sind die Daten korrekt, speichert der Server die IP-Adresse des Clients und gibt eine entsprechende Antwort dem Client 4. Der Client empfängt diese Antwort und startet das Spiel
Fehler	1. Antwortet der Server nach einer gewissen Zeitspanne nicht wird der Client beendet

Tabelle 3: Anwendungsfall: Spiel starten

Anwendungsfall:Wegdarstellung ein/ausschalten	
Akteur	Spieler
Systeme	Clientprogramm
Referenzen	Anwendungsfall: Spiel starten
Bemerkungen	Der Client muss mit dem Server Kontakt aufgenommen haben und gestartet worden sein
Ablauf	1. Die Positionsdaten werden laufend im Client bearbeitet und zwischengespeichert 2. Der Spieler betätigt das entsprechende Bedienelement auf der Benutzeroberfläche 3. Der Client liest die zwischengespeicherten Wegpunkte aus, und stellt diese mit der Darstellungsart, welche aus der Konfigurationsdatei ausgelesen wurde, entsprechend dar 4. Will der Spieler die Wegdarstellung ausschalten, betätigt er wiederum das entsprechende Bedienelement auf der Benutzeroberfläche des Clients
Fehler	-

Tabelle 4: Wegdarstellung ein/ausschalten

Anwendungsfall:Wegdarstellung ändern	
Akteur	Spieler
Systeme	Clientprogramm
Referenzen	-
Bemerkungen	Der Client muss beendet worden sein. Bei jeder Änderung der Wegdarstellung in der Konfigurationsdatei muss der Client neu gestartet werden
Ablauf	1. Läuft der Client bereits, so wird dieser beendet 2. Die Konfigurationsdatei wird geöffnet und der entsprechende Eintrag bei „ph_visualization“ abgeändert 3. Die Änderung wird gespeichert 4. Der Client wird neu gestartet
Fehler	Wird beim Eintrag „ph_visualization“ eine ungültige Darstellungsart angegeben, so wird die Wegdarstellung nicht angezeigt

Tabelle 5: Anwendungsfall: Wegdarstellung ändern

Anwendungsfall:Wegpunkte bearbeiten	
Akteur	Client
Systeme	Clientprogramm
Referenzen	Anwendungsfall: Wegpunkte speichern
Bemerkungen	Der Client ist gestartet. Die Wegpunkte- Übermittlung von Server zum Client findet laufend statt
Ablauf	1. Der Client schaut sich jeden neu eingetroffenen Wegpunkt an 2. Ist der neue Wegpunkt identisch mit dem vorherigen Wegpunkt, so wird dieser nicht zwischengespeichert 3. Ist der Wegpunkt nicht identisch mit dem vorherigen Wegpunkt, so wird dieser in Betracht gezogen 4. Je nach gewählter Darstellungsart, welche aus der Konfigurationsdatei ausgelesen wird, wird zusätzlich geprüft, ob der neu erhaltene Wegpunkt spezifische Kriterien erfüllt 5. Die neuen Wegpunkte, welche diese Kriterien, abhängig von der gewählten Wegdarstellung, erfüllen, werden endgültig abgespeichert und für die finale Darstellung zur Verfügung gestellt
Fehler	-

Tabelle 6: Anwendungsfall: Wegpunkte bearbeiten

Anwendungsfall:Wegpunkte speichern	
Akteur	Client
Systeme	Clientprogramm
Referenzen	Anwendungsfall: Wegpunkte bearbeiten
Bemerkungen	Die Speicherung der Wegpunkte geschieht peristent. Sie werden daher bei Beendigung des Spieles wieder gelöscht
Ablauf	1. Die im Client bearbeiteten bzw. gefilterten Wegpunkte werden als Wegpunktobjekte in einer Hashtabelle gespeichert 2. Die Wegpunkte enthalten neben den Positionsangaben in Form von Koordinaten zusätzliche Informationen wie Zeit, Zeitdifferenz zum vorherigen Wegpunkt, Datum, Kartenname, Wegpunkttyp usw.
Fehler	-

Tabelle 7: Anwendungsfall: Wegpunkte speichern

Anwendungsfall: Wegpunktedarstellen	
Akteur	Client
Systeme	Clientprogramm
Referenzen	Anwendungsfall: Wegdarstellung ein/ausschalten
Bemerkungen	Die Wegpunkte wurden vorgängig bearbeitet und gespeichert
Ablauf	1. Wird vom Spieler die Wegdarstellung über das Bedienelement eingeschaltet, so wird intern ein entsprechendes „Flag“ gesetzt 2. Ist das Flag gesetzt und die Darstellungsart bekannt, so wird in einer separaten „Renderklasse“ der zurückgelegte Weg direkt auf der Übersichtskarte des Clients gezeichnet.
Fehler	-

Tabelle 8: Anwendungsfall: Wegpunkte darstellen

Anwendungsfall:Wegpunkt setzten	
Akteur	Spieler
Systeme	Clientprogramm
Referenzen	-
Bemerkungen	Es kann nur immer der letzte Punkt des Weges als spezieller Wegpunkt gesetzt werden
Ablauf	1. Spieler aktiviert über ein Bedienelement die Wegdarstellung 2. Über ein weiteres Bedienelement ändert er den letzten Wegpunkt der Darstellung in einen speziellen Wegpunkt um 3. Der letzte Wegpunkt der Aufzeichnung ändert seine Form und ist nun als spezieller Wegpunkt erkennbar
Fehler	-

Tabelle 9: Anwendungsfall: Spezielle Wegpunkte setzen

4.1.3 Prüfen der Anforderungen

Bei der Prüfung von Anforderung geht es darum, Abweichungen von der geforderten Qualität der Anforderungen festzustellen [GLINZ_SE]. Redundanzen, Fehler und Unklarheiten sollen vermieden werden. Primär geht es darum, die am Anfang dieses Kapitells festgehaltenen Kriterien wie Widerspruchsfreiheit, Vollständigkeit, Adäquatheit, Eindeutigkeit usw. zu überprüfen. Die Prüfung der Anforderungen fand immer dann statt, als die Anforderungen festgelegt waren und noch genügend Zeit vorhanden war, die gefundenen Mängel zu beheben. Die festgelegten Anforderungen wurden von mir vor allem durch Prototypen mehrfach geprüft. Prototypen sind eine der mächtigsten verfügbaren Mittel, um die Anforderungen zu überprüfen, auch wenn sie einen großen Aufwand erfordern [GLINZ_SE]. Die von mir entwickelten Prototypen wurden konsequent den festgehaltenen Anforderungen entgegen gestellt. Die einzelnen Prototypen enthielten anfangs oft nur eine bestimmte geforderte Funktionalität, welche zwischen den verschiedenen Prüfprozessen ständig weiter entwickelt, verbessert oder in gewissen Fällen auch komplett neu entwickelt wurde. Neben der Prüfung mit Prototypen wurden auch Reviews in Form von einzelnen Treffen mit meinem Betreuer Christoph Göth geführt. Diese Treffen war sehr wichtig, da generell das Einbinden des Kunden bzw. des Betreuers in den Prüfprozess zwingend und daher sehr wichtig ist, da schlussendlich nur er die Adäquatheit und Vollständigkeit der Spezifikation objektiv beurteilen kann.

4.2 Entwurf der History-Funktionalität

Nach dem die Anforderungen spezifiziert, dargestellt und geprüft wurden, folgt nun der Entwurf der Architektur der History-Funktionalität. Grundlage des schon vorhandenen Systems ist die objektorientierte Architektur. Konsequenterweise entschied ich mich, diese Architektur bezüglich der Entwicklung der History-

Funktionalität beizubehalten. Eine objektorientierte Architektur beinhaltet folgende Aspekte [GLINZ_SP]:

- **Vererbung**
  Bezeichnet die Relation zwischen Klassen, welche eine Hierarchie von Ober- und Unterklassen bilden, so dass jede Unterklasse neben ihren eigenen Eigenschaften auch alle Eigenschaften aller Oberklassen hat, sofern sie diese nicht explizit abändert oder ausschließt.

- **Polymorphismus**
  Möglichkeit, eine Operation gleichen Namens für Objekte verschiedener Klassen mit möglicherweise verschiedener Wirkung zu definieren

- **Das Geheimnisprinzip** (Information Hiding)
  Dem Benutzer bzw. Anwender wird nur das Ergebnis präsentiert, wie die Software auf das Ergebnis gekommen ist, bleibt für den Benutzer geheim.

- **Datenabstraktion**
  Eigene Datentypen mit Konstruktions-, Selektions- und Typprüffunktionen werden gebildet, mit dem Ziel, die Benutzung der Daten von der eigentlichen Implementierung zu trennen.

Die Architektur besteht aus Objekten, die in Form von Klassen repräsentiert werden. Die objektorientierte Architektur stellt auch durch den Einsatz der Programmiersprache Java eine Notwendigkeit dar.

Die Struktur des mobileGame Client ist in drei verschiedene Schichten unterteilt, die untereinander neben den normalen Aufrufen auch über Ereignisse (Events) kommunizieren. Die Ereignisbasierenden Architekturen unterstützen die Entkoppelung der Komponenten stark. Die Unterteilung in drei Schichten erhöht generell auch die Erweiterbarkeit, was mir bei der Entwicklung der History-Funktionalität ebenfalls sehr entgegen kam. Das mobileGame ist in folgende Schichten unterteilt.

- **Netzwerkschicht**
  Die Netzwerkschicht oder auch Verbindungsschicht ist für die Kommunikation des Clients mit dem Server des mobileGames zuständig. Die Kommunikation basiert auf Ereignissen (Event-Based).

- **Logikschicht**
  Die Logikschicht des Clients beinhaltet die zentralen Datenstrukturen und Funktionen des mobileGames. Annotationen, Teams, Gruppen, POI (Point of Interest), Positionen bzw. Positionsinformationen usw. werden in dieser Schicht verarbeitet und abgespeichert.

- **Präsentationsschicht**
  Die Präsentationsschicht umfasst diverse Funktionen bezüglich der Benutzeroberfläche des Clients. Unter anderem die verschiedenen Bedienelemente und die Darstellung der Übersichtskarte und des Spieles.

Meine Absicht war es nun, die vorhandene Unterteilung in drei Schichten beizubehalten und auch die bereitgestellten Funktionalitäten und Klassen so weit wie möglich zu verwenden. Das Ziel war es möglichst viel vorhandene Software

zu verwenden und diese problemspezifisch anzupassen. Von schon vorhandenen Klassen neue selbst entwickelte Klassen abzuleiten, so dass diese die gestellten Anforderungen erfüllen. Die Prinzipien der Delegation und der Generalisierung bleiben dabei erhalten. Neue Klassen sollten sich wie die schon vorhandenen Klassen auf vorhandene Dienstleistungen abstützen (Delegation) und ähnliche Dienstleistungen verschiedener neuen und alten Klassen sollten in einer neuen gemeinsamen Oberklasse vereinigt werden (Generalisierung). Auch der Aspekt der Komposition, welcher gemeinsame Klassen zu einer Einheit zusammenfasst, sollte berücksichtigt werden. Bei der Verwendung von vorhandenen Klassen kamen zwei Verfahren in Frage:

- **Black-Box-Verwendung**
  Bei dieser Art der Verwendung werden die vorhandenen Operationen genutzt. Das Geheimnisprinzip wird beibehalten und zusätzlich verstärkt. Die potentiellen Nebenwirkungen sind gering.

- **White-Box Verwendung**
  Die Verwendung bzw. die Nutzung erfolgt hier durch Ableitung von Unterklassen. Diese Methode ist sehr flexibel kann aber das Geheimnisprinzip beeinträchtigen und größere Nebenwirkungen als Folge haben.

Bezüglich der History-Funktionalität wurden hauptsächlich die Logik- und Präsentationsschicht tangiert. Alle notwendigen Informationen, die bei der Implementierung der neuen Funktionalitäten essentiell waren, sind schon in der Kommunikationsstruktur des Clients integriert, so dass die Netzwerkschicht von der geplanten Erweiterung nicht betroffen wurde. Die Netzwerkschicht wurde also so belassen wie sie ist.

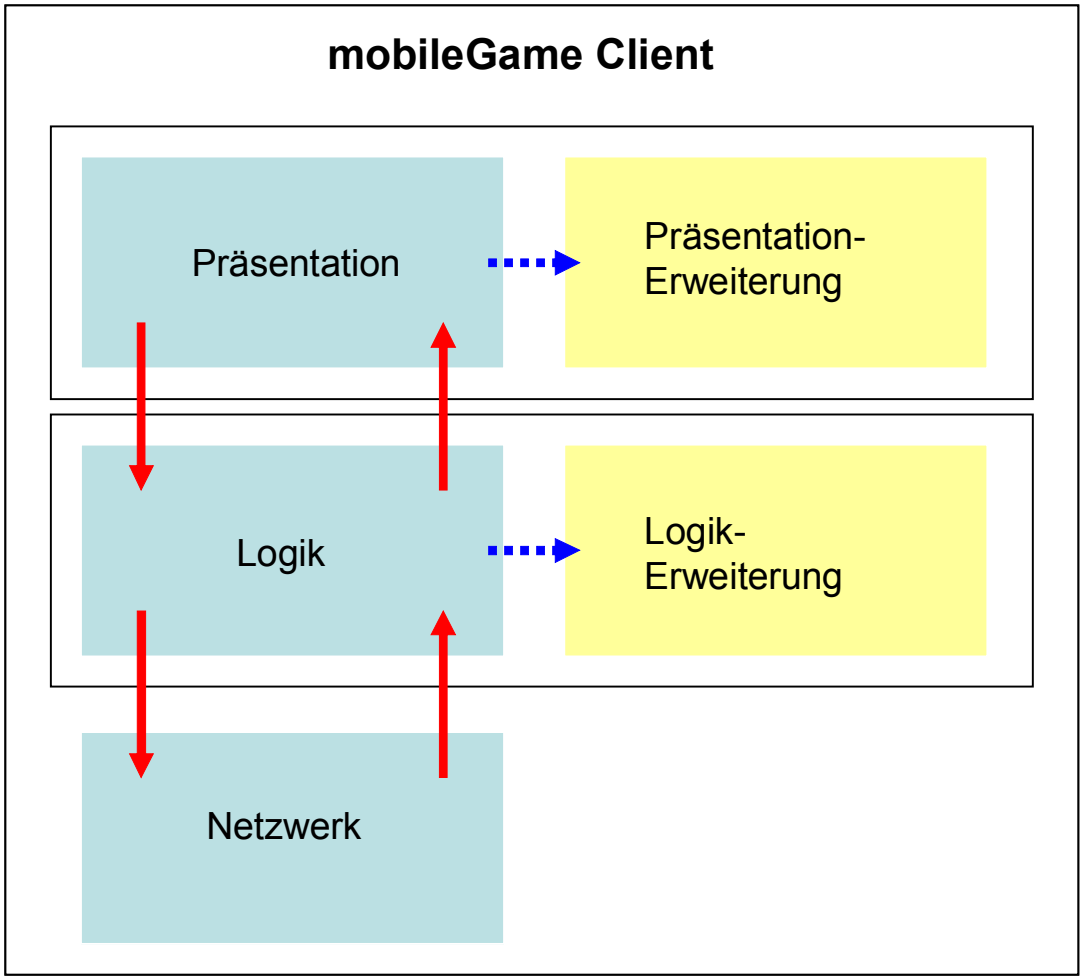

Abbildung 4. 3: Schichten des mobileGames und die Erweiterungen

4.3 Implementierung der History-Funktionalität

Bei der Implementierung der History-Funktionalität wurde ausschließlich der Client tangiert. Der Server des mobileGames war von keiner Änderung betroffen. Weiter musste bei der Benutzung von schon vorhandenen Java-Klassenbibliotheken darauf geachtet werden, dass die für den PDA passende Java-Runtime diese auch unterstützten. Beim mobileGame wurde eine Version der IBM J9-Runtime[38] verwendet. Der Einsatz der J9-Runtime brachte gewisse Restriktionen mit sich. Die SWT[39] Version, die diese J9-Runtime beinhaltet, ist eine Grundversion und beinhaltet nicht alle Funktionalitäten. Ich musste deshalb die verschiedenen Klassenbibliotheken mit Vorsicht benutzen und immer darauf achten, dass diese schlussendlich auch von der J9-Runtime unterstützt werden. Von diesen Restriktionen war primär die Erweiterung bezüglich der Präsentationsschicht betroffen, da gerade bei der Darstellung der History-Funktionalität, die Auswahl an Java-Bibliotheken aber auch an anderen Ressourcen wie zum Beispiel OpenGL[40]- Bibliotheken sehr groß ist.

[38] http://www-306.ibm.com/software/wireless/weme/
[39] Standard Widget Toolkit, http://www.eclipse.org/swt/
[40] http://opengl.j3d.org/swt/

4.3.1 Implementierung der erweiterten Logikschicht

Jeder Client meldet sich über die Netzwerkschicht beim Server an. Ist die Anmeldung erfolgreich so wird auf dem Client das Spiel geladen. Nach dem der Client erfolgreich gestartet wurde, wird unter anderem die Klasse *GameStore* mit einbezogen. Diese wird bei Applikationsstart über die *ClientLogicFacade* Klasse initialisiert. Die *ClientLogicFacade* bildet die Fassade für den Zugriff auf die Logikschicht. Hier werden alle Logik-Module erzeugt. Über die verschiedenen Logik-Module werden die Funktionalitäten des mobileGames ausreichend gekapselt. Diese Kapselung fördert die Erweiterbarkeit massiv. Auch alle Zugriffe der Benutzeroberfläche müssen über diese Klasse laufen. Die Klasse *GameStore* stellt die zentrale Einheit der Logikschicht stellt dar. In dieser Klasse wird das aktuell laufende Spiel abgelegt und verwaltet. Die gesamten Spieleraktionen sowie die verschiedenen eintreffenden Aktionen werden in dieser Klasse verarbeitet und an die Präsentationsschicht weiter geleitet. Die *GameStore* Klasse beinhaltet alle Annotationen, Teams, Gruppen, POI's (Point of Interests) und generelle Positionsinformationen der Spieler Im *GameStore* wird das *PropertyChangeListener* Interface implementiert, mit welchem auf die verschiedenen einkommenden Nachrichten reagiert werden kann.

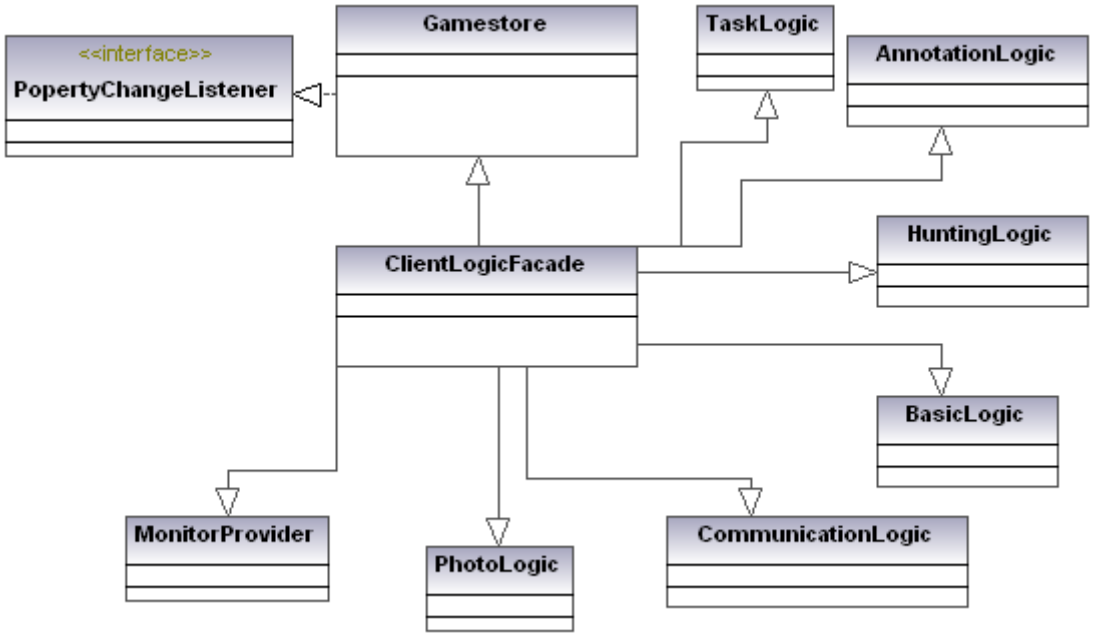

Abbildung 4. 4: Teil der Logikschicht des Clients

Das Herzstück des Clients, die *GameStore* Klasse, enthält unter vielen anderen Methoden auch die Methode *positionChanged.* Diese Methode bekommt als Parameter ein Ereignis welches durch eine Positionsänderung ausgelöst wurde. Trifft ein solches Ereignis ein, so wird die neue Position samt Teamname in einem Array zwischengespeichert. Dieses Array wird wiederum später für die History-Funktionalität in der *FadingHistory* Klasse benutzt. Die *FadingHistory* Klasse befindet sich im *data.game.history* Packet. Ursprünglich war es so, das die empfangenen neuen Positionen eines Teams vollumfänglich und ohne Berücksichtigung gewisser Kriterien in einem Array zwischengespeichert wurden. Es wurden also alle Positionen die der Server dem Client übermittelte aufgenommen und abgespeichert. Dieser Umstand war insofern kritisch und

daher fraglich, da die Menge an unbrauchbaren bzw. ineffizienten Daten enorm war. Mehrere Analysen, die ich bezüglich dieses Umstands durchgeführt habe, zeigten deutlich, dass die Redundanz in diesem Zwischenspeicher enorm war. Daher, ein sehr großer Anteil der Daten waren identisch. Zudem stellt der Datentyp Array, in welchem Unmengen an Daten zwischengespeichert wurden, ebenfalls einen kritischen Faktor dar, da die Größe des Array bei seiner Erzeugung festgelegt werden muss und sich nicht mehr verändern lässt. Diese Einschränkung stellt gerade bei Echtzeitanwendungen ein nicht zu unterschätzendes Risiko dar, da hier eine große Flexibilität und Dynamik bezüglich den Datentypen von Vorteil ist. Weiter stellte ich bei diesen Analysen fest, dass trotz der sehr hohen Redundanz, die Daten korrekt waren. Mit korrekt ist vor allem die Aktualität der Daten gemeint. Der zuletzt vom Server übermittelte und vom Client zwischengespeicherte Datensatz widerspiegelte tatsächlich die zuletzt eingenommene Position des Clients.

Wie ich in Abschnitt 4.2 erwähnt habe, war es mein Ziel vorhandene Software zu nutzen, soweit diese den Anforderungen gerecht werden. Durch meine Analysen der betroffenen Teile der Logikschicht und der dazu gehörenden *FadingHistory* Klasse, entschied ich mich diese bestehen zu lassen, sie aber an gewissen kritischen Stellen zu verändern und generell um zusätzliche, notwendige Funktionalitäten zu erweitern. Mein Augenmerk richtete sich anfangs also primär auf die *FadingHistory* Klasse.

Um die zwischengespeicherten Datensätze effizienter und generischer zu gestalten, mussten gemäß den Ergebnissen meiner Analysen folgende Aspekte angegangen werden:

- Die Redundanzen der zwischengespeicherten Positionsinformationen.
- Die Verwendeten Datentypen zur Speicherung der History-Daten.
- Der Informationsgehalt der Daten.

Als erstes verfolgte ich das Ziel, die auftretenden Redundanzen in den Datensätzen zu vermeiden. Die Tatsache, dass ein großer Teil der Positionen identisch waren, erforderte eine grundlegende Änderung. Zu diesem Zeitpunkt machte ich mir auch Gedanken, ob das Weglassen von Redundanzen später zu Problemen führen kann. Dabei stellte sich die Frage, ob diese Redundanzen nicht auch nützlich sind. Ein Weglassen dieser mehrfach vorkommenden identischen Datensätzen stellte in einem gewissen Sinne auch eine Verringerung des Informationsgehaltes dar. Aus diesen Überlegungen kam ich zum Schluss, das Array mit den redundanten Daten nicht ganz weg zu lassen, sondern den Speichertyp beizubehalten. Zusätzlich entschied ich mich, die drei aufgezeigten Problembereiche gemeinsam zu lösen, da sie implizit voneinander abhängig waren. *Die FadingHistory* Klasse wurde folgendermaßen verändert:

- Eine neue Methode *DataFilterung* wurde implementiert. Diese Methode erzeugt eine Hashtabelle (engl. Hashtable) mit Vektoren als Elemente. Diese Vektoren sind nach Kartennamen benannt und enthalten verschiedene Informationen bezüglich eines übermittelten Datensatzes. Die Vektoren bestehen aus *HistoryData* Objekten. Diese Objekte widerspiegeln einen Datensatz und enthalten folgende Informationen:
    - Die Positionskoordinaten in Pixel

- Die genau Zeitangabe in Stunden, Minuten und Sekunden in Integer und String Form
- Ein Feld für spezielle Informationen in Form eines Strings
- Ein Feld in dem der Typ eines Datensatzes festgehalten wird. Der Typ legt im Moment fest, ob es sich um einen normalen oder speziellen Wegpunkt handelt. In Zukunft können aber noch andere Typen definiert werden
- Eine Nummerierung die den Wegpunkt eindeutig kennzeichnet.

Die Vektoren welche in der *DataFilterung* Methode erzeugt werden, enthalten keine Redundanzen, daher, identische Datensätze werden gefiltert. Es kann also nicht vorkommen, dass in den Vektoren zwei aufeinander folgende Wegpunkte identisch sind. Die Hashtabelle ist das eigentliche Fundament der History-Funktionalität. Alle gefilterten Positionen aller Karten werden hier persistent gespeichert.

- Eine zusätzliche Methode *getHistoryFromMapAsVector* wurde eingebaut. Diese Methode, liefert einen Vektor von Datensätzen zurück. Diese Vektoren enthalten keine Redundanzen dafür zusätzliche nützliche Informationen

- Der Konstruktor der *FadingHistory* Klasse wurde leicht abgeändert und erweitert

Durch die Vermeidung von Redundanzen in den Vektoren und das Hinzufügen anderer nutzbarer Informationen wurde die Qualität der Informationen und daher auch der abgespeicherten Datensätzen entscheidend erhöht. Bezüglich der Filterung von Daten beschränkte ich mich auf die Koordinaten der Positionen. Identische Datensätze sind demnach diejenigen welche die gleichen x- und y-Koordinaten besitzen. Eine weitere Filterung nach anderen Kriterien erfolgt später bei der Darstellung dieser Daten.

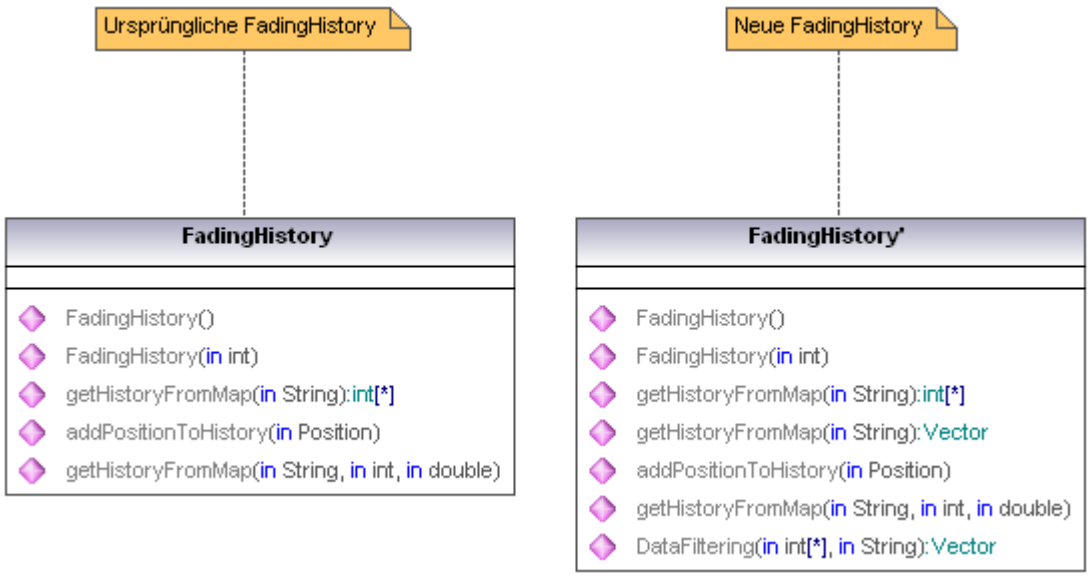

Abbildung 4. 5: Die alte und neue *FadingHistory* Klasse

Das *HistoryData* Objekt ist der eigentliche Daten-Container. Diese Klasse kann sehr leicht um weitere Datenfelder erweitert werden. Bei der Darstellung der History-Daten werden hauptsächlich diese Objekte verwendet. Die folgende Abbildung zeigt die *HistoryData* Klasse. Aus Platzgründen wurden die Parameter leicht zusammengefasst. Unter der Grafik folgt eine kurze Legende, die noch mal alle in dieser Klasse enthaltenen Parameter beschreibt.

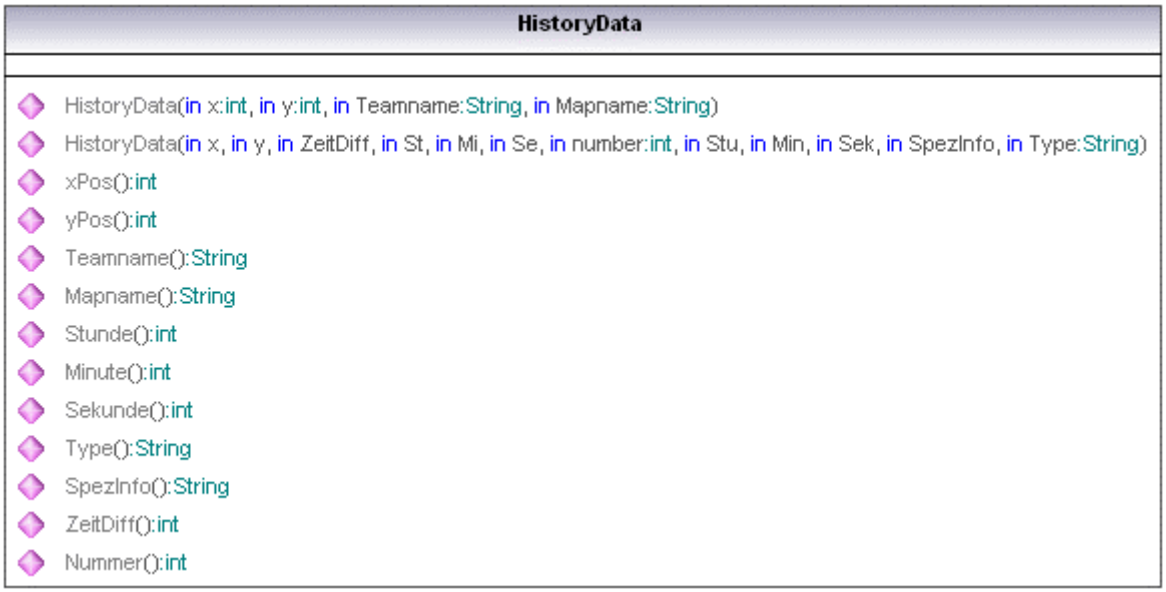

Abbildung 4. 6: Die *HistoryData* Klasse

Legende:

- *xPos und yPos:* Koordinaten des Wegpunktes in Pixel.
- *Teamname:* Name des Teams.
- *Mapname:* Name der Karte auf der sich der Wegpunkt befindet.
- *Stunde/Minute/Sekunde:* Die Zeit bei welcher der Wegpunkt vom Client empfangen wurde.
- *Type:* Gibt an ob es sich um einen normalen oder um einen speziellen Wegpunkt handelt. Im Moment gibt es drei Arten von speziellen Punkten. Die Wegpunkte die der Benutzer selber über das Bedienelement als speziell kennzeichnet, die Punkte bei denen ein Stockwerkwechsel stattgefunden hat und der Startpunkt.
- *SpezInfo:* Falls nötig kann hier eine spezielle Information gespeichert werden.
- *ZeifDiff:* Zeitdifferenz zum vorherigen Wegpunkt.
- *Nummer:* Nummerierung der Wegpunkte. Wichtig für die Typbestimmung.

Durch den Aufruf des Konstruktors mit den dazu nötigen Parametern wird ein History-Objekt erzeugt und initialisiert. Die Hilfsmethoden liefern die entsprechenden Informationen zurück. Diese Objekte werden erst wieder gelöscht, nachdem das mobileGame beendet wurde.

Die Hashtabelle, welche Vektoren mit *HistoryData* Objekten enthält, stellt das Kernelement der Datenstruktur dar. Die Schlüssel (Key) der Hashtabelle sind die unterschiedlichen Namen der Karten. Der Name einer Karte wird durch den Namen der Grafikdatei bestimmt. Heißt eine Grafik mit ihrem Format zum Beispiel „Stockwerk1.jpg", so lautet ein Schlüssel der Hashtabelle „Stockwerk1".

Mittels dieser Schlüssel kann sehr effektiv auf die Elemente der Hashtabelle zugegriffen werden. Die Laufzeit einer Suchanfrage (Query-Request) ist bei Hashtabellen generell äußerst klein, was im Falle der History-Funktionalität jedoch nicht von großer Bedeutung ist, da die Anzahl der Elemente gering ist. Zudem muss die Größe der Hashtabelle, wie auch bei Vektoren nicht vorgängig festgelegt werden. Diese Flexibilität und der daraus resultierenden Dynamik der Datenstruktur stellen einen großen Vorteil dar. Folgende Abbildung zeigt die Datenstruktur als Ganzes:

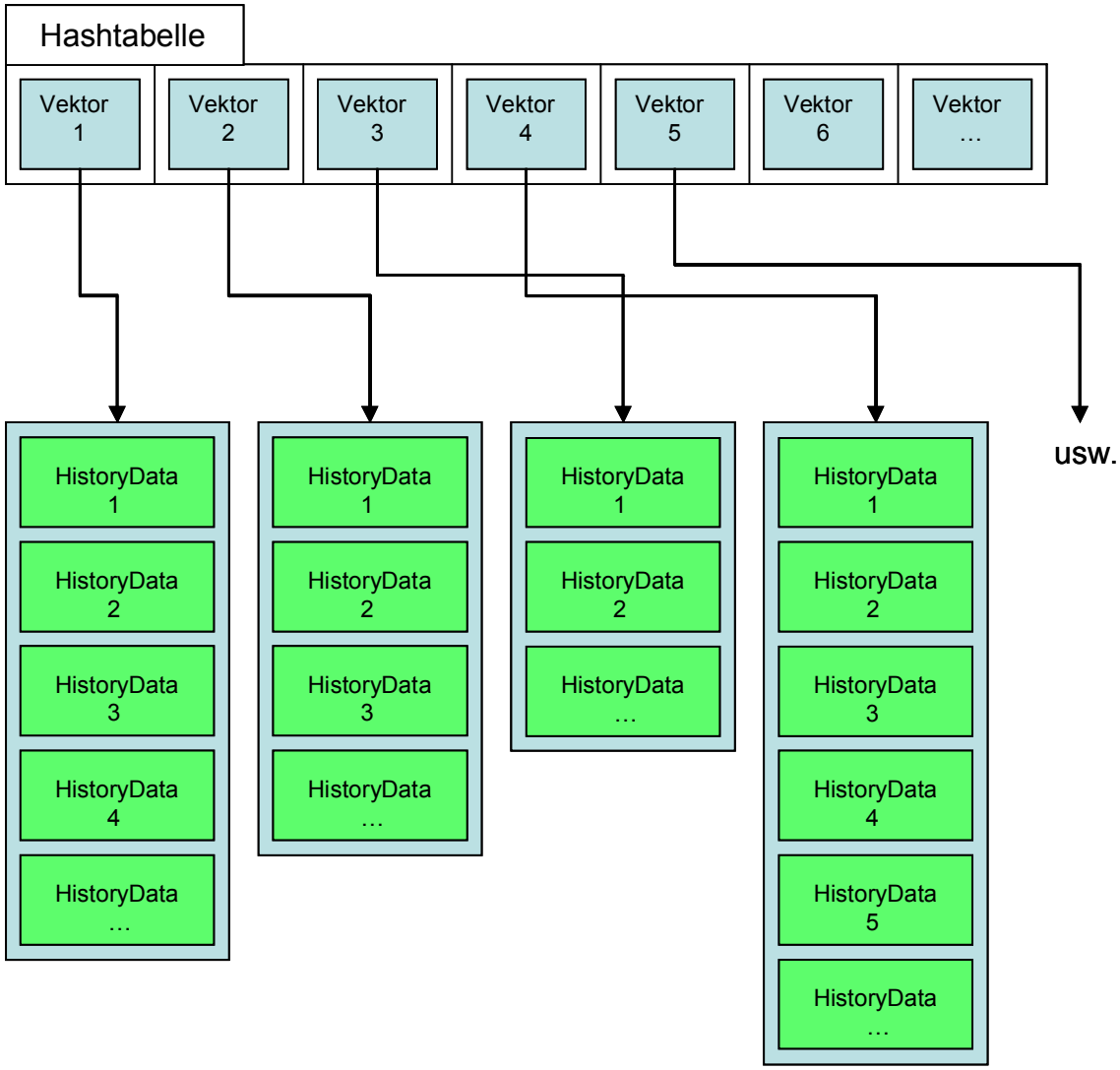

Abbildung 4. 7: Hashtabelle mit den Vektoren

Durch die verschiedenen Veränderungen und Erweiterungen der *FadingHistory* Klasse waren auch andere Klassen der Logikschicht indirekt betroffen. Diese mussten entsprechend angepasst werden.

Wie in diesem Abschnitt deutlich wurde, beinhaltete die Logikschicht vor meiner Arbeit einige wenige rudimentäre Ansätze bezüglich einer History-Funktionalität. Diese Ansätze wurden von mir teilweise übernommen, verändert und vor allem erweitert.

4.3.2 Implementierung der erweiterten Präsentationsschicht

Wie im letzten Abschnitt beschrieben wurde, werden in der erweiterten Logikschicht die History-Daten gefiltert, verarbeitet und schlussendlich in Form von Vektoren in einer Hashtabelle gespeichert. Diese steht der Präsentationsschicht zur Verfügung. Wie im letzten Abschnitt gezeigt wurde, enthält die Klasse *FadingHistory* der Logikschicht die Methode *getHistoryFromMapAsVector,* welche Vektoren als Rückgabeobjekt implementiert. Auf diese Methode greift die Klasse *NGMapRenderer* der Präsentationsschicht zu. Die *getHistoryFromMapAsVecto*r Methode erwartet als Parameter einen Kartennamen. Mit diesem Name wird aus der Hashtabelle der entsprechende Vektor ermittelt und der Präsentationsschicht übergeben. Neben der *getHistoryFromMapAsVector* Methode gibt es noch die *getHistoryFromMap* und eine *getHashtable* Funktion. Die erste liefert statt einem Vektor ein Array gefüllt mit Wegpunkten zurück. Wie in Abschnitt 4.3.1 beschrieben wurde, erfüllen Arrays mit History-Daten nicht die erwünschten Anforderungen. Trotzdem wurde die Methode so belassen um in gewissen zukünftigen Fällen darauf zurückgreifen zu können. Die *getHashtable* liefert die ganze Hashtabelle zurück. Der *NGMapRenderer* enthält folgende Methoden:

- **NGMapRenderer**
  Es handelt sich um den Konstruktor, der ein *NGMapRenderer* Objekt erzeugt und initialisiert. Als Parameter erwartet der Konstruktor eine Referenz eines SWT-Displays.

- **RenderMap**
  Diese Methode zeichnet die ganz Karte neu, inklusive den sich darauf befindenden Objekten wie Spieler, Teams, Gruppen, POI's (Point of Interests) und den zurückgelegten Weg. Dies geschieht durch den Aufruf anderer Methoden, welche sich ebenfalls in der *NGMapRenderer* Klasse befinden. Folgende Methoden sind gemeint:

  - **renderObjects**
    Hier werden alle Objekte des Spiels mit dem entsprechenden Renderer gezeichnet. Die Renderer sind spezielle Klassen, die auf die Darstellung eines Teilbereiches spezialisiert sind.

  - **renderPathHistoryWithPath/renderPathHistoryWithVector**
    Diese zwei Methoden rufen den *NGPathHistoryRenderer* auf. Dieser Renderer stellt den zurückgelegten Weg auf verschiedene Arten dar. Die aufgerufene *NGPathHistoryRenderer* Klasse verfügt mehrere Methoden. Jede davon ist für eine Darstellungsart zuständig.

Die *NGMapRenderer* Klasse stützt sich beim Zeichnen der verschiedenen Objekte auf die entsprechenden Regeln, die in der *Rules* Klasse definiert sind. Es werden nur diejenigen Objekte dargestellt, welche den Regeln entsprechen. Bezüglich der History-Funktionalität mussten hier einige Regeln angepasst bzw. neu hinzugefügt werden. Das eigentliche Zeichnen der verschiedenen Objekte wird an die jeweiligen Renderer delegiert. Jeder Renderer übernimmt dabei das Zeichnen eines bestimmten Objektes. Alle Renderer erben die abstrakte Klasse *NGMapObjectRenderer*, welche einzelne abstrakte Methoden enthält. Bei jeder

Änderung der Kartendarstellung werden über den *NGMapRenderer* die einzelnen Renderer aufgerufen und so die gesamte Spieldarstellung aktualisiert.

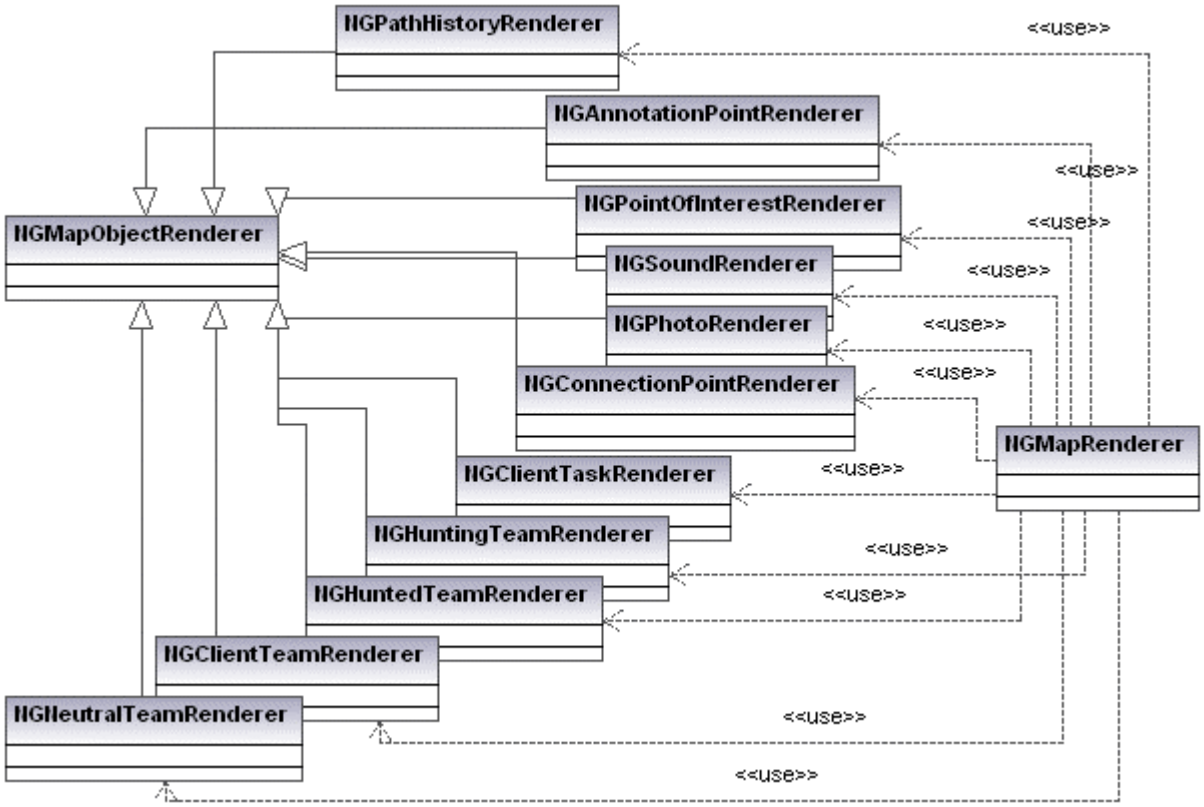

Abbildung 4. 8: Klassendiagramm bezüglich den Renderern

Für die Visualisierung des zurückgelegten Weges wurde die Klasse *NGPathHistoryRenderer* neu hinzugefügt. Dieser Renderer erbt wie alle anderen auch den *NGMapObjectRenderer* und zeichnet alle Objekte eines Weges auf, abhängig von der definierten Darstellungsart in der Konfigurationsdatei. Der *NGPathHistoryRenderer* enthält mehrere Methoden. Darstellungsmethoden und Hilfsmethoden. Jede Darstellungsmethode ist für eine Darstellungsart verantwortlich. Jede dieser Methoden erwartet als Parameter ein SWT-Display, einen Vektor gefüllt mit History-Daten, einen SWT-Grafikkontext (engl. Graphics Context) und eine Ganzzahl. Gewisse wenige Methoden benötigen zusätzlich ein Array statt einen Vektor als Parameter. Der *NGPathHistoryRenderer* kann sehr leicht um neue Methoden bzw. Darstellungsarten erweitert werden. Neben den Darstellungsmethoden enthält die Klasse noch einige Hilfsmethoden, die von den verschiedenen Darstellungsarten benutzt werden. Nun werden alle implementierten Darstellungsarten bzw. Darstellungsmethoden kurz erläutert und für jede Visualisierungsart einige Bilder aus dem mobileGame hinzugefügt. Zudem werden die Vor- und Nachteile jeder Darstellungsart angesprochen. Danach wird noch das neue Bedienelement bezüglich der History-Funktionalität gezeigt und die in der Konfigurationsdatei notwendigen Einträge pro Darstellungsart aufgelistet.

- **drawNormalLine(GC, int[], Display) Beispiel 1**

  Die vom mobileGame Server gelieferten Positionen werden ohne weitere Bearbeitung und Filterung gezeichnet und mit Linien verbunden. Die Positionen werden nicht speziell gezeichnet. Auch sonst werden keine weiteren Informationen dargestellt. Stockwerkwechsel werden nicht speziell dargestellt. Das Array welches die Positionen liefert enthält in diesem Falle auch identische Wegpunkte.

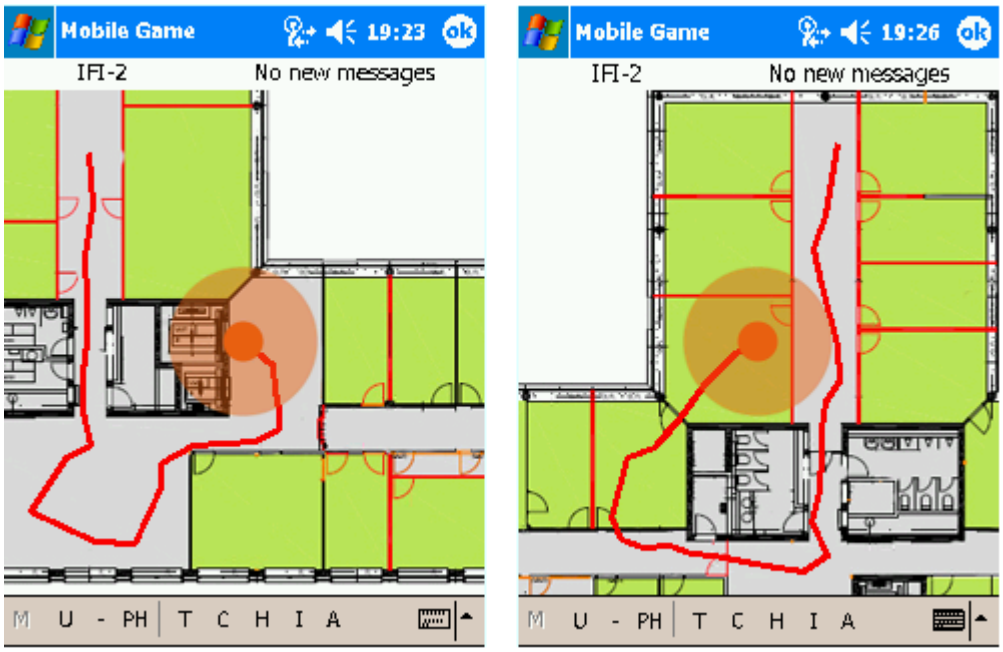

Abbildung 4. 9: Darstellung bei Berücksichtigung aller Punkte

Vorteile:

- Alle Wegpunkte werden berücksichtig. Im Prinzip die genaueste Wegaufzeichnung keine Punkte verloren gehen.
- Braucht am wenigsten Rechen- und Speicherressourcen.

Nachteile:

- Darstellung wird schon bei wenigen Überschneidungen sehr schnell unübersichtlich bzw. der genaue Wegverlauf kann nicht mehr erkannt werden.
- Nur Liniendarstellung, d.h. keine zusätzlichen Informationen. Stockwerkswechsel und Startpunkt werden nicht explizit angezeigt.

- **drawTimeSelected (GC,Vector, Display, int) Beispiel 2**

  Die in der Logikschicht gefilterten und bearbeiteten History-Daten werden zusätzlich nach einem Zeit-Kriterium ausgewählt. Der Methode wird unter anderem ein Zeit-Parameter übergeben. Dieser Parameter stellt eine minimale Zeitdifferenz dar, welche die ausgewählten Daten erfüllen müssen. Es werden also nur diejenigen aufeinander folgenden Wegpunkte in Betracht gezogen, deren Zeitabstände außerhalb dieser minimalen Zeitspanne liegen. Die selektierten Wegpunkte werden als Punkte dargestellt. Zusätzlich werden die Punkte nummeriert. Sinnvoll sind hier Zeitabstände zwischen 4 bis 20 Sekunden. Folgende Abbildungen zeigen eine solche Darstellung. Links werden die Punkte mit Linien verbunden, rechts werden nur die Punkte gezeichnet. Bei der Darstellung der Nummerierung wurde die Transparenz weggelassen.

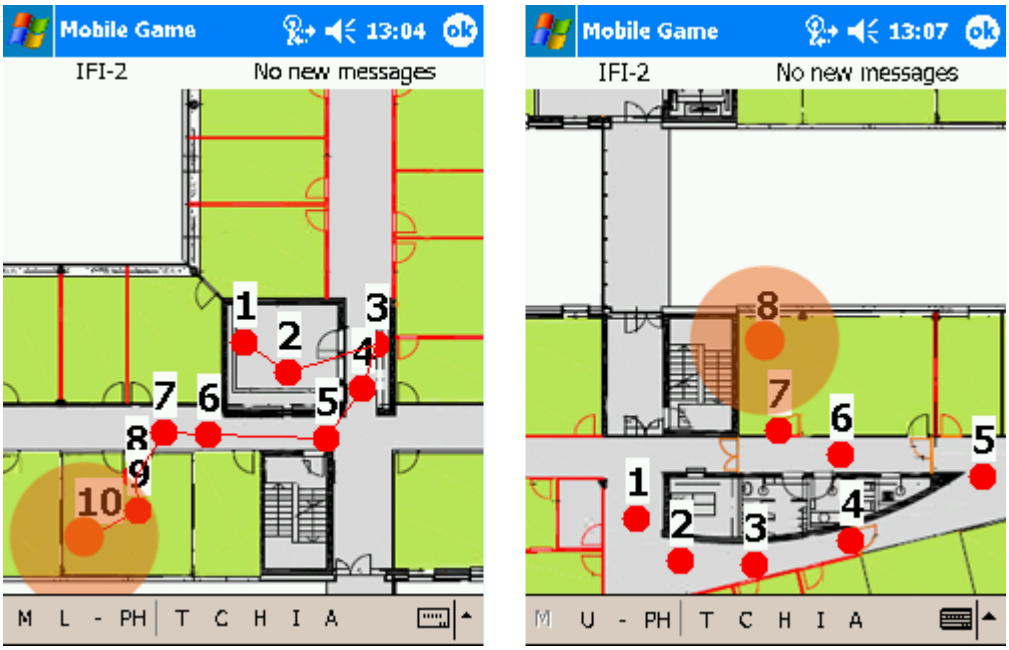

Abbildung 4. 10: Darstellung bezüglich dem Zeit-Kriterium

Vorteile:

- Die Darstellung bleibt bei passender Wahl der minimalen Zeit übersichtlich.
- Nummerierungen der Wegpunkte und die Möglichkeit spezielle Wegpunkte zu markieren erhöhen die Übersichtlichkeit und den Informationsgehalt. Startpunkt und Stockwerkwechsel werden ebenfalls angezeigt.

Nachteile:

- Bei kreisförmigen Bewegungen kann auch hier die Darstellung schnell unübersichtlich werden.
- Ist die minimale Zeitspanne zu lang gewählt, gehen viele Positionen verloren und die Wegdarstellung wird ungenau.
- Ist die minimale Zeitspanne zu klein gewählt, wird die Visualisierung sehr schnell unübersichtlich.

- **drawDistSelected (GC, Vector,Display, int) Beispiel 3**

  Diese Methode ist sehr ähnlich der *drawTimeSelected* Methode. Hier wird nach einem zusätzlichen Distanz-Kriterium ausgewählt. Aufeinander folgende Wegpunkte welche zueinander einen gegebenen minimalen Abstand haben werden berücksichtigt. Liegen aufeinander folgende Wegpunkte zu nahe zusammen, werden sie nicht dargestellt. Die Darstellungen selber erfolgt wie bei der vorherigen Methode mittels Punkten, Linien und Nummerierungen. Normale Wegpunkte werden als rote Punkte gezeichnet, Startpunkt als hellgrüner Kreis und die speziellen Wegpunkte, die manuell über ein Bedienelement gesetzt werden müssen, werden mit einem roten Rahmen umrahmt. Die blauen Punkte mit der Überschrift „Floor" repräsentieren einen Stockwerkwechsel. Die Distanz selber wird in Pixel angeben. Sinnvolle Distanzen liegen hier zwischen 30 und 60 Pixel, je nach Umgebung. In diesem Beispiel wurde eine Distanz von 50 Pixels genommen. Die Visualisierungen der verschiedenen Punkte können einfach verändert werden.

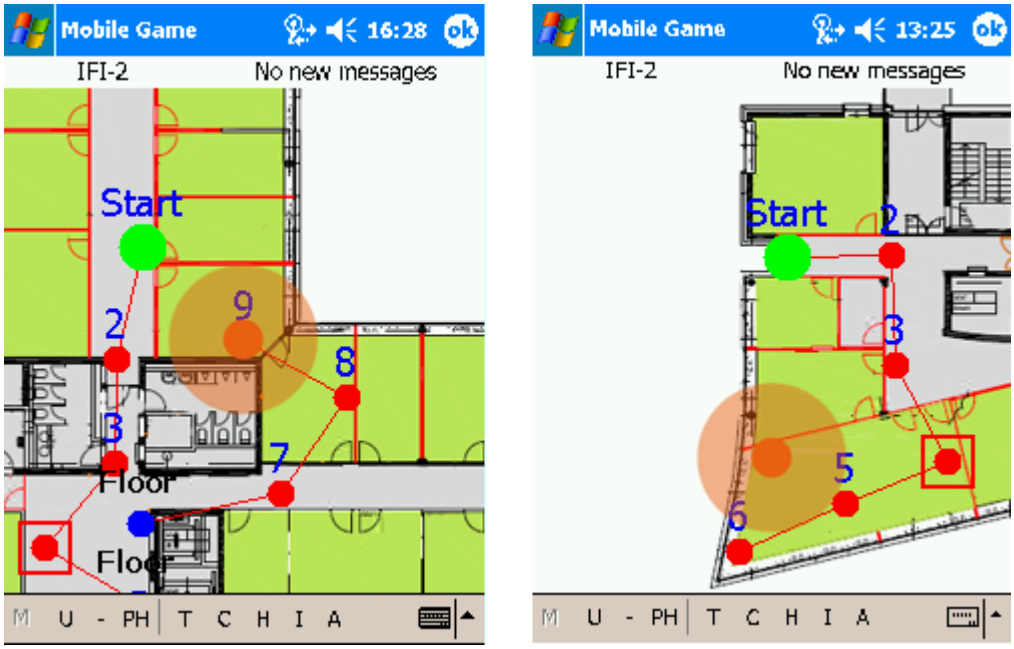

Abbildung 4. 11: Darstellung unter Berücksichtigung des Abstands-Kriterium

Vorteile:

- Die Darstellung enthält nützliche Zusatzinformationen wie Startpunkt und Stockwerkswechsel.
- Bei geschickter Wahl der Minimaldistanz wird die Übersichtlichkeit auch bei kreisförmigen Bewegungen gewährleistet.
- Die wohl effizienteste Darstellung.

Nachteile:

- Wird ein längerer Wegabschnitt mehrmals benutzt, so kann auch diese Darstellung unübersichtlich werden.

- **drawFadingColor(GC, Vector, Display, int) Beispiel 4**

  Diese Darstellungsart berücksichtigt das Abstands-Kriterium, zeichnet die Wegpunkte jedoch mit einem Farbverlauf. Nahe bzw. neue Wegpunkte werden mit einem niedrigen Transparenzwert versehen, ältere und weit weg liegende Wegpunkte verblassen mit zunehmendem Abstand und werden im Endeffekt nicht mehr dargestellt. Farbverlauf, Größe der Wegpunkte usw. können im Programmcode leicht verändert werden.

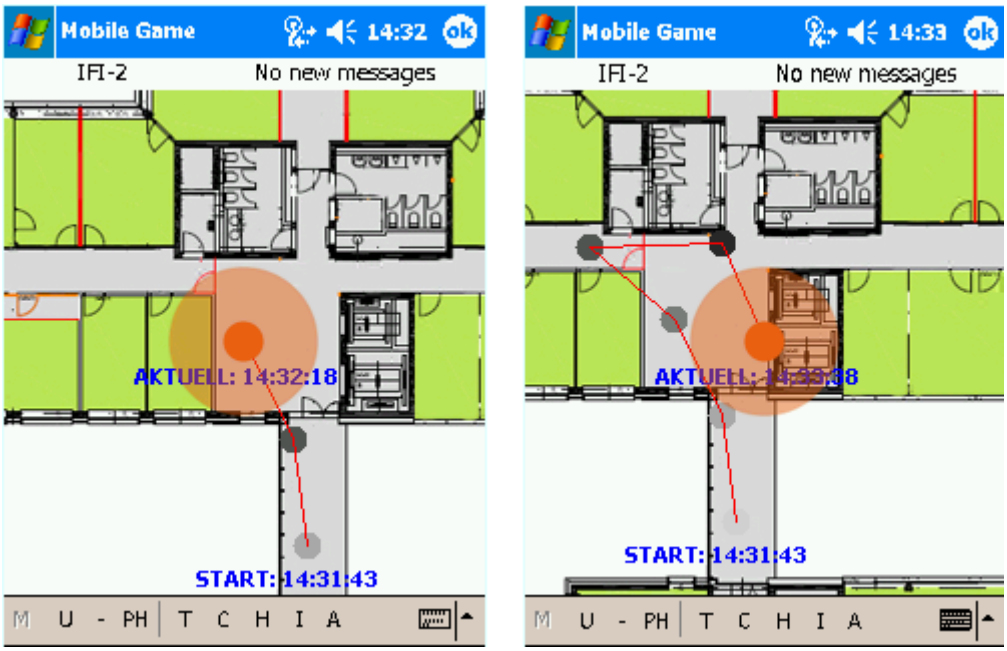

Abbildung 4. 12: Darstellung mit Farbverlauf und Zeitangaben

Vorteile:

- Die Darstellung bleibt auch bei kreisförmigen Bewegungen übersichtlich.
- Bei längeren Wegaufzeichnungen werden ab einen gewissen Punkt die alten Wegpunkte nicht mehr gezeichnet. Die Übersichtlichkeit bleibt so auch bei langen und sich überschneidenden Wegaufzeichnungen erhalten.

Nachteile:

- Bei konstant kreisförmigen Bewegungen kann die Darstellung auch unübersichtlich werden.

- **drawFadingSizeAndColor(GC, Vector, Display, int) Beispiel 5**

  Wie in der *drawFadingColor* Methode wird hier das Abstandskriterium benutzt und die Wegpunkte mit einem Farbverlauf dargestellt. Zusätzlich wurde hier noch ein Größenverlauf hinzugefügt. Desto älter bzw. weiter entfernt die Wegpunkte von der aktuellen Position des Spielers sind, desto kleiner sind sie dargestellt. Kombiniert mit dem Farbverlauf wird so eine ansprechende Visualisierungsart realisiert.

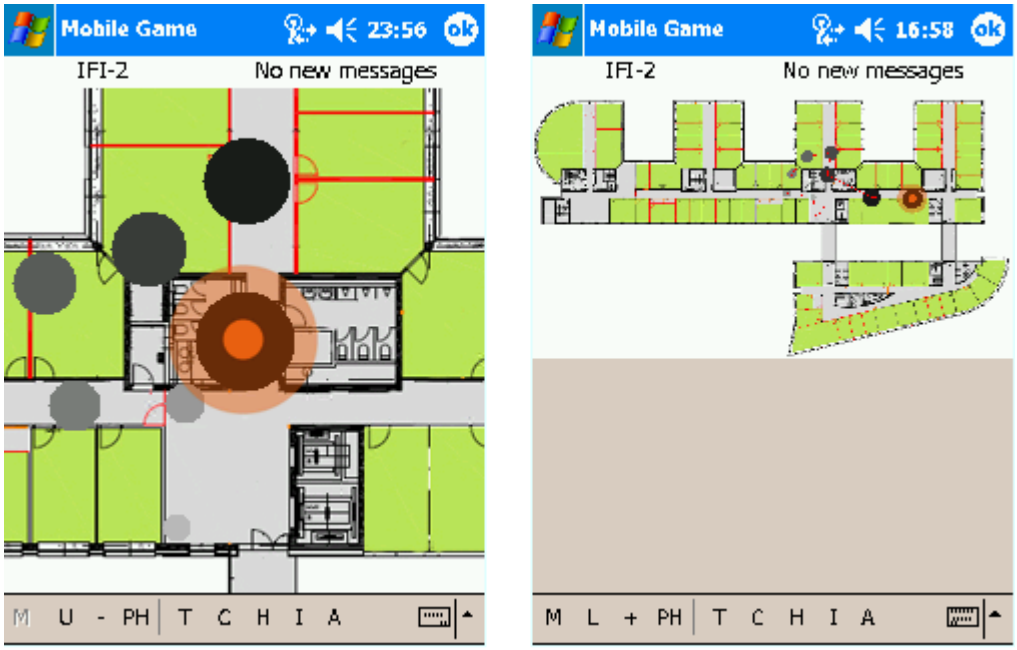

Abbildung 4. 13: Darstellung Größen- und Farbverlauf

Vorteile:

- Bei leicht kreisförmigen Bewegungen immer noch übersichtlich.
- Wie bei der vorherigen Methode werden bei längeren Wegaufzeichnungen die alten Wegpunkte nicht mehr gezeichnet. Zudem werden hier die Wegpunkte immer kleiner dargestellt, was die Übersichtlichkeit bei langen zusätzliche erhöht

Nachteile:

- Auch hier kann bei extremen Wegüberscheidungen die Übersichtlichkeit verloren gehen.
- Keine Nummerierungen der Wegpunkte.

- **drawFog (GC, Vector, Display, int)** **Beispiel 6**

  Diese Darstellung versucht den zurückgelegten Weg mit einer Art von "Nebel" zu zeigen. Neue Punkte sind Blau. Bei älteren Punkten nehmen die Rot, Grün und Blau- Farbenwerte mit zunehmender Distanz ab. Ab einem gewissen Punkt werden alte Wegpunkte nicht mehr berücksichtigt und werden nicht mehr dargestellt. Die Darstellungsfarbe kann leicht auf einen beliebig anderen Farbverlauf verändert werden. Die Länge der Nebeldarstellung kann ebenfalls je nach Kartengröße angepasst werden.

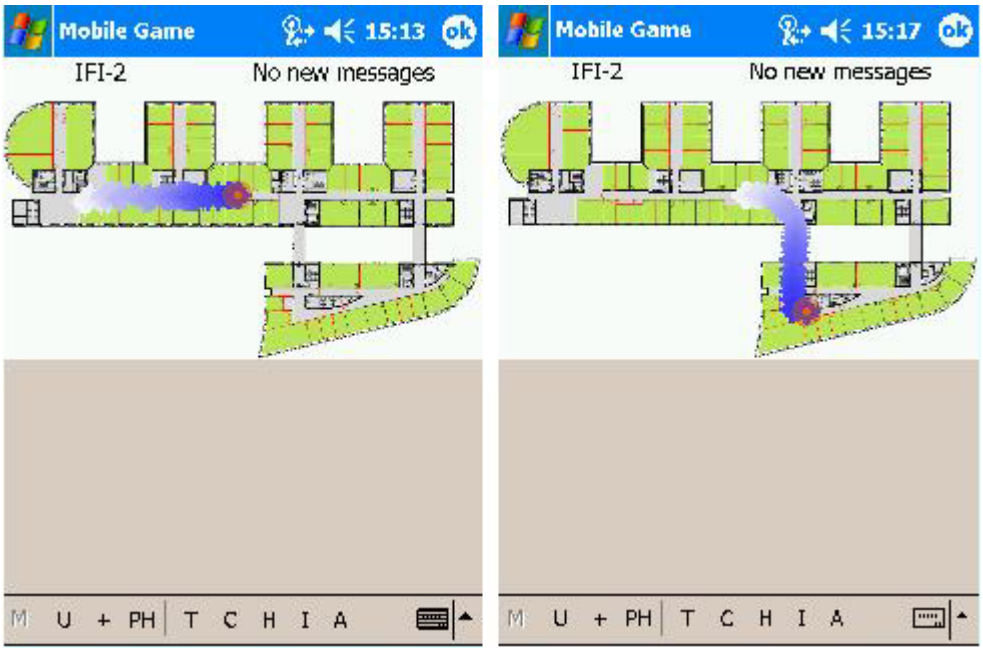

Abbildung 4. 14: Nebelartige Darstellung

Vorteile:

- Die Darstellung ist optisch die schönste.
- Bei kreisförmigen Bewegungen bleibt die Übersicht bis zu einem gewissen Grade erhalten.
- Verbleibt das sich bewegende Objekt länger an gewissen Positionen, so sind diese durch unregelmäßige Farbverläufe gut erkennbar.

Nachteile:

- Bei zeitlich sehr unregelmäßigen Positionsübermittlungen des Servers oder bei längeren Unterbrechungen wird die Darstellung zunehmend unbrauchbar.
- Die Darstellungsart benötigt relativ viel Rechenleistung und besetzt auch viel Speicher.
- Enthält wenig explizite Informationen (Stockwerkswecksel).

Die Benutzeroberfläche des mobileGames wurde minimal erweitert. Neu hinzugekommen ist ein „PH" Knopf. Bei Betätigung öffnet sich ein Untermenu. In diesem Menu kann über das Bedienelement „PH – ON/OFF" die Wegdarstellung aktiviert und wieder deaktiviert werden. Mit dem Bedienelement „Set Waypoint"

wird der letzte dargestellte Wegpunkt speziell markiert. Folgende Abbildung zeigt die erweiterte Benutzeroberfläche:

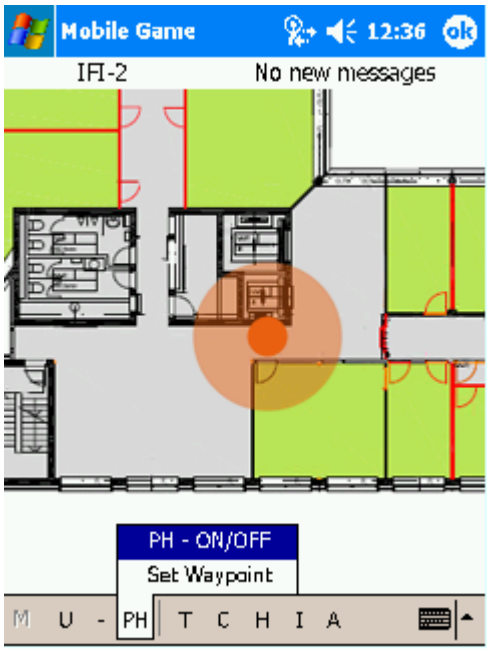

Abbildung 4. 15: Die erweiterte Benutzeroberfläche

In der Konfigurationsdatei wird festgehalten welche Darstellungsart gewählt wird. Folgende Einträge sind im Moment unter „ph_visualization" möglich:

- *Einfache Linie*: Zeichnet eine einfache Linie ohne Punkte (siehe Beispiel 1).
- *Linie 20px* (Statt 20 kann 30, 40, 50, 60 oder 80 stehen): Darstellung bezüglich dem Distanzkriterium. (siehe Beispiel 3).
- *Linie 4s* (Statt 4 kann 8, 12, 16, oder 20 stehen): Darstellung bezüglich dem Zeitkriterium. (siehe Beispiel 2).
- *Farbverlauf Kreise*: Zeichnet die Kreise mit einem Farbverlauf (siehe Beispiel 4).
- *Größe und Farbverlauf:* Zeichnet die Kreise mit einem Farb- und Größenverlauf. (siehe Beispiel 5).
- *Nebel:* Zeichnet die Nebeldarstellung (siehe Beispiel 6).
- *Nebel_Trans:* Zeichnet die Nebeldarstellung leicht anders (kein Beispiel).

4.4 Tests

Um die erfolgten Veränderungen und neue implementierten Funktionalitäten zu überprüfen, wurden während der ganzen Entwicklungsphase bis zum Schluss regelmäßige Tests durchgeführt. Auch wenn damit die Korrektheit eines Programms nicht bewiesen werden kann, so stellt das Testen einen effektiven Prozess zur Fehlerfindung dar. Durch sorgfältiges Testen des Programms steigt die Wahrscheinlichkeit, dass das Programm sich auch in den nicht getesteten Fällen wunschgemäß verhält [GLINZ_KVSE]. Grundsätzlich wurden zwei verschiedene Arten von Tests durchgeführt, Laufversuche und Benutzertests. Bei den Laufversuchen testete ich als Entwickler selber das Programm. Bei den Benutzertests testeten dagegen externe, nicht in die Entwicklung involvierte Personen die Software. Diese zwei Testarten widerspiegeln sich auch in den zwei folgenden Testverfahren, welche beim testen der History-Funktionalität und dem erweiterten mobileGame als Ganzes zum Einsatz kamen:

- **Black-Box-Verfahren**
  Beim so genannten Black-Box-Verfahren handelt es sich um einen funktionsorientierten Test. Die Auswahl der Testfälle erfolgt hier aufgrund der Spezifikation. Black-Box-Tests können ohne Kenntnisse der inneren Programmstruktur und der Funktionalitäten durchgeführt werden. Die Tester kennen zwar das erwartetet Ergebnis, wissen aber nicht wie das Programm zu diesem Ergebnis gekommen ist. Black-Box-Tests sind deswegen für größere Programmteile besser geeignet. Bei den durchgeführten Benutzertests handelte es sich um Black-Box-Tests. Auf die Benutzertests wird später in einem speziellen Abschnitt genauer eingegangen.

- **White-Box-Verfahren**
  Im Gegensatz zum Black-Box Test muss bei diesem Verfahren der Tester die Struktur des Programmcodes kennen. Der interne Aufbau des Testobjektes muss bekannt sein. White-Box-Tests werden deswegen auch als strukturorientierte Test bezeichnet. Für Entwickler sind White-Box-Tests bzw. strukturorientierte Tests sehr geeignet, da dieser logischerweise seine Programme am besten kennt. Der White-Box-Test wird auch sehr oft „codebasierter" Test genannt, da sich die Testfälle sehr stark auf einzelne kleinere Codefragmente konzentrieren. Die von mir durchgeführten Laufversuche widerspiegeln solche White-Box-Tests.

Die zwei aufgezeigten Testverfahren haben Vor- und Nachteile. Ein großer Vorteil von Black-Box-Tests ist die Tatsache, dass die Tests aus der Benutzerperspektive durchgeführt werden. Die Benutzer testen das Gesamtsystem unabhängig voneinander und von den Entwicklern. Ein nicht zu unterschätzender Nachteil ist, dass die durchgeführten Tests redundant und deswegen weniger effizient sein können. Zudem besteht die Gefahr, das gewisse Programmteile bzw. Funktionalitäten vergessen und so nicht überprüft werden. White-Box-Tests dagegen können sehr umfangreich und aufwendig werden. Desto größer das Programm ist, desto mehr potentielle strukturorientierte Tests sind notwendig. Auf der anderen Seite deckt dieses Testverfahren oft versteckte Fehler im Programmcode auf, welche dann sofort korrigiert werden können, so dass sie keinen negativen Einfluss auf zukünftige Programmteile haben.

In den nächsten zwei Abschnitten werden die durchgeführten Tests näher erklärt und die Vorgehensweisen beschrieben.

4.4.1 Laufversuch

Um die Funktionstüchtigkeit der erweiterten und neu implementierten Teile der Client-Software zu überprüfen, führte ich während der Entwicklungsphase regelmäßige Tests durch. Testgegenstand der Laufversuche waren anfangs einzelne Komponenten der Software. Diese Komponententests beschränkten sich auf kleine Teile des Programms wie Module, Methoden, Klassen oder auf einzelne Codefragmente. Im Laufe der Entwicklungszeit testete ich immer größere Teile der Software, was im Endeffekt zu einem kompletten Systemtest führte. Testen setzt voraus, dass die erwarteten Ergebnisse bekannt sind. Zu Beginn der Laufversuche testete ich ausschließlich gegen die Spezifikation, da zu diesem Zeitpunkt noch keine Testergebnisse vorhanden waren. Später konnte ich entsprechend einem Regressionstest zusätzlich gegen vorhandene Testergebnisse testen, da die meisten Komponententests mehrfach durchgeführt wurden. Bei den Komponententests war es anfangs nicht einfach, gegen die Spezifikation zu testen, da die Spezifikation sich primär auf das gesamte System bezieht und nicht auf einzelne Programmteile. So musste ich aus der Spezifikation Bedingungen und Vorraussetzungen ableiten, die sich explizit auf die entsprechenden Programmteile bezogen. Ich musste mich fragen, welche Bedingungen bzw. Voraussetzungen ein gewisser Teil der Software erfüllen muss, damit diese den spezifizierten Anforderungen des Gesamtsystems gerecht werden. Einer der wichtigsten Aufgaben beim Testen ist die Auswahl der Testfälle. Testfälle müssen vor allem repräsentativ, redundanzarm und fehlersensitiv sein. Mein Ziel war es, mit möglichst wenigen Testfällen möglichst viele Fehler zu finden. Folgend werden einige wenige Testfälle gezeigt, die ich neben anderen während der Entwicklungszeit benutzt habe. Jeder Testfall enthält die Testfallnummer, die erfolgte Eingabe, das erwartete Resultat und der Befund des Ergebnisses. Testfälle werden in Testabschnitten zusammengefasst. Zu jedem Testabschnitt werden Zweck, Hinweis, Vorbereitungs- und Aufräumarbeiten dokumentiert [GLINZ_KVSE].

Testabschnitt	Korrekte Eliminierung von Redundanzen
Zweck	Testet ob die DataFiltering Methode aufeinander folgenden identische Wepunkte korrekt filtert
Vorbereitungszeit	keine
Aufräumaufgaben	keine
Hinweis	- Die x und y Koordinaten müssen ganze Zahlen sein - Die Eingabe sind Folgen von x,y-Koordinaten
Testsequenz	Keine aufeinander folgende identische Wegpunkte

TestfallNr.	Eingabe	Erwartetes Resultat	Befund
1-1-1	[x,y] → [2,4], [3,1], [5,8]	[x,y]‘ → [2,4], [3,1], [5,8]	ok
1-1-2	[x,y] → [2,4], [2,4], [5,8]	[x,y]‘ → [2,4], [5,8]	ok
1-1-3	[x,y] → [2,4], [2,4], [2,4]	[x,y]‘ → [2,4]	ok
1-1-4	[x,y] → []	[x,y]‘ → []	ok
1-1-5	[x,y] → [2,4], [10,5], [3,6], [2,4]	[x,y]‘ → [2,4], [10,5], [3,6], [2,4]	ok

Tabelle 10: Testfall-Beispiel 1

Testabschnitt	Abstands-Kriterium wird korrekt berücksichtigt
Zweck	Testet ob Wegpunkte nicht berücksichtigt werden, welche das Kriterium des minimalen Abstand von 10 nicht erfüllen.
Vorbereitungszeit	keine
Aufräumaufgaben	keine
Hinweis	- Die x und y Koordinaten müssen ganze Zahlen sein - Die Eingabe sind Folgen von x,y-Koordinaten
Testsequenz	Minimaler Abstand zwischen zwei aufeinander folgenden Wegpunkte

TestfallNr.	Eingabe	Erwartetes Resultat	Befund
2-1-1	[x,y] → [10,15], [30,60], [15,8]	[x,y]‘ → [10,15], [15,8]	ok
2-1-2	[x,y] → [29,40], [22,42], [5,8]	[x,y]‘ → [29,40], [22,42]	ok
2-1-3	[x,y] → [50,55], [10,1], [100,120]	[x,y]‘ → [50,55]	ok
2-1-4	[x,y] → []	[x,y]‘ → []	ok
2-1-5	[x,y] → [2,4]	[x,y]‘ → [2,4]	ok

Tabelle 11: Testfall-Beispiel 2

Testabschnitt	Zeit-Kriterium wird korrekt berücksichtigt
Zweck	Testet ob die Wegpunkte korrekt nach dem Zeit-Kriterium selektioniert werden. Zeitdifferenz mindesten 5 Sekunden (als Beispiel).
Vorbereitungszeit	keine
Aufräumaufgaben	keine
Hinweis	- Jedes History-Data Objekt hat einen Zeitangabe - Die Eingaben sind Zeitstempel, m=Minute, s=Sekunde
Testsequenz	Minimale Zeitdifferenz zwischen zwei benachbarten Wegpunkten

TestfallNr.	Eingabe	Erwartetes Resultat	Befund
3-1-1	{m,s} → {34,24}, {34,27} {34,50}	{m,s}‘ → {34,24}, {34,50}	ok
3-1-2	{m,s} → {04,42}, {04,59}	{m,s}‘ → {04,42}, {04,59}	ok
3-1-3	{m,s} → {54,42}, {54,43},{54,46}	{m,s}‘ → {54,42}	ok
3-1-4	{m,s} → {21,42}, {21,50},{22,00}	{m,s}‘ → {21,42}, {21,50},{22,00}	ok
3-1-5	{m,s} → {}	{m,s}‘ → {}	ok

Tabelle 12: Testfall-Beispiel 3

Diese gezeigten Testfälle richteten ihren Fokus primär auf einzelne Codefragmente und Methoden der History-Funktionalität. Sie widerspiegeln nur einen kleinen Teil den von mir durchgeführten Tests. In regelmäßigen Abständen fanden auch Tests bzw. Versuche unter praxisnahen Bedingungen am Institut für Informatik statt. Dabei wurden größere Komponenten wie zum Beispiel eine neu entwickelte Darstellungsart getestet. Primär ging es darum zu sehen, ob die fundamentalen Aspekte einer Wegdarstellung unter reellen Bedingungen funktionieren. Für jede neue Darstellungsart prüfte ich ob die Wegpunkte korrekt angezeigt und nummeriert werden, ob die Stockwerke unterschieden werden, ob die Wegpunkte korrekt miteinander verbunden werden usw. Diese praxisnahen Tests erforderten einen hohen Zeitaufwand, da korrigierte oder neu entwickelte Programmteile immer sofort auf den PDA transferiert und darauf getestet werden mussten.

4.4.2 Benutzertest

Neben den Laufversuchen die ich während meiner Arbeit selber durchgeführt habe, fanden auch Benutzertests statt. Diese starteten nachdem ich den ersten Prototyp des erweiterten mobileGames fertig hatte. Beim Benutzertest handelt es sich um einen Black-Box-Test, daher, die Tester haben keine Kenntnisse der inneren Bauteile und der Programmstruktur. Durch die Informationen die ich ihnen anlässlich des Benutzertests gegeben habe, wussten sie nur wie das Ergebnis aussehen soll, aber nicht wie das Programm zu diesen Ergebnissen kommt. Benutzertests sind dazu da, die verschiedenen Funktionalitäten unter praxisnahen Bedingungen zu testen und zu evaluieren, aber auch die

Benutzerfreundlichkeit des Programms zu überprüfen. Ich wollte primär untersuchen, ob die Benutzer intuitiv richtig mit dem erweiterten System bzw. mit der History-Funktionalität umgehen können und auf welche Schwierigkeiten und Probleme sie dabei stoßen. Unter anderem sollte auch eine Beurteilung der verschiedenen Darstellungsarten einer Wegaufzeichnung stattfinden, um sich schlussendlich auf eine paar wenige Visualisierungsarten konzentrieren zu können.

4.4.2.1 Vorbereitungen und Probleme

Die Benutzertests fanden alle am Institut für Informatik statt. Da der Standort des Institutes erst vor kurzer Zeit geändert hat, war es notwendig, die Grundrisskarten des Gebäudes bezüglich der Ekahau Positioning Engine komplett neu zu kalibrieren. Die Kalibrierungen von zwei Stockwerken verlief erfolgreich, dennoch stellte ich fest, dass die Qualität bzw. die Abdeckung des WLAN's[41] im Gebäude noch nicht optimal war. Die Kalibrierung erfolgte mit den Ekahau Tags, da diese dafür speziell konzipiert wurden und somit sehr exakt und zuverlässig sind. Trotzdem kam es immer wieder vor, dass die Verbindung zwischen Ekahau Positioning Engine und dem Ekahau Tag unterbrochen wurde. Später musste ich auch feststellen, dass die Positionierung trotz sorgfältiger Kalibrierung der Grundrisskarten stark verzögert und nicht selten äußerst ungenau war. Gemäß meinem Betreuer Christoph Göth war dieses Problem bekannt. Der Grund dafür sind anscheinend die vielen Metalleinrichtungen des neuen Institutgebäudes. Das Problem mit der sehr ungenauen Positionierung beeinflusste die Qualität des Benutzertests massiv, da die History-Funktionalität bzw. die Wegdarstellung stark von korrekten und aktuellen Positionsinformationen abhängt. Auf der anderen Seite wurde bei der Spezifikation der Anforderungen festgehalten, dass die Wegdarstellung auch bei Störungen wie geringe Bandbreite, kurzzeitige Verbindungsabbrüche und bei Schwierigkeiten mit der eigentlichen Positionierung adäquat funktionieren sollte. Die Tests fanden trotz diesen Umständen statt. An den Benutzertests nahmen mehrere Personen teil. Für jede Person wurde ein PDA mit installiertem mobileGame und ein Ekahau Tag bereitgestellt.

4.4.2.2 Testszenario - Durchführung

Für die Benutzertests wurde ein Szenario erstellt. Da die Probleme mit der eigentlichen Positionierung das mobileGame generell negativ beeinflusste, entschied ich mich das Szenario einfach bzw. klein zu halten, damit sich die Tester primär auf die History-Funktionalität konzentrieren können und sich nicht mit allen anderen Funktionalitäten des mobileGames auseinander setzen müssen. Wie ich schon erwähnt habe, war das primäre Ziel die neuen Funktionalitäten unter praxisnahen Bedingungen zu testen und die verschiedenen Darstellungsarten zu beurteilen. Das Szenario sah folgendermaßen aus:

Jeder Tester bekam von mir neben dem PDA mit darauf laufendem mobileGame und einem Ekahau Tag jeweils eine Kopie der Stockwerkkarten des Institutes auf welchem ein Weg eingezeichnet war. Alle Tester hatten unterschiedliche Wege und Startpunkte. Die Ziele waren jedoch bei allen Personen gleich. Die Tester durften untereinander keine Informationen bezüglich ihres auf der Karte

[41] Wireless Locar Area Network

aufgezeichneten Wegs bekannt geben. Tester A wusste also nicht welchen Weg Tester B nehmen muss und umgekehrt. Die Tester hatten nun die Aufgabe, dem auf der Karte eingezeichneten Weg nach zu gehen und am Ziel zu warten, bis alle Beteiligten eingetroffen waren. Die eingezeichneten Wege führten über zwei Stockwerke und endeten alle im gleichen Aufenthaltsraum. Waren alle Teilnehmer am Ziel angekommen, tauschten diese ihre PDA's untereinander aus. Jeder hatte nun einen anderen PDA in der Hand als er zu Beginn der Tests bekommen hatte. Das nächste Ziel war nun, mit Hilfe der nun aktivierten Wegdarstellung des mobileGames über den exakt gleichen aber fremden Weg zurück zum Ausgangspunkt zu finden ohne die dazu gehörenden Grundrisskarten mit eingezeichnetem Weg zu benutzen. Die Tester hatten dabei die Aufgabe, die verschiedenen Darstellungsarten auszuprobieren und diese auch beim zurück laufen zu benutzen. Der Test endete als alle beteiligten Personen wieder beim Ausgangspunkt angekommen waren. Anschließend füllten sie noch einen von mir angefertigten Fragebogen aus. Der Fragebogen enthielt vor allem spezifische Fragen bezüglich der History-Funktionalität und generelle Fragen zum mobileGame als Ganzes.

4.4.2.3 Evaluation und Schlussfolgerungen

Die Benutzertests verliefen im Grossen und Ganzen erfolgreich. Die beteiligten Personen hatten viel Spaß und zeigten großes Interesse am mobileGame. Obwohl die Mehrheit keine Kenntnisse bezüglich der Handhabung eines PDA's hatte, konnten die Teilnehmer trotzdem die Tests mit den PDA's ohne größere Probleme durchführen. Dementsprechend gut war auch der Gesamteindruck über das System. Die Testpersonen lobten unter anderen die Bedienungsfreundlichkeit des gesamten mobileGames, die übersichtlichen Kartendarstellungen, die generelle Idee des mobileGames und auch die Darstellung des zurückgelegten Weges. Diese hat gut funktioniert und erfüllte ihren Zweck zufrieden stellend. Alle Personen konnten mit Hilfe der History-Funktionalität den zurückgelegten Weg einer anderen Testperson folgen und so über diesen zum Ausgangspunkt gelangen. Bezüglich der verschiedenen Darstellungsarten waren die Aussagen der Testpersonen interessant und auch ein bisschen überraschend. Fast alle waren der Ansicht, dass die Darstellungsart mit einfachen Punkten und Verbindungslinien, unter Berücksichtigung von minimalen Distanz- und Zeitabständen die übersichtlichste und daher effizienteste ist. Die ansatzweise implementierte nebelartige Darstellung dagegen wurde von den meisten Testpersonen als zu unübersichtlich und weniger geeignet beurteilt. Grund dafür sei die oft verzögerte Positionsübermittlung, welche die nebelartige Darstellung im Endeffekt unbrauchbar macht. Die Idee, dass der Benutzer die Darstellungsart verändert kann, stieß ebenfalls auf eine positive Resonanz. Einige Testpersonen waren der Ansicht, dass das System selber die Darstellungsart ändern soll, und zwar abhängig von der im Spiel aktuell gestellten Aufgabe oder der Eigenschaften der momentan dargestellten Übersichtskarte. Eine Darstellungsart sei nicht für alle Anwendungen, Situationen und Karten geeignet, und das Spiel soll die Darstellungsart dynamisch an die jeweiligen Umstände anpassen.

Zu den negativen Aspekten die von den Testpersonen genannt wurden waren hauptsächlich die langsame und ungenaue Positionsanzeige sowie der mit der Zeit auftretende Geschwindigkeitsverlust der PDA's.

Ich von meiner Seite war zufrieden mit den durchgeführten Tests. Die History-Funktionalität hat gut funktioniert, die verschiedenen Darstellungsarten wurden von den Testpersonen ausreichend untersucht und bewertet und alle beteiligten Personen, ich als Entwickler sowie die Testpersonen, zeigten sich am Schluss sehr zuversichtlich und beeindruckt.

4.5 Versuche und Experimente

Während meiner Arbeit habe ich viele Versuche durchgeführt. Viele davon waren kleine und einfache Versuche, die sich mit den Darstellungen einzelner Objekte der Wegaufzeichnung wie Wegpunkte, Linien, Startposition usw. befassten. Dabei probierte ich etliche Darstellungskombinationen aus. Verschiedene Kreisfarben und Kreisgrößen bei Wegpunkten, verschiedene Schriftarten und Schriftfarben für die Nummerierungen, transparente und nicht transparente Hintergründe bei Texten, verschiedene Linienformen, Liniendicken und Linienfarben bei den Verbindungslinien. Auch die Reihenfolge in der die verschiedenen Objekte auf die Übersichtskarte gezeichnet werden wurde verändert. Sollten die Wegpunkte die Verbindungslinien überdecken? Oder sollten die Linien sichtbar bleiben? Ist eine transparente Darstellung des „Nebels" besser als eine solide und verblassende Nebeldarstellung, wie sie im vorherigen Abschnitt zu sehen ist? Auch wenn diese Veränderungen in ihrer Form sehr einfach sind, so ist ihr Einfluss auf das Gesamtbild der Wegdarstellung nicht zu unterschätzen. Die Menge an verschiedenen Kombinationen ist sehr groß. So entschied ich mich einige Kombinationen die zum Beispiel auch bei den Benutzertests erfolgreich eingesetzt wurden zu behalten und noch ein paar wenige hinzuzufügen. Schlussendlich ist es auch eine Geschmackssache, welche Farben, Größen und Linienarten gewählt werden.

Neben diesen eher einfachen Versuchen, führte ich auch noch Versuche bezüglich verschiedener Teilfunktionalitäten durch. Diese waren den während der Entwicklungszeit durchgeführten praxisnahen Tests sehr ähnlich. Dabei ging es darum, ungewöhnliche Situationen zu simulieren und die Reaktion der neu implementierten Funktionalität zu beobachten. Mehrere nacheinander folgende Stockwerkwechsel, schnelle Bewegungen, langes verharren an einem bestimmten Ort usw.

Bezüglich der Wahl der Minimaldistanzen führte ich einige Experimente am Institut durch. Ich wollte herausfinden welche Distanzen für die Darstellung geeignet sind und welche nicht. Die bisherigen Beobachtungen am Institut zeigten das Distanzen zwischen 30 und 60 Pixel optimal sind. Die Eignung einer Distanz widerspiegelt sich allgemein in der Übersichtlichkeit und der Genauigkeit der erzeugten Wegdarstellung. Ganz wichtig ist die Tatsache, dass die Eignung einer Darstellung sich nur auf die entsprechende Umgebung bezieht. Eine Minimaldistanz die am Institut für Informatik den Weg übersichtlich und genau darstellt, kann in einer anderen Umgebung sehr ungeeignet sein. Mein Augenmerk richtete sich demnach primär auf den Aspekt der Übersichtlichkeit und der Genauigkeit auf den Karten des Instituts. Um diese beiden Eigenschaften objektiv beurteilen oder sogar messen zu können, musste ich passende Kriterien definieren. Ich versuchte also Kriterien und Faktoren zu finden, mit welchen ich die Übersichtlichkeit und Genauigkeit bewerten konnte. Dieses Unterfangen war nicht einfach. Ich musste mich selber fragen was übersichtlich ist, und was nicht, wie entsprechende Eigenschaften zu erkennen sind und was eine Genauigkeit

ausmacht. Schlussendlich legte ich mich auf folgende Eigenschaften bzw. Faktoren fest von denen ich der Meinung war, dass sie für die Beurteilung der Übersichtlichkeit und der Genauigkeit relevant sind. Die Häufigkeit dieser Eigenschaften in einer Wegdarstellung bestimmt maßgebend die Übersichtlichkeit und Genauigkeit der Aufzeichnung. Diese Eigenschaften sind aber nur bei den Darstellungsarten ersichtlich, die das Abstands- und Zeitkriterium berücksichtigen, da diese alle betroffenen Objekte standardmäßig zeichnen. Folgende Eigenschaften sind gemeint:

- **Bezüglich Übersichtlichkeit:**
  Die Übersichtlichkeit nimmt ab mit zunehmender Anzahl an Überschneidungen von Textfeldern, Wegpunktkreisen und Linien untereinander. Wegpunkte die sich gegenseitig tangieren, Texte welche andere Texte teilweise überdecken, Linien die andere Linien schneiden und Linien die durch Texte laufen verringern die Übersichtlichkeit.

- **Bezüglich Genauigkeit:**
  Die Genauigkeit der Wegdarstellung ist höher desto weniger Überschneidungen von Verbindungslinien zwischen Wegpunkten mit Wand bzw. Mauerdarstellungen der Übersichtskarte vorhanden sind. Verläuft zum Beispiel eine Verbindungslinie durch mehrere Räume ohne die Türen zu beachten, so gilt sie als ungenau.

Um nun die verschiedenen Minimaldistanzen zu untersuchen brauchte ich viele Wegaufzeichnungen. Dazu lief ich mit den gewählten Minimaldistanzen mehrere festgelegte Wege ab. Dies wiederholte ich einige Male und wertete nach jedem Durchgang die vom PDA aufgezeichnete und dargestellte History nach den oben genannten Kriterien aus. Da wie oben erwähnt, abhängig von der gewählten Darstellungsart, unterschiedlich viele Objekte dargestellt werden, musste ich mich bei den Versuchen auf eine Darstellungsart beschränken. Die von mir gewählten Minimaldistanzen lagen zwischen 10 und 200 Pixel. Das Resultat der Versuche war wenig überraschend. Wie schon die vergangenen Beobachtungen verdeutlicht hatten, liegen die idealen Distanzen zwischen 30 und 60 Pixel. Bei Distanzen unter 30 Pixel geht die Übersichtlichkeit sehr schnell verloren, da die Nummerierungen und Beschriftungen der Wegpunkte sich gegenseitig stark überdecken. Bei Distanzen über 60 Pixel wird die Darstellung zu ungenau. Da die Laufbereiche am Institut eher eng sind, verliefen die Verbindungslinien häufig über mehrere Wände und Räume.

Ein weiterer Versuch den ich geplant hatte war die Entwicklung und Erprobung eines Verfahrens, welches Wegpunkte, die zusammen in einem gewissen Bereich fester Größe sind, nicht als einzelne Punkte darstellt, sondern als eine Art von Fläche. Dieses Verfahren sollte bei vielen Wegüberschneidungen, die in gewissen Fällen auch unter Berücksichtigung von minimalen Distanzen auftreten können, die Übersichtlichkeit der Wegdarstellung gewährleisten. Aus zeitlichen Gründen konnten die Versuche jedoch nicht durchgeführt werden.

5 Fazit

In diesem Abschnitt erfolgt eine abschließende Betrachtung meiner Diplomarbeit. Das Erreichte wird analysiert und schlussendlich ein persönliches Fazit gezogen. Darin wird unter anderem beschrieben, auf welche Schwierigkeiten ich gestoßen bin und welche positiven Umstände mich bei meiner Arbeit unterstützt haben.

5.1 Das erweiterte mobileGame

Mit Hilfe des mobileGames sollen neue Studenten die Möglichkeit haben, die weitläufige Universität auf spielerische Weise kennen zu lernen. Die Integration in das universitäre Leben erfolgt schneller und problemloser. Der Übergang in einen neuen Lebensabschnitt unkomplizierter. Das mobileGame, so wie es vor meiner Arbeit war, bot den Benutzern viele nützliche und interessante Funktionalitäten an. Die Studenten konnten alleine oder in Gruppen diverse interessante Aufgaben lösen, neue unbekannte Orte finden, sich gegenseitig folgen und jagen, und sogar miteinander über Chat oder Skype kommunizieren. Der Anwendungsbereich des mobileGames ist äußerst groß und wird immer größer. Nicht zu letzt auch wegen den regelmäßigen Verbesserungen und Erweiterungen am System. Die von mir implementierte History-Funktionalität widerspiegelt so eine Erweiterung. Eine Erweiterung welche die Orientierung in großen und unübersichtlichen Umgebungen stark verbessert. Die History-Funktionalität erhöht in einem gewissen Masse den Komfort des mobileGames. Die Benutzer müssen sich nicht mehr selber um den Rückweg kümmern, sondern haben bequem die zusätzliche Option, sich ihren zurückgelegten Weg anzeigen zu lassen. Durch einen einfachen Knopfdruck erscheint die Wegdarstellung auf der Übersichtskarte des PDA's. Linien, Kreise, Zahlen und Muster in verschiedenen Farben und Formen stellen die History-Daten adäquat dar. Zudem habe ich versucht, diese zusätzliche Option so unauffallend wie auch nutzbringend wie möglich zu gestalten. Die am Anfang meiner Diplomarbeit festgelegten Muss- und Sollanforderungen wurden realisiert. Die History-Funktionalität wurde so implementiert, dass die strukturelle Form des mobileGames beibehalten wurde. Dieser Aspekt stand während meiner Arbeit immer im Vordergrund. Die Objektorientierung, die Einteilung in Schichten, die quasi Modularisierung der Komponenten, alle diese Eigenschaften wurden in die Entwicklung miteinbezogen. Die Erweiterbarkeit des mobileGames wurde ebenfalls beibehalten. Die neu eingefügten Klassen und Methoden wurden sauber programmiert und dokumentiert.

5.2 Ausblick bzw. Erweiterbarkeit

Die mit Hilfe meines Betreuers Christoph Göth festgehaltenen Muss- und Sollanforderungen bezüglich der History-Funktionalität wurden berücksichtigt und im System realisiert. Dennoch bietet diese neu entwickelte Komponente wie auch das mobileGame als Ganzes eine Menge an Erweiterungsmöglichkeiten. Folgende potentielle Weiterentwicklungsmöglichkeiten bezüglich der History-Funktionalität bieten sich für zukünftige Arbeiten an:

- Implementierung von weiteren Darstellungsarten. Denkbar ist auch die Benutzung von anderen komplexeren Bibliotheken zur Grafikprogrammierung wie zum Beispiel OpenGL[42] oder DirectX[43] Ressourcen. Zu beachten sind neben der Kompatibilität mit der J9-Runtime die dazu nötigen Systemressourcen.

- Entwicklung eines dynamischen und intelligenten Darstellungsmechanismus. Der Client wählt die Darstellungsart in Abhängigkeit gewisser festgelegten Kriterien selber aus und ändert diese dynamisch bei Veränderungen dieser Kriterien. Eine intelligente und selbständige Visualisierung welche von sich aus eine optimale Darstellungsart zu finden versucht.

- Eine Analysefunktionalität der Daten einer Wegaufzeichnung. Die gespeicherten History-Daten werden ausgewertet und die Ergebnisse mittels entsprechenden Grafiken und Tabellen explizit dargestellt. Die effektive Länge des Weges, die benötigte Zeit usw. können durch verschiedene Darstellungsarten visualisiert werden.

- Eine Funktionalität welche die Möglichkeit bietet, während eines Spieles die Eigenschaften der einzelnen Dargestellten Komponenten zu verändern. Unter anderem Linienfarbe, Liniendicke, Wegpunktsymbol, Wegpunktfarbe, Farbverlauf usw. Der Benutzer hat die Möglichkeit, die Darstellungen individuell an seine eigenen Ansprüche anzupassen.

- Implementierung einer persistenten Speicherungsfunktion. Die Wegaufzeichnung wird in einer separaten Datei gespeichert. Diese Datei wird in einem Verzeichnis im PDA abgelegt und kann bei Bedarf mittels einer speziell dafür entwickelten Funktionalität geöffnet und mit einer speziellen Visualisierung angezeigt werden.

- Bei Spielen mit mehreren Gruppen kann es unter Umständen von Vorteil sein, die zurückgelegten Wege allen anderen oder zumindest gewissen Gruppen zu übermitteln. Diese können dann den von anderen Gruppen zurückgelegten Weg auf ihrem PDA anschauen.

- Implementierung einer Funktionalität die es ermöglicht, bei Wegpunkten verschiedene Informationen bzw. Annotationen zu speichern. Wie beim mExplore [GOETH] können die Benutzer Texte, Klänge und sogar kurze Filme an gewissen interessanten Wegpunkten speichern, wodurch die Orientierung durch verbessert wird.

5.3 Persönliches Fazit

Bei der Wahl meiner Diplomarbeit hab ich mir sehr viel Zeit genommen. Dementsprechend lange dauerte es auch bis ich mich endgültig für eine Arbeit entschieden habe. Mir war es vor allem wichtig, dass die Arbeit einen wesentlichen technischen bzw. praktischen Teil beinhaltet. Da ich mich generell für die Entwicklung und Implementierung von Software interessiere, suchte ich

[42] http://opengl.j3d.org/swt/
[43] http://msdn.microsoft.com/directx/

eine Diplomarbeit die ihren Fokus zumindest zur Hälfte auf eine praktische Umsetzung legt. Die Arbeit die mir Christoph Göth angeboten hat und für dich ich mich schlussendlich entschieden habe, erfüllte diese Wünsche und Ansprüche sehr. Bei der Wahl dieser Diplomarbeit war ich mir bewusst, dass ich mich teilweise auf bekannten aber auch auf unbekanntem Terrain bewegen werde. So stellte zum Beispiel die Entwicklung von Java Applikationen für den Pocket PC eine komplett neue Herausforderung dar. Dagegen konnte ich beim Entwurf und Testen der Software, sowie der eigentlichen Java Programmierung auf durchwegs fundierte Erfahrungswerte zurückgreifen. Diese Mischung aus bekannten Aspekten und neuen Herausforderungen führte schlussendlich zu einer sehr interessanten Konstellation von abwechslungsreichen Aufgaben.

Während meiner Arbeit bin ich auf verschiedenartige Probleme gestoßen. Einige davon waren zu erwarten andere weniger. Das wohl größte Problem auf das ich gestoßen war bezog sich auf die eigentliche erstmalige Installation des kompletten mobileGames. Dieses Problem hatte in dieser Form nicht erwartet. Vor allem hätte ich mir nie gedacht, wie problemreich und tückisch eine solche Installation sein könnte. So benötigte ich unerwartet viel Zeit bis ich den Server, den Client und die Ekahau Engine erfolgreich zum Laufen bringen konnte. Dieser Umstand war vor allem anfangs sehr nervenaufreibend, da ich ohne ein lauffähiges mobileGame nicht mit der eigentlichen Entwicklungsarbeit beginnen konnte. Mit der Hilfe von Christoph Göth, den verschiedenen Erkenntnissen die ich im Laufe der Zeit erarbeitet habe und durch unzählige Versuche meinerseits konnten die Probleme bezüglich der Erstinstallation schlussendlich gelöst werden. Die eigentliche Entwicklung der Software verlief trotz verschiedenen unerwarteten Gegebenheiten im Grossen und Ganzen problemlos. Einige Schwierigkeiten traten während den ersten praxisnahen Tests am Institut für Informatik auf. Diese konnten jedoch in adäquater Zeit behoben werden. Bei der Entwicklung konnte ich auf viele Erfahrungswerte und Kenntnisse, die ich während meiner Studienzeit erlangt habe, zurückgreifen. So benutzte ich zum Beispiel beim Entwurf, bei der Implementierung und beim Testen der Software viele bekannte Verfahren des Software Engineering.

Im Allgemeinen bin ich sehr zufrieden mit der ausgewählten Diplomarbeit, deren Ablauf und dem erreichten Ergebnis. Das mobileGame beinhaltet nun eine zusätzliche History-Funktionalität, die den Benutzer bei seiner Orientierung unterstützt und den Komfort der mobileGames generell erhöht. Zum Schluss möchte ich mich bei meinem Betreuer Christoph Göth für seine Unterstützung beim Schreiben dieser Arbeit sowie bei der Entwicklung der neuen Funktionalität danken.

Anhang

A Inhalt der CD-Rom

Die der Diplomarbeit beiliegende CD-Rom enthält folgende Dateien und Verzeichnisse:

- **Ordner Arbeit:**
  In diesem Ordner befindet sich die schriftliche Fassung der Diplomarbeit im pdf-Format. Die Datei heißt *Diplomarbeit_AB.pdf*

- **Ordner mobileGame Client**
  Hier sind alle Dateien vorhanden um den mobileGame Client auf dem PDA zu installieren und zum Laufen zu bringen. Der Ordner enthält folgende Unterverzeichnisse:

  - **Unterverzeichnis Quellcode**
    Hier befindet sich der ganze mobileGame Client Quellcode, die benötigten Grafiken, die Konfigurationsdatei und sonstige Dateien sind hier ebenfalls vorhanden.

  - **Unterverzeichnis zusätzliche Software**
    Dieses Unterverzeichnis enthält die notwendigen Dateien, welche vorgängig auf dem PDA installiert werden müssen.

- Die Dateien *Zusfsg.pdf* und *Abstract.pdf*

B Verwendete Software

Bei der Entwicklung der History-Funktionalität und für das mobileGame allgemein wurde folgende Software eingesetzt:

- **Entwicklungsumgebungen**
  - Eclipse 3.1, http://www.eclipse.org
  - WebSphere Studio Device Developer, http://www-306.ibm.com/software/wireless/wsdd/

- **Java-Ressourcen**
  - Standart Widget Tool (SWT), http://www.eclipse.org/swt

- **Java-Runtimes**:
  - Java JDK 5.1, http://java.sun.com/javase/downloads/index.jsp
  - IBM WEME 5.7.2, http://www-306.ibm.com/software/ wireless/weme

- **Sonstige Tools**
  - Creole, http://www.thechiselgroup.org/?q=creole/
  - Altova UModel, http://www.altova.com
  - HauteCapture,http://www.vvse.com/cur/de/products/HauteCapture/index.html

C Beispiel der erweiterten Konfigurationsdatei

Beim starten des Clients werden die verschiedenen Informationen der Konfigurationsdatei eingelesen und verarbeitet. Diese Datei enthält die IP-Adresse und Portnummer des Servers, Teamname und Teampasswort, Karten, Icon und Audioverzeichnisangabe, den Ekahau Tagname, Skype-Server und Port, und neu auch die Darstellungsart der Wegaufzeichnung. Folgend wird eine Beispiel-Konfiguration gezeigt:

```
#Client Configuration File
#@Author Ante Bujas
team_name=Team1
team_pass=t1
server_ip=130.60.157.142
server_port=3222
map_path=maps\\
icon_path=icons\\
audio_path=audio\\
skype_name=unizh_game_1
skype_port=23233
tag_name=Tag-10
ph_visualization=Linie 30px
```

D Installationsanleitung des Clients

Die Installation des mobileGames bringt gewisse Schwierigkeiten mit sich. Um das mobileGame starten zu können, müssen Client und Server korrekt installiert und konfiguriert sein. Ich möchte hier primär auf die Einrichtung des mobileGame Clients eingehen und hoffe, dass dadurch die Installation des Clients bei zukünftigen Arbeiten einfacher wird. Als Ausgangslage nehme ich dieselbe Situation wie ich sie am Anfang meiner Arbeit vorgefunden habe.

Am Anfang meiner Arbeit hatte ich zwei CD-Roms zur Verfügung. Eine CD enthielt alle Ressourcen die zur Installation des mobileGames nötig sind. Auf der zweiten CD befand sich der Web Device Developer. Als erstes sollte der Device Developer installiert werden. Man startet die *launchpad.exe* Datei und gelangt so zum Installationsmenu. Dort wählt man *Websphere Studio Device Developer installieren*. Nachdem dieser erfolgreich installiert wurde, importiert man über den Device Developer den mobileGame Client, der sich auf der anderen CD befindet. *Datei → Importieren → Vorhandenes Projekt in den Arbeitsbereich* usw. Bei der Installation des Device Developers ist es wichtig das der PDA angeschlossen ist und somit das ActiveSync mit installiert wird. Wurde der Client erfolgreich importiert muss nun die Ausführung des Programms eingerichtet werden. *Ausführen → Ausführen*. Links im Konfigurationsfenster wählt man *Java on Device* mit einem Rechtsklick an und wählt *neu.* Dort sollte dann ein Eintrag *Neue Konfiguration* erscheinen. Nun muss man die Konfiguration entsprechend einrichten. Gewisse Einträge der Konfiguration können je nach System und Verzeichnispfaden unterschiedlich sein. Die Konfiguration ist in vielen Fällen sehr heimtückisch, deswegen als Beispiel meine lauffähige Konfigurationen in Bildern:

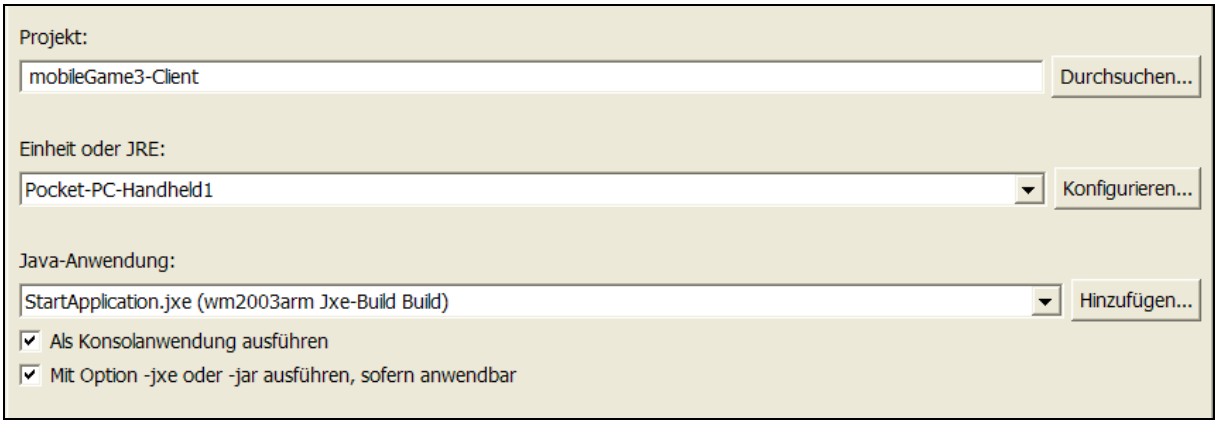

Abbildung C. 1: Konfiguration 1

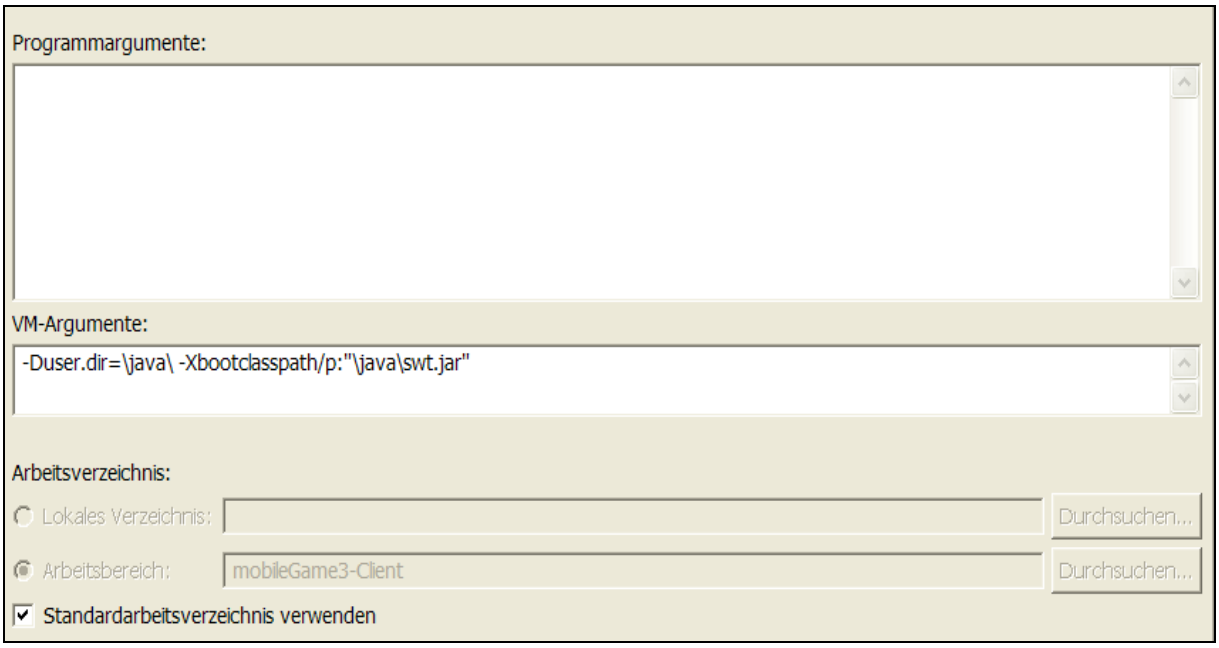

Abbildung C. 2: Konfiguration 2

Abbildung C. 3: Konfiguration 3

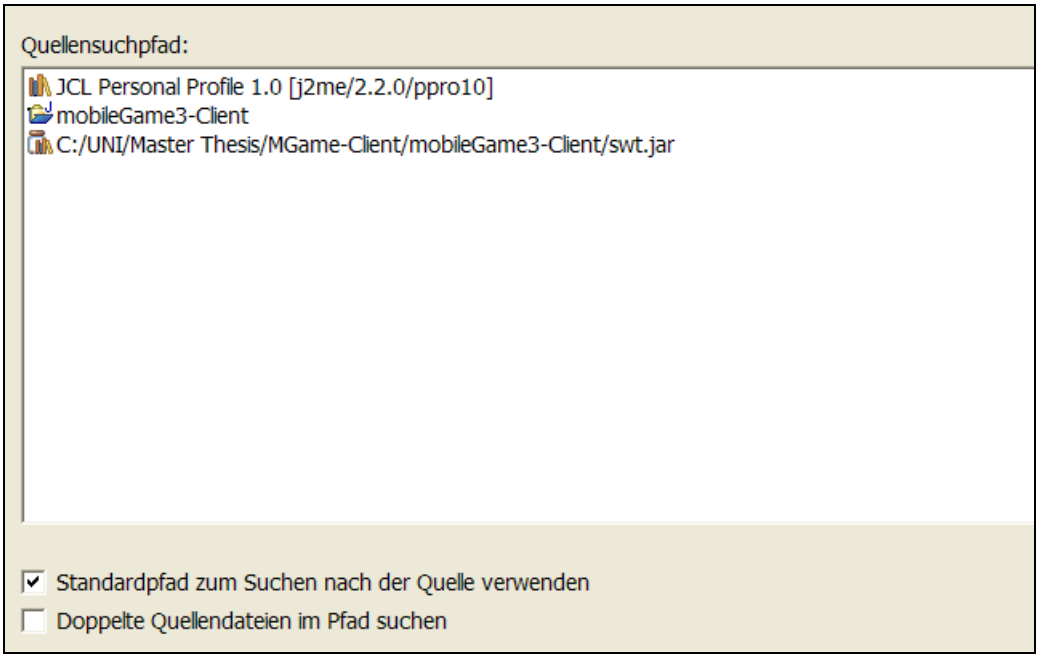

Abbildung C. 4: Konfiguration 4

Um zu überprüfen ob die Konfiguration in Ordnung ist muss zuerst auf dem PDA die nötige J9-Runtime installiert werden. Ist der PDA angeschlossen so startet man wieder das Websphere Device Developer Installationsmenu. Dort wählt man nun *Websphere Everyplace Micro Environment Personal Profile 1.0 für Windows Mobile installieren.* Somit wird auf dem PDA ein Java Ordner eingerichtet mit der J9-Runtime. Ist die Konfiguration im Device Developer in Ordnung und die J9-Runtime auf dem PDA vorhanden kann man nun versuchen über den Device Developer den Client zu starten. Bei jedem ausführen des Clients werden automatisch die Java Dateien in ein PDA-Kompatibles Format umgewandelt und auf den PDA transferiert. Wichtig ist hier zu wissen, das nur Java Dateien übermittelt werden und nicht die Konfigurationsdatei, der Image-, Icon- und Mapordner. Diese müssen nun manuell auf den PDA transferiert werden. Auf dem Arbeitplatz sollte bei korrekter Installation des Websphere Device Developers ein Ordner Namens *Mobiles Gerät* vorhanden sein. Über diesen Ordner kann man nun einfach die Konfigurationsdatei, der Image-, Icon- und Mapordner auf den PDA übertragen. Ist die Konfiguration im Device Developer in Ordnung und sind alle nötigen Dateien auf dem PDA, so sollte nun der Client funktionstüchtig sein. Um das ganze mobileGame zu starten, muss jedoch noch der Server mit der Ekahau-Engine korrekt installiert sein. Auch wenn diese Anleitung sicher nicht ganz vollständig ist, so hoffe ich, dass sie trotzdem eine Hilfe darstellt.

E Abbildungsverzeichnis

F Tabellenverzeichnis

G Abkürzungsverzeichnis

AOA	Angel of Arrival
DOI	Degree of Interest
GSM	Global System for Mobile Communication
GC	Graphic Context
GPS	General Positioning System
JPG	Joint Photographic Experts Group
SWT	Standard Widget Toolkit
TDOA	Time Difference of Arrival
TOA,	Time of Arrival
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network
IP	Internet Protocol
PDA	Personal Digital Assistant
POI	Point of Interest

H Bibliographie

H1 Literatur- und Publikationen

[BAUER] Manfred Bauer, *Vermessung und Ortung mit Satelliten*, Wichmann, 5.Auflage

[BOEHM] Winfried Boehm, *Handbuch der Navigation*, 3.Auflage, Busse-Seewald Verlag, 1994

[BURKHARDT] Sascha Burkhardt, *Aufwind im Wohnzimmer*

[BUJA] Andreas Buja et. all.; *Interactive Data Visualization using Focussing and Linking*; Proceedings of IEEE Visualization 1991, pp. 156 – 163

[DORNBUSCH] Peter Dorbusch und Max Zündt, *Realisierung von Positionsortungen in WLAN*, TU München, München, Deutschland

[EARNSHAW] Rae Earnshaw, *Visualization and Modelling*, Academic Press, Kalifornien, USA, 1997

[ENTREMONT] Tricia d'Entrement und Margaret-Anne Storey, *Using a Degree-of-Interest Model for Adaptive Visualizations in Protege*, University of Victoria, Victoria, Canada

[FAIRCHILD] Kim Michael Fairchild, *Information Management Using Virtual Reality-Based Visualization,* Virtual Reality: Applications and Explorations, Academic Press, 1993

[FURNAS] George W. Furnas, *Generalized Fisheye Views,* Bell Communications Research, St.Morristown, New Jersey, 1986

[GOETH] C. Göth, C.Lueg, N.Bidwell, *Learning From Insects? Towards Supporting Reflective Exploration of Unfamiliar Areas of Interest,*USA, 2005

[GIBSON] Jerry Gibson, *The mobile Communication Handbook*, 2.Auflage, CRC Press, USA, 1996

[GLINZ_SE] Martin Glinz, *Software Engineering*, Vorlesungsskript, Institut für Informatik, Universität Zürich, 2005

[GLINZ_SP] Martin Glinz, *Spezifikation und Entwurf von Software*, Vorlesungsskript, Institut für Informatik, Universität Zürich, 2005

[GLINZ_KVSE] Martin Glinz, *Kernvorlesung Software Engineering*, Vorlesungsskript, Institut für Informatik, Universität Zürich, 2005

[KNOPP] Helmut Knopp, *Astronomische Navigation*, Busse-Seewald Verlag, 1995

[HABERLAH] David Haberlah, *Aufbau und exemplarische Nutzung eines Satellitendatengestützten Navigations-Informations-Systems*, Freie Universität Berlin, Diplomarbeit

[PASCKE] Uwe Paschke, *Welt Geschichte*, Karl Müller Verlag, 1994

[PIROLLI] Peter Piroli und Stuart K. Card, *Information Foraging,* UIR Technical Report, 1999

[MEIER] Dieter Meier, *Segel Fliegen die Praxis*, Nymphenburger, 1982

[WIENHOLZ] Kerstin Wienholz, *GNSS und DGNSS -absolute und relative Positionsbestimmung*, TU Berlin, 2002

[ZHAO] Yilin Zhao, *Vehicle Location and Navigation Systems*, Artech House INC, Norwood, 3. Auflage, 1997

[ZIMMER] Stefan Zimmer, *Geschichte der Navigation und Positionsbestimmung*

H2 Linkverzeichnis

Diplomarbeit: Aufbau und exemplarische Nutzung
Eine Satellitendatengestützten Navigations-Informations-Systems
http://www.haberlah.com/diploma/abu_tabari.pdf

Publikation: Aufwind im Wohnzimmer
http://www.gleitschirm-magazin.com/pdf/Compe.pdf

Publikation: Interactive Data Visualization using Focussing and Linking
http://delivery.acm.org/10.1145/950000/949633/p156-buja.pdf?key1=949633&key2=5823835511&coll=GUIDE&dl=GUIDE,ACM&CFID=11111111&CFTOKEN=2222222

Publikation: Using a Degree-of-Interest Model for Adaptive Visualizations in Protege
http://www.cs.uvic.ca/~chisel/pubs/protege06.pdf

Publikation: Realisierung von Positionsortungen in WLAN
http://www4.in.tum.de/~dornbusc/pubs/RvPiWLAN_final.pdf

Publikation: Using a Degree-of-Interest Model for Adaptive Visualizations in Protege
http://www.cs.uvic.ca/~chisel/pubs/protege06.pdf

Publikation: Generalized Fisheye Views
http://www.si.umich.edu/~furnas/Papers/FisheyeCHI86.pdf

Publikation: Information Foraging
http://citeseer.ist.psu.edu/cache/papers/cs/8049/http:zSzzSzwww.parc.xerox.comzSzistlzSzprojectszSzuirzSzpubszSzpdfzSzUIR-R-1999-05-Pirolli-Report-InfoForaging.pdf/pirolli99information.pdf

Publikation: GNSS und DGNSS -absolute und relative Positionsbestimmung
http://www.survey.tu-berlin.de/gielsdorf/skripte/gnss_2.pdf

Publikation: Geschichte der Navigation und Positionsbestimmung
http://interaction.hgkz.ch/projects/gps/wp-content/files/GPS_Zusammenfassung.pdf