ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Entwicklung einer In-House-AR-Navigation unter Verwendung von AREA und Wifinder

Bachelorarbeitan der Universität Ulm

Vorgelegt von:
Tobias Waldenmaier
tobias.waldenmaier@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Rüdiger Pryss

2014

Fassung 12. Juni 2014

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

In dieser Arbeit wird eine In-House Navigationsanwendung entwickelt, die zur Darstellung der Navigationsinformationen Augmented-Reality benutzt. Dabei wird ein standortbasiertes Augmented-Reality System benutzt, das von der App AREA, entwickelt von Philip Geiger, zur Verfügung gestellt wird. Zur Ermittlung des Standortes innerhalb eines Gebäudes kommt die Positionierungsfunktion der App Wifinder, entwickelt von Alexander Bachmeier, zum Einsatz. Auch die Routingfunktion wird von dieser App zur Verfügung gestellt. Die Zusammenführung und Anpassung der beiden Apps ist Teil dieser Arbeit.

Danksagung

Ich danke meinem Betreuer Rüdiger Pryss für seine Geduld und fortwährende Unterstützung während der gesamten Zeit. Außerdem danke ich Alexander Bachmeier und Philip Geiger für die gute Vorarbeit und ihre Hilfe bei meiner eigenen Arbeit.
Besonders Danken möchte ich meiner Freundin Julia, die mir nicht nur beim Korrigieren geholfen hat, sondern auch großes Verständnis für die durchgearbeiteten Wochenenden aufgebracht hat. Danke!

Inhaltsverzeichnis

1

Einleitung

Seit einiger Zeit werden mobile Augmented Reality Anwendungen für Smartphones immer populärer. Der Grund dafür ist, dass moderne Smartphones inzwischen genug Rechenleistung haben, um Augmented Reality benutzerfreundlich, das heißt mit ausreichender Geschwindigkeit, zu realisieren. Bei Augmented Reality wird eine Szene von einer Kamera erfasst und um zusätzliche, künstliche Informationen erweitert. Es liegt daher nahe, auch Navigationsanwendungen mit Hilfe von Augmented Reality zu realisieren, vor allem vor dem Hintergrund von "Wearable Devices", wie zum Beispiel Google Glass, der Kamera-Datenbrille von Google. Eine solche oder ähnliche Brille mit Headup-Display würde Augmented Reality zu einer sehr natürlichen Möglichkeit zur Darstellung von relevanten Informationen machen. Deshalb werden im Bereich der Augmented Reality immer mehr und vor allem auch aufwendigere Anwendungen dieser Art entwickelt. Da sich solche Datenbrillen jedoch noch immer in der Entwicklung befinden

und nur als Prototypen verfügbar sind, bleibt als Entwicklungs- und Testplatform nur ein Smartphone oder ein ähnliches mobiles Gerät wie zum Beispiel ein Tablet-PC.

In dieser Arbeit soll eine Augmented-Reality Navigationsanwedung für Innenbereiche in großen Gebäuden entwickelt werden. Dazu werden zwei bereits bestehende, und ebenfalls als Abschlussarbeiten entstandene Applikationen verschmolzen. Diese beiden Apps sind zum einen die Anwendung AREA von Philip Geiger [Gei12] und zum anderen die App Wifinder von Alexander Bachmeier [Bac13]. AREA stellt hierbei eine GPS benutzende, standortbasierte Augmented-Reality Engine zur Verfügung [GSP+14], [GPSR13]. Da GPS in Gebäuden nicht funktioniert, muss in diesem Fall die Position auf andere Weise bestimmt werden. Hierfür kommt Wifinder zum Einsatz, das die Position mit Hilfe von Wifi Signalen bestimmt. Ziel ist es, mögliche Probleme beim Zusammenführen dieser beiden Apps aufzudecken und, nach Möglichkeit, auch Lösungen dafür zu finden. Außerdem wird eine geeignete Darstellung für die Navigationsinformationen vorgestellt werden, die leicht verständlich und übersichtlich ist und so eine angenehme Nutzung der Augmented-Reality Technologie zur Navigation innerhalb eines Gebäudes ermöglicht.

Im Folgenden werden zunächst einige allgemeine Grundlagen zu Augmented-Reality diskutiert, bevor eine Einführung in die beiden schon bestehenden Apps AREA und Wifinder erfolgt. Zudem werden andere, ähnliche Apps und Forschungsarbeiten zu diesem Thema vorgestellt. Anschließend werden die Anforderungen festgelegt, die an den hier entwickelten Prototypen gestellt werden. Im vierten Kapitel werden dann die wichtigsten Schritte bei der Implementierung, bzw. der Zusammenführung von AREA und Wifinder erläutert. Hiernach wird der Prototyp vorgestellt und seine Funktionsweise erläutert. Nach einer Zusammenfassung erfolgt dann ein Ausblick auf mögliche Erweiterungen und Verbesserungen, die am entstandenen Prototypen noch vorgenommen werden können.

2

Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der hier entwickelten Indoor-Augmented-Reality Navigationsapp. Um Augmented Reality zu ermöglichen gibt es zwei grundsätzlich verschiedene Ansätze. Die erste Möglichkeit setzt dabei auf Bilderkennung und Computer Vision. Dabei wird das Livebild der Kamera eines Mobilen Gerätes mit Hilfe von Bilderkennungsalgorithmen aus dem Bereich der Computer Vision analysiert. Werden signifikante Merkmale oder Strukturen im Livebild entdeckt, können im zweiten Schritt Informationen dazu eingeblendet werden. Diese Art der Augmented Reality ermöglicht es, Objekte unabhängig von ihrem Ort zu erkennen.Damit ist sie zum Beispiel für interaktive Bauanleitungen geeignet, bei denen zu jedem erkannten Bauteil Anweisungen eingeblendet werden können. Andere Einsatzfelder umfassen Stadtführer, die Fakten zu Sehenswürdigkeiten anzeigen oder Navigationsanwendungen, die entsprechende Informationen direkt im Livebild einblenden. Allerdings hat diese Vorgehensweise einen großen Nachteil: Es wird sehr viel Rechenleistung zur Bildverarbeitung

benötigt, vor allem da dies sehr schnell geschehen muss, damit die Informationen in das Livebild eingeblendet werden können und der Benutzer möglichst wenig Verzögerung zu spüren bekommt. Außerdem würde man für eine Navigationsanwendung immer noch die Position des Benutzers bestimmen müssen. Dies geht unter freiem Himmel recht einfach per GPS, in einem Gebäude jedoch nur mit einer sehr großen Datenbank an Vergleichsbildern, was viel zusätzlichen Aufwand bedeuten würde. Die zweite Möglichkeit ist eine standortbasierte Augmented-Reality Engine. Bei dieser Art der Augmented Reality wird nicht erkannt, was sich im Bild befindet, sondern es wird errechnet, was sich wo im Bild befinden müsste. Dazu wird zunächst die Position bestimmt. Dies geschieht meistens per GPS, manchmal aber auch über spezielle Marker, die von der Kamera erkannt werden und einer festen Position zugeordnet werden können. Anschließend wird mit Hilfe der Sensoren des mobilen Gerätes die genaue Ausrichtung der Kamera bestimmt. Unter Zuhilfenahme einer Datenbank mit Points-of-Interest (kurz: POI) und deren genauer Position kann jetzt errechnet werden, welche Punkte sich in der Sichtlinie der Kamera befinden und wo auf dem Bildschirm diese sind. Auf diese Weise können an der entsprechenden Stelle Informationen eingeblendet werden. Diese Vorgehensweise kostet erheblich weniger Rechenleistung und funktioniert auch schnell genug, dass es angenehm zu benutzen ist. Der größte Vorteil hierbei ist, dass die genaue Position des Benutzers gewissermaßen als Nebenprodukt schon bekannt ist, was vor allem für eine Navigationsanwedung von enormem Nutzen ist. Auch hier stellt sich jedoch das Problem, dass es innerhalb von Gebäuden kein GPS-Signal gibt. Die hier verwendete Lösung des Problems ist die Wifi-basierte Navigationsanwendung Wifinder von Alexander Bachmeier [Bac13]. Diese Anwendung ermittelt die Position eines Benutzers in einem Gebäude unter Verwendung von Wifi-Signalen. Die erhaltene Positionsinformation wird mit der Augmented Reality Engine von AREA von Philip Geiger [Gei12] verwendet um, eine Indoor-Augmented Reality Navigationsapp zu ermöglichen. Beide Anwendungen werden nun kurz in ihrer Funktionsweise erläutert.

2.1 AREA

Die für den Augmented-Reality Teil dieser Arbeit notwendige Augmented-Reality Engine wird von AREA zur Verfügung gestellt [Gei12]. AREA ist eine Augmented-Reality Sightseeing App und hat eine vergleichsweise einfache, gleichzeitig aber doch leistungsfähige Augmented-Reality Engine. Diese basiert nicht, wie viele andere Lösungen, auf Computer Vision und Bilderkennung, wie es zum Beispiel bei openCV [ope] oder dem Vuforia SDK [vuf] der Fall ist. Diese Varianten sind zwar leistungsfähiger, allerdings auch deutlich langsamer und komplexer. Außerdem erscheinen sie wenig geeignet für eine Verwendung in einem Gebäude, da es dort zu wenige unterschiedliche Marker gibt. Stattdessen verwendet AREA ein sogenanntes sensorenbasiertes Tracking. Dabei werden die Sensoren des Gerätes verwendet um den Kontext, in dem es sich befindet, zu erkennen und dementsprechend Informationen auf dem Display einzublenden. Dazu wird zunächst die genaue Position des Benutzers bestimmt. Hierfür kann sowohl GPS zum Einsatz kommen, als auch, als Unterstützung, eine netzwerkbasierte Positionsbestimmung. Ist die Position bekannt, wird als nächstes die Ausrichtung des Gerätes mit Hilfe der internen Lagesensoren (Kompass, Gyroskop, Beschleunigungssensor) bestimmt. Die genaue Lage wird benötigt, um zu bestimmen in welche Richtung die Kamera des Gerätes gerichtet ist, und berechnet werden kann, welche Punkte (zum Beispiel GPS-Koordinaten) im Blickfeld der Kamera liegen. Diese Berechnung muss bei jeder Bewegung des Gerätes wiederholt werden damit immer aktuelle Informationen auf dem Display eingeblendet werden können. Sobald bekannt ist, was im Blickfeld der Kamera liegt, kann dies mit einer Datenbank mit sogenannten Points of Interest, kurz POIs, abgeglichen werden. Bei einem Treffer, das heißt ein POI liegt im Blickfeld der Kamera, wird dieser an der entsprechenden Stelle auf dem Display eingeblendet. Zusätzliche Informationen zu diesem POI lassen sich dann durch Antippen abrufen. Dies ist, wie schon angedeutet, eine relativ simple, gleichzeitig aber sehr leistungsfähige Vorgehensweise um Augmented Reality zu ermöglichen. Speziell der Einsatz innnerhalb eines großen Gebäudes ist möglich, da die Engine nicht auf externe Marker angewiesen ist. Das einzige Problem, das sich stellt, ist die Positionsbestimmung. Innerhalb eines großen Gebäudes gibt es normalerweise keine Möglicheit mit GPS die Position zu

bestimmen, da kein Kontakt zu den GPS Satelliten hergestellt werden kann. Um dieses Problem zu lösen, wird hier das Wifinder Projekt verwendet.

2.2 Wifinder

Wifinder ist eine Navigationsanwendung, die eine In-House Navigation ermöglicht, ohne dabei auf GPS zu setzen. Um dies zu erreichen müssen nur einige wenige Bedingungen erfüllt sein:

1. Es muss möglichst genaues Kartenmaterial des Gebäudes vorliegen.
2. Das Kartenmaterial muss mit den korrekten GPS-Koordinaten verknüpft sein (zum Beispiel OpenStreetMap)
3. Im Gebäude muss es ein gut ausgebautes Wifi-Netzwerk geben. Dabei gilt: je mehr Zugangspunkte desto besser.

Mehr ist nicht erforderlich, um Wifinder zu verwenden. Vor allem muss hervorgehoben werden, dass keinerlei zusätzliche Peripheriegeräte, zum Beispiel für ein künstliches GPS Netz, benötigt werden. Nur die drei oben genannten Bedingungen müssen erfüllt sein. Da die meisten Gebäude, bei denen eine In-House Navigation sinnvoll wäre, große öffentliche Gebäude sind, gibt es dort in den häufigsten Fällen ein gut ausgebautes Wifi-Netzwerk, sowie detailliertes Kartenmaterial. Um Wifinder nun verwenden zu können muss es zunächst konfiguriert werden. Dies geschieht, indem man an jedem Punkt, der später ein Punkt im Navigationsgraph werden soll, einen Wifi-Scan durchführt. Dabei werden für jeden dieser Punkte die SSID, die Signalstärke aller empfangbaren Zuganspunkte und natürlich die aus dem Kartenmaterial gewonnenen GPS Koordinaten gespeichert. Dadurch entsteht für jeden Punkt ein Wifi-Fingerabdruck, der hoffentlich möglichst einzigartig ist. Ist dies für alle relevanten Punkte geschehen, ist Wifinder bereit zum Einsatz. Wenn man jetzt mit einem Gerät mit Wifinder durch das Gebäude geht, scannt Wifinder alle empfangbaren Wifi-Zugangspunkte und vergleicht sie mit den in der Datenbank gespeicherten Punkten. Durch diesen Vergleich kann die Position des Gerätes und damit des Benutzers bestimmt werden. Dabei gilt, dass je mehr unterschiedliche

Wifi-Zugangspunkte und Netze existieren, desto genauer kann die Position bestimmt werden. Ein Zugang zu diesen Netzen ist nicht nötig, da nur Signalstärke und SSID benötigt werden.

2.3 Ähnliche Arbeiten

In der Vergangenheit gab es schon einige andere Versuche eine Indoor-AR Navigation aufzubauen. Sie verwenden meistens spezielle Marker die im Gebäude angebracht werden.

So zum Beispiel Signpost [RS03], eine Augmented Reality Anwendung, die im Jahr 2001 von Gerhard Reitmayr und Dieter Schmalstieg entwickelt wurde. Die von ihnen entwickelte standortbasierte Augmented-Reality Engine setzt zur Positionsbestimmung ganz auf spezielle Marker, die im Gebäude im Abstand von ungefähr zwei Metern angebracht werden müssen. Da absehbar ist, dass auf diese Weise in einem großen Gebäude mehrere hundert oder sogar mehrere tausend einzigartige Marker benötigt werden, haben sie außerdem ein System entwickelt, welches ein mehrmaliges Benutzen der Marker ermöglicht. Der Benutzer muss nur einen dieser Marker im Blickfeld haben, damit das System die Position bestimmen kann. Wird jetzt ein Ziel ausgewählt, so wird mit Hilfe eines Algorithmus für den kürzesten Weg die Route berechnet und die Richtungsanweisungen als Pfeile im Bild eingeblendet. Diese Anwendung wurde zwar für ein tragbares System mit Head-Up-Display entwickelt, würde so aber wohl auch auf einem modernen Smartphone funktionieren.

In der App Junaio [jun] wird die Position ebenfalls durch Marker erfasst. Diese sind auf dem Boden angebracht. Nach dem scannen eines Markers kann man sich mit der Kamera des mobilen Gerätes umsehen und bekommt alle relevanten Informationen ins Bild eingeblendet. Bei jeder Änderung der Position muss hier zuerst wieder ein Marker gescannt werden, damit die Junaio App wieder weiß, wo sich das Gerät befindet.

In der Arbeit von Hisashi Aoki, Bernt Schiele und Alex Pentland [ASP99] wird ein System beschrieben, das Bildsequenzen analysiert und mit einer Datenbank abgleicht. Dies

geschieht in Echtzeit und ermöglicht die Bestimmung des Standortes des Benutzers mit Hilfe von Computer Vision.

Ein weiterer Ansatz aus dem Jahr 2011 stammt von Chi-Yi Lin, Chih-Yuan Chang und Hui-Huang Hsu. In ihrer Arbeit [LCH11] beschreiben sie ein standortbasiertes Augmented-Reality System bei dem die Lokalisierung über RFID gelöst wird. Hierbei verwendet der Benutzer entweder den im Smartphone verbauten RFID-Chip oder, wenn dieser nicht vorhanden ist, einen normalen RFID-Tag. Im Gebäude werden RFID Lesegeräte aufgestellt. Die Position des Benutzers lässt sich nun bestimmen, indem überprüft wird welche Lesegeräte ein Signal vom RFID-Chip des Benutzers empfangen und wie stark dieses Signal ist. Die Verarbeitung dieser Daten findet auf einem externen Server statt, der die ermittelte Position über Wifi an das Smartphone des Benutzers sendet. Eine schematische Darstellung findet sich in Abbildung 2.1.

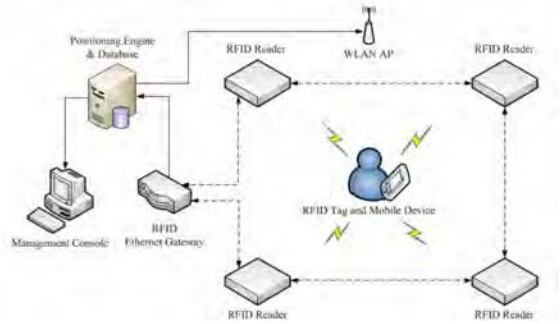

Abbildung 2.1: Schema des vorgeschlagenen RFID-Positionierungssystems. Vgl. [LCH11]

Die App “AR Navigation for Pedestrians” bietet einen Augmented-Reality Navigation für Fussgänger an. Diese Funktioniert jedoch nicht in Innenräumen, da zur Standortbestimmung nur GPS zum Einsatz kommt. Die Navigationsinformationen werden mittels eines Pfeils im Display eingeblendet. Der Pfeil zeigt dabei die Richtung an, in die der Benutzer gehen muss.

Genauso funktioniert die App “AR GPS DRIVE/WALK NAVIGATION”. Auch sie benutzt zur Standortbestimmung GPS. Der Unterschied ist, dass die Navigationsinformationen hier direkt als Route eingeblendet werden. Man folgt also nicht der Richtung eines Pfeiles,

sondern einer Linie, die schlussendlich zum Ziel führt. Die App ist laut Herstellerangaben für Fussgänger und Autofahrer geeigent . Da sie ebenfalls nur auf GPS setzt, funktioniert sie nur unter freiem Himmel.

In der Arbeit “AR-based indoor navigation system for personal locating” [JBKD06] wird ein System zur Bestimmung der eigenen Position mittels Bildsequenzen und Farb-Histogramm-vergleichen beschrieben. Auch hier erfolgt die Positionsbestimmung in Echtzeit. Das Ergebnis wird dann in einer Augmented-Reality Anwendung verwendet.

3

Anforderungen

In dieser Arbeit wird eine In-House Navigationsapp entwickelt. Dazu werden AREA, als Augmented Reality Engine und Wifinder, als Lokalisierungs- und Routendienst zusammengeführt. So entsteht ein Hybrid, der die Navigation über Augmented Reality ermöglicht.

Damit die Navigation des Benutzers möglichst reibungslos funktioniert, werden einige Anforderungen an den Prototypen gestellt:

- Die App soll schnell sein, das heißt, es darf nicht zu lange dauern eine gewünschte Route einzugeben, dann die Route zu berechnen und in ein Augmented Reality-taugliches Format umzuwandeln und schließlich dem Benutzer anzuzeigen.
- Das Bedienen der App sollte einfach von der Hand gehen. Die entsprechenden Bedienelemente sollten leicht zu erreichen sein und dabei nicht zu viel von der Kameraansicht der Augmented Reality Engine verdecken. Generell ist es eine

sehr komplexe Aufgabe, bedienbare Apps für einen bestimmten Kontext zu entwickeln [PMLR14], [SSP+14], [PLRH12], [RPR11].

- Die vorhandenen Bedienelemente müssen intuitiv sein, sodass ein Benutzer nicht lange überlegen muss, welche Funktion sich hinter welchem Element versteckt. Auch dieser Aspekt kann sehr entscheidend sein [CNB+13], [IRLP+13], [RLPL+13], [SRLP+13].
- Die Informationen, die dem Benutzer in der Augmented-Reality-View angezeigt werden, sollen genau sein, das heißt die Navigationssymbole müssen an der richtigen Stelle angezeigt werden. Dies hängt natürlich unmittelbar von der Genauigkeit der Lokalisierung des Benutzers ab.
- Die Augmented Reality Ansicht soll eine Routenführung ermöglichen. Dies kann zum Beispiel durch eingeblendete Navigationssymbole erfolgen.
- Der Benutzer soll ein beliebiges Ziel innerhalb des abgedeckten Navigationsbereiches wählen können.
- Bei Bedarf soll auch der Startpunkt frei wählbar sein.

Für eine bessere Übersicht sind hier die Anforderungen noch einmal in Tabellenform und mit der Unterscheidung zwischen funktionalen und nicht-funktionalen Anforderungen dargestellt:

Anforderung	Typ der Anforderung
Hohe Geschwindigkeit des Prototyps	nicht-funktional
Kurze Wartezeiten	nicht-funktional
Einfache Bedienung	nicht-funktional
Routenführung mittles AR	funktional
Navsymbole mit nötigen Informationen	funktional
Interaktion mit Navsymbolen gibt zusätzliche Informationen	funktional
Hohe Präzision bei der Routenführung	nicht-funktional
Bei Bewegung wird die Position der Punkte aktualisiert	funktional
Wahl des Startpunktes	funktional
Wahl des Zielpunktes	funktional
Hohe Stabilität	nicht-funktional

4

Verwendete Technologien und Architektur des Prototypen

Dieses Kapitel befasst sich im ersten Abschnitt mit den im hier entwickelten Prototypen verwendeten Technologien.
Im zweiten Abschnitt wird ein Überblick über die Architektur des Prototypen gegeben.

4.1 Architektur

Die Architektur des neuen Prototyps ist naheliegenderweise eine Mischung aus den Architekturen von Wifinder und AREA. Einen Überblick über den Aufbau gibt Abbildung 4.1. Der Server ist unverändert aus Wifinder übernommen. Die Clientanwendung folgt dem Model-View-Controller Konzept. Der REST-Serviceteil gehört formal ebenfalls

zum Controllerteil, ist aber aus Übersichts- und Verständnisgründen in der Abbildung ausgelagert dargestellt. Außer dem REST-Service befindet sich im Controller alles was mit der Bestimmung der Position, der Berechnung der Route und der Darstellung der Route in der Kameransicht zu tun hat. Besonders wichtig sind der Location-Controller und der Sensor-Controller. Sie erledigen den größten Teil der Arbeit in der Client Anwendung.
Alle Datenmodelle die verwendet werden, zum Beispiel `AREAGeoLocation`, sind im Model vereint. Es stellt diese Modelle an den Rest der Anwendung zur Verfügung und ermöglicht so einen wohldefinierten Datenfluss.
Der View-Teil besteht nur aus der Kameraansicht. Genau genommen ist hier auch noch die Bedienoberfläche von Wifinder enthalten, diese ist aber zur Benutzung des neuen Prototypen nicht relevant und kann später auch ohne Probleme entfernt werden.

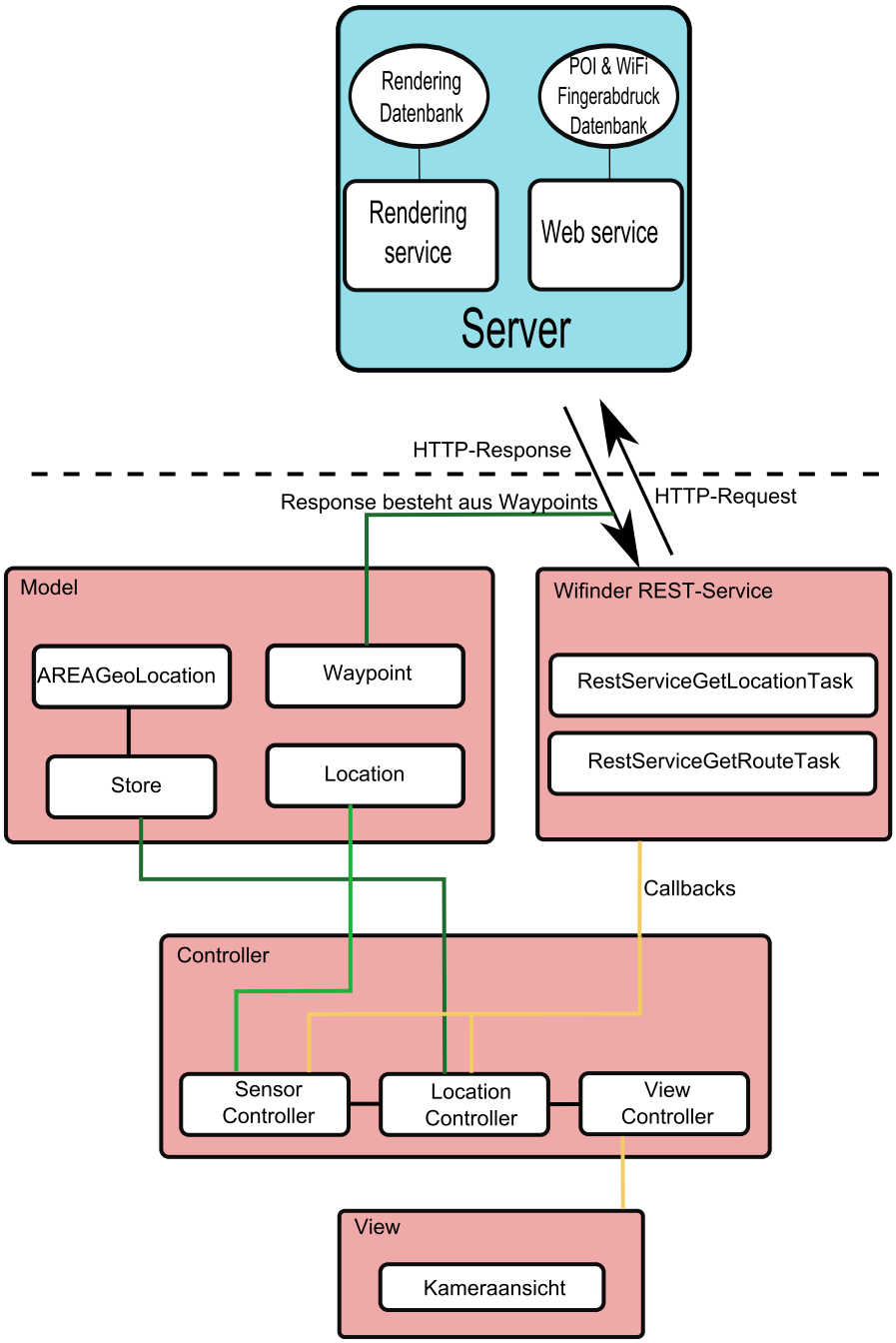

Abbildung 4.1: Überblick über die Architektur.

4.2 Verwendete Technologien und Mechanismen

Der hier entwickelte Prototyp vereint zwei bereits existierende Applikationen miteinander. Um eine Reibungslose Kommunikation zwischen den einzelnen Komponenten beider Applikationen zu gewährleisten, kommen hier verschiedene Mechanismen und Technologien zum Einsatz. Zwei von ihnen sind besonders hervorzuheben, da sie den größten Teil der Kommunikationsaufgaben bewältigen.
Um eine schnelle und einfache Kommunikation zwischen dem Wifinder-Server und der Client-Anwendung zu ermöglichen, wird eine HTTP Verbindung genutzt. Die Details zu dieser Verbindung werden komplett vom REST-Service übernommen. Der HTTP-Request enthält dabei entweder eine Standortanfrage oder eine Routenanfrage. Die HTTP-Response auf die Standortanfrage enthält dann die entsprechenden GPS Koordinaten des Client-Gerätes. Die Antwort auf eine Routenanfrage enthält eine Liste von `Waypoint`s, die die angefragte Route definieren.
Die Kommunikation innerhalb des Clients wird in Wifinder über Callbacks gelöst. Dieses Prinzip wird auch vom hier entwickelten Prototypen übernommen. Bei einem Callback wird eine Funktion aufgerufen, der zusätzlich zu deren normalen Parametern auch noch eine andere Funktion übergeben wird. Diese übergebene Funktion wird, unter vorher definierten Umständen, aufgerufen(vgl. [cal]). Da in Java keine Funktionen als Parameter übergeben werden können, und eine Android App in Java geschrieben wird, müssen Callbacks mit Hilfe von Interfaces implementiert werden. In diesem Prototypen werden Callbacks vor allem zur Kommunikation zwischen den REST-Service Klassen und dem Controller benutzt. Die Daten, die vom Server kommen, werden in Callbacks an die entsprechenden Ziele weitergeleitet, wo sie dann weiter verarbeitet werden können.
In der AREA App wurde zum Hinzufügen von Points-of-Interest ein XML-basiertes Verfahren eingesetzt. Die POIs wurden in einer XML Datei gespeichert und von einem XML-Parser ausgelesen und dem `AREAStore` hinzugefügt. Dies ist in diesem Prototypen nicht mehr nötig, da alle Punkte, die zum `AREAStore` hinzugefügt werden müssen, per Callback vom REST-Service kommen.

4.3 Der Wifinder-Server

Damit der hier entwickelte Prototyp funktioniert, benötigt er einen laufenden Wifinder-Server. An der Implementierung dieses Servers muss für den Prototypen nichts geändert werden, er wird daher im Implementierungskapitel nicht beschrieben. Da er aber trotzdem essentiell ist, wird der Aufbau der Vollständigkeit halber hier erläutert.
Der Server besteht aus zwei Teilen: dem Rendering-Service und dem Web-Service.
Der Rendering-Service ist in Wifinder für das Rendern und zur Verfügung stellen des Kartenmaterials zuständig. Dieser Teil des Servers wird für die Augmented-Reality Navigation aber nicht benötigt und daher ignoriert. In einer späteren Version kann dieser Teil des Servers aber durchaus wieder Bedeutung finden (siehe Ausblick).
Der Web-Service ist der wichtige Teil des Servers. Er beinhaltet die wichtige WiFi-Fingerabdruck Datenbank mit deren Hilfe die gesamte Positionierung des Prototyps funktioniert. Die Kommunikation zwischen dem Server und der hier entwickelten Clientanwendung funktioniert über HTTP-Requests und HTTP-Responses.
Eine ausführlichere Beschreibung des Servers, sowie eine Installationsanleitung ist in der Arbeit von Alexander Bachmeier [Bac13] zu finden.

5

Implementierungsaspekte

In diesem Kapitel werden die wichtigsten Implementierungsaspekte erläutert. Im ersten Teil werden relevante Implentierungsschritte diskutiert.

Der Augmented Reality Navigations-Prototyp der hier entwickelt wird, baut, wie schon erwähnt, auf den beiden Apps AREA und Wifinder auf. Beide Apps liegen als Android App vor, weshalb auch der Prototyp als Android App entwickelt wird. Als Entwicklungsumgebung kommt Eclipse mit den Android-Developement-Tools [adt] zum Einsatz. Generell müssen Implementierungsmodelle und verwendete Frameworks sorgsam beachtet werden [SSP+13].

5.1 Implementierung

Bei der Implementierung, bzw. Zusammenführung von AREA und Wifinder, sind vor Allem zwei Aspekte besonders zu beachten:

1. Die Erkennung und Festlegung der Position des Benutzers, bzw. des mobilen Gerätes.
2. Die Beschaffung und anschließende Darstellung der Routeninformationen in der Augemted-Reality Kameraansicht.

Beide Punkte erfordern Eingriffe in verschiedene Teile des Controllers der AREA-Engine. Die genauen Änderungen und Anpassungen werden im Folgenden beschrieben.

5.1.1 Konvertierung der Wifinder GPS Koordinaten in AREA-GeoLocations

In Wifinder werden zur Berechnung und auch Darstellung in der Kartenansicht echte GPS Koordinaten verwendet. Das ist möglich, weil Kartenmaterial und Geoinformationen von OpenStreetMap [osm] verwendet werden, und so echte GPS Koordinaten auf dem Kartenmaterial registriert werden können. Allerdings werden anstatt echter Höhenangaben nur Stockwerknummern vergeben. In AREA wird hingegen eine eigene Klasse für den Standort verwendet, nämlich `AREAGeoLocation`. `AREAGeoLocation` enthält zwar auch GPS Informationen, allerdings mit echten Höhenwerten und nicht im Android-eigenen `Location`-Format. Deshalb müssen alle Wifinder Koordinaten in `AREAGeoLocation` umgewandelt werden, bevor sie in der AREA-Engine verwendet werden können. Diese Konvertierung wird sowohl für die Bestimmung der eigenen Position benötigt, als auch für die Bestimmung einer Route. Deshalb ist dies der erste wichtige Schritt in der Zusammenführung der beiden Anwendungen.

Als erstes müssen die Stockwerkangaben, die Wifinder an Stelle von echten Höhenangaben verwendet, durch echte, passende Höhenangaben für die betreffenden Stockwerke ersetzt werden. Dazu wird eine einfache Funktion genutzt, die aus einer an sie übergebenen WifiLocation die Stockwerke ausliest und dann dementsprechende korrekte

Höhenwerte zurückliefert:

```
1  public double getAlti(WifiLocation location){
2                  switch (location.getLevel()){
3                          case 1: return 600.0;
4                          case 2: return 610.0;
5                          case 3: return 620.0;
6                          case 4: return 630.0;
7                          case 5: return 640.0;
8                          default: return 600.0;
9                  }
10         }
```

Ohne diesen Schritt könnten keine korrekten Navigationspunkte erzeugt weden, da sie nicht in der richtigen Höhe des Gebäudes dargestellt werden würden. Dies ist allerdings nur der Erste Schritt der Konvertierung. Im zweiten Teil erfolgt die tatsächliche Umwandlung der WifiLocation in ein gültiges `AREAGeoLocation` Format, bzw. ein Android Standart Location Format, wie es bei der Bestimmung der eigenen Position benötigt wird:

```
1          public void updateMap(WifiLocation location){
2                  Location loc = new Location("myLocation");
3                  loc.setAltitude(getAlti(location));
4                  loc.setLongitude(location.getLongitude());
5                  loc.setLatitude(location.getLatitude());
```

Wie zu sehen ist, bekommt die Funktion `updateMap` eine WifiLocation übergeben. Um diese nun zu konvertieren, wird zunächst eine neue Location “loc” angelegt. Der Konstruktor hierfür nimmt als Parameter einen beliebigen String an. Es ist nicht wichtig, was hier übergeben wird. Anschließend werden nacheinander die Höhe und die Längen- und Breitengrade aus der `WifiLocation` ausgelesen und dazu verwendet, die Werte in der zuvor angelegten Location “loc” zu überschreiben. Offensichtlich kommt hier auch die

oben gezeigte Funktion `getAlti` zum Einsatz, um die ausgelesenen Stockwerke sofort in echte Höhenangaben umzuwandeln. In “loc” sind nun alle korrekten Positionsdaten gespeichert und können entweder als GPS Koordinate zur Bestimmung der eigenen Position benutzt werden, wie es im nächsten Abschnitt gezeigt wird, oder “loc” wird benutzt, um eine `AREAGeoLocation` zu erstellen, was einem Navigationspunkt entspricht. Dies wird im übernächsten Abschnitt näher erläutert.

5.1.2 Bestimmung der eigenen Position

Nachdem gerade gezeigt wurde, wie aus der WifiLocation eine normale Android `Location` bzw. eine `AREAGeoLocation` erzeugt wird, soll jetzt erläutert werden, wie diese genutzt wird, um die eigene Position von Wifinder bestimmen zu lassen und anschließend an die AREA-Engine übergeben wird. Dazu muss zunächst gesagt werden, dass Wifinder zur Positionsbestimmung eine Anfrage zum Wifinder Server stellt. Der Server wird hier nicht näher erläutert, da an ihm keinerlei Änderungen oder Anpassungen erfolgen. Details zu diesem Server können der ursprünglichen Arbeit von Alexander Bachmeier [Bac13] entnommen werden. Die Anfrage zum Server besteht aus den momentan empfangbaren Wifi-Zugangspunkt SSIDs und deren Signalstärken. Dies entspricht dem Wifi-Fingerabdruck. Anhand dieses Fingerabdrucks identifiziert der Server die Position des mobilen Gerätes und sendet diese zurück zum Benutzer. Dieser Vorgang ist in Wifinder als REST-Service implementiert. Nachdem die Informationen des Severs angekommen sind, werden sie per Callback an die nächsten Verarbeitungsschritte weitergeleitet:

```
1 public void getLocation(List<ScanResult> scan) {
2 PositionScan mPositionScan = PositionScan.fromScanResult(scan);
3
4 RestServiceGetLocationTask getLocationTask = new
5                 RestServiceGetLocationTask(
6                 mContext, new FragmentCallback() {
7
```

```
8           @Override
9           public void updateLocation(WifiLocation mLocation) {
10                  onLocationChanged(mLocation);
11
12          }
13  });
14
15  getLocationTask.execute(mPositionScan);
16
17  }
```

Wie zu sehen ist, wird der Prozess zur Bestimmung der Position über die Funktion `getLocation` im `WifinderLocationProvider` angestoßen. Hier wird ein HTTP-Request an den Server erzeugt und hier befindet sich auch der “FragmentCallback”, der die erhaltenen Daten weiterleitet. Bei Erhalt der Daten vom Server bzw. der unten abgebildeten `onPostExecute` Funktion werden sie an die Funktion `onLocationChanged` weitergegeben.

```
1   @Override
2   protected void onPostExecute(HttpResponse response) {
3   HttpEntity entity = response.getEntity();
4
5   if (response.getStatusLine().getStatusCode()==
6                   HttpStatus.SC_OK) {
7           Reader reader = null;
8
9   WifiLocation mLocation;
10  try {
11  reader = new InputStreamReader(entity.getContent());
12  mLocation = new Gson().fromJson(reader, WifiLocation.class);
13  Toast.makeText(mContext,
```

```
14          "Location Response: " + mLocation.toString(),
15          Toast.LENGTH_LONG).show();
16
17  mFragmentCallback.updateLocation(mLocation);
18
19  [...]
```

In `onLocationChanged` wird ein weiterer Callback genutzt, um die Daten, die hier nur aus einer WifiLocation bestehen, an alle wichtigen Stellen weiterzuleiten. Ursprünglich war dies nur auf Wifinder begrenzt.

```
1  public void onLocationChanged(WifiLocation location) {
2                  updateMapCallback.updateMap(location);
3          }
```

Da dieser Callback über das Interface `updateMapCallback` funktioniert, kann man durch Implementieren dieses Interfaces eine Verbindung herstellen und die Daten auch an ein neues Ziel senden. In diesem Fall an die Klasse `AREASensorController`, welche das Interface nun implementiert:

```
1  public void updateMap(WifiLocation location){
2          Location loc = new Location("myLocation");
3
4          loc.setAltitude(getAlti(location));
5          loc.setLongitude(location.getLongitude());
6          loc.setLatitude(location.getLatitude());
7
8          currentLocation = loc;
9          Log.v(TAG, "Changed location to: " + loc.getLatitude()
10                 + " "
11                 + loc.getLongitude()
```

```
12                  + " " + loc.getAltitude());
13          notifyLocationChanged();
14          }
```

Die hier gezeigte `updateMap` Methode des `AREASensorControllers` wandelt zunächst die erhaltene `WifiLocation` in eine normale Android `Location` um und benutzt dabei auch die im vorherigen Abschnitt vorgestellten Mechanismen. Nachdem die WifiLocation konvertiert wurde, wird die neue Location als “currentLocation” gesetzt. Am Ende dieser Methode wird `notifyLocationChanged` aufgerufen. Diese Methode leitet den Prozess ein, der überprüft, welche Navigationspunkte sich im Sichtfeld der neuen eigenen Position befinden und auf dem Display angezeigt werden müssen.

5.1.3 Bestimmung der Routinginformationen

Im vorangegangenen Abschnitt wurde gezeigt wie die eigene Position, die Wifinder über den Wifi-Fingerabdruck bestimmt, an die AREA-Engine übergeben wird. Dieser Schritt ist der wichtigste Schritt zur Verwendung der Augmented Reality Engine von AREA für In-House-Bereiche. Der nächste bedeutende Schritt ist, dass in der Augmented Reality Kameraansicht, statt der in AREA üblichen Points-of-Interest, die Routeninformationen angezeigt werden. In AREA wurden die Points-of-Interest in einer Liste im sogenannten `AREAStore` gespeichert. Die offensichtlich einfachste und naheliegendste Lösung ist also, anstatt der Points-of-Interest, die Navigationspunkte der Routen in dieser Liste zu speichern. Auf diese Weise muss nichts weiter an der Funktionsweise der AREA-Engine geändert werden, außer die Befüllung dieser Liste.
Um überhaupt an die Routeninformationen zu gelangen, muss zunächst wieder eine Anfrage an den Server geschickt werden. In dieser Anfrage müssen Start und Ziel der gewünschten Route spezifiziert werden. Damit dies komfortabel geschehen kann, wurde in der Oberfläche von AREA ein Button “Neue Route” hinzugefügt. Mit diesem lässt sich ein kleines Dialogfenster (siehe Abbildung 5.1) öffnen, in dem Start und Ziel eingegeben werden können.

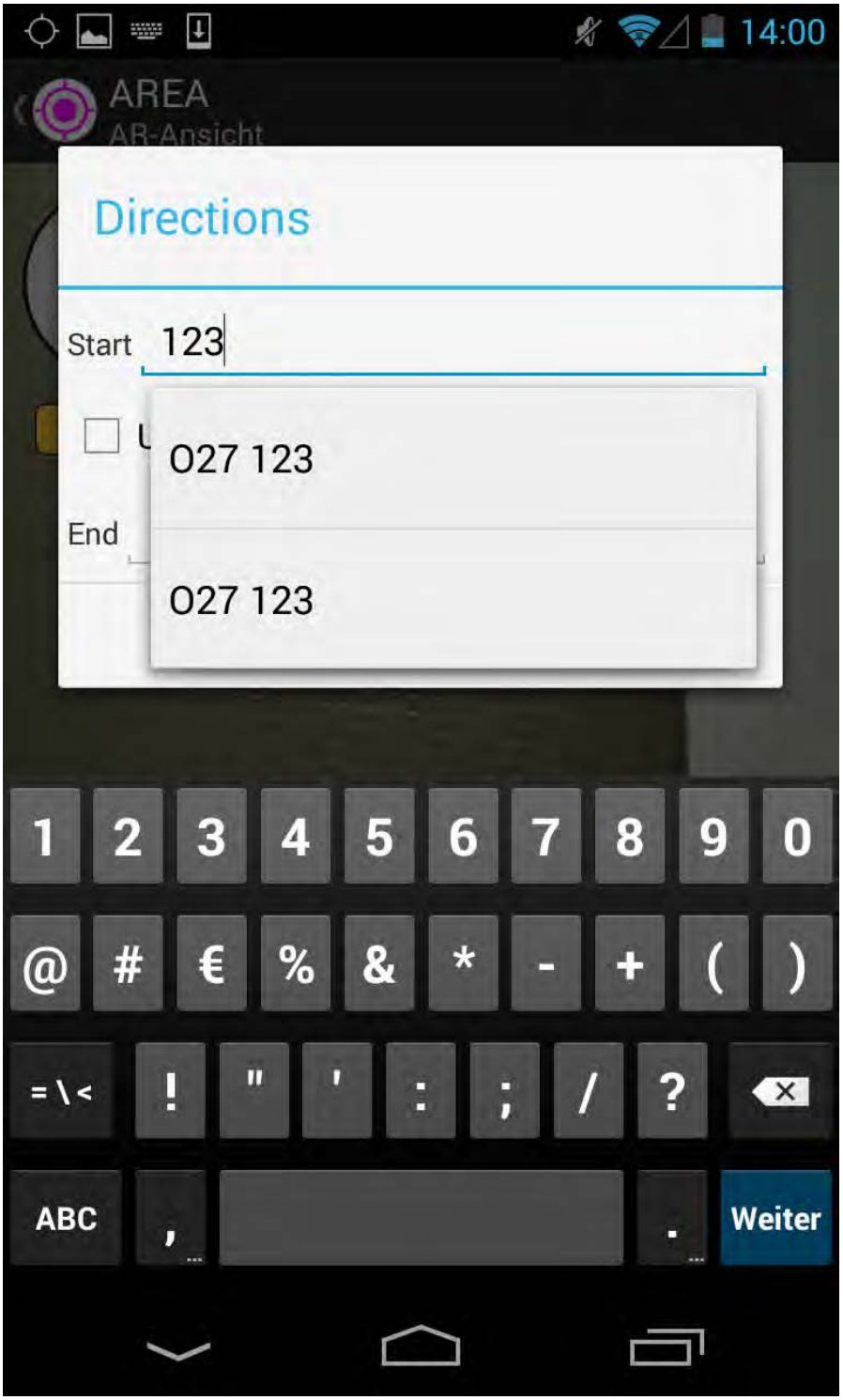

Abbildung 5.1: Eingabe von Start und Ziel. Dialog ist übernommen aus Wifinder.

Der Button für den Dialog befindet sich in der Menüleiste am unteren Rand der Kameraansicht.

```
public boolean onOptionsItemSelected(MenuItem item) {

if (item.getTitle() == "Neue Route"){
 GetDirectionsDialog dialog = new GetDirectionsDialog();
 dialog.setOnDialogClickedListener(new OnDialogClickedListener
    () {

 @Override
 public void onDialogClicked(String start, String end, boolean
 useCurrentLocation) {
        getRoute(start, end, useCurrentLocation);
 }
 });

 dialog.show(this.getSupportFragmentManager(), "dialog2");
 return true;
}
```

Nach Eingabe von Start und Ziel wird die Methode `getRoute` aufgerufen und damit auch die Anfrage an den Server gestartet. Die Antwort wird an die Methode `newRoute` im AREALocationController übergeben.

```
private void getRoute(String start, String end,boolean
   useCurrentLocation) {
if (useCurrentLocation) {
        startNavigation = true;
    navigationTarget = end;
    if (!scannerActive) {
```

```
6         startLocationScanner();
7       }
8   } else {
9       RestServiceGetRouteTask mRouteTask = new
            RestServiceGetRouteTask(
10      this, new RouteCallback() {
11
12      @Override
13      public void getRouteListener(Route mRoute) {
14          lc.newRoute(mRoute);
15      }
16      });
17
18          String[] params = new String[]{start, end};
19          mRouteTask.execute(params);
20
21          }
22  }
```

Im AREALocation Controller erfolgt das Speichern in den `AREAStore`. Dazu müssen die Wegpunkte, die vom Wifinder Server geliefert wurden und im `Waypoint` Format vorliegen, zunächst in `AREAGeoLocation` bzw. `AREAPointofInterest` umgewandelt werden. Dabei werden sie auch mit den nötigen zusätzlichen Informationen versehen, die für die Navigation wichtig sind. Dazu gehört zunächst die Nummer des Wegpunktes. Diese ist wichtig, um den Überblick zu behalten wenn mehr als ein Punkt im Sichtfeld der Kamera liegt, was mit hoher Wahrscheinlichkeit immer der Fall ist. Außerdem kann noch mehr detaillierte Information angegeben werden, die angezeigt wird wenn man den entsprechenden Wegpunkt anklickt. Hier wird zusätzlich zur Nummer des Wegpunktes auch noch das Stockwerk angezeigt, in dem der Wegpunkt sich befinden sollte. Dies ist eine Maßnahme, um mögliche Ungenauigkeiten bei der Positionsfindung aufzudecken. Es kann manchmal vorkommen, dass bei der Bestimmung der eigenen Position das

Stockwerk nicht richtig erkannt wird und dadurch der Punkt in der falschen Höhe angezeigt wird. Damit man dies zur Not manuell umgehen kann, wird das korrekte Stockwerk des Punktes hier noch einmal zur Verfügung gestellt.

```
1  public void newRoute(Route mRoute){
2
3  if (mRoute != null){
4   List<Waypoint> wpList = mRoute.getWaypoints();
5   Location loc = new Location("myLocation");
6   List<AREAPointOfInterest> list = new ArrayList<
       AREAPointOfInterest>();
7   for(int i=0; i<wpList.size();i++){
8    loc.setAltitude(getAlti(wpList.get(i)));
9    loc.setLatitude(wpList.get(i).getLatitude());
10   loc.setLongitude(wpList.get(i).getLongitude());
11   AREAPointOfInterest nav =  new AREAPointOfInterest(loc);
12   nav.setName(">"+ (i+1)+"<");
13   nav.setInformation("Das ist der "+ (i+1)+". Nav-Punkt");
14   AREAStore.getInstance().addGeoLocation(nav);
15         }
16  } else{
17                          [...]
18           }
19  }
```

5.1.4 Anpassen des User-Interface

In der ursprünglichen Version von AREA wurden die Points-Of-Interest mit Punkten im der Kameraansicht auf dem Display eingeblendet. Dies wird auch in diesem Prototyp so umgesetzt, nur dass die Punkte jetzt die Bedeutung von Kontrollpunkten im Verlauf der Route haben, und nicht mehr POIs sind.

Eine Route durch ein Gebäude kann allerdings schnell mehrere dutzend solcher Kontrollpunkte enthalten, die dann auch noch in einem relativ kleinen Raum beieinander liegen. Dies führt sehr schnell zu einer unübersichtlichen und schwer zu Interpretierenden Darstellung. Um diese Darstellung zu verbessern werden einige Anpassungen an den einfachen Punkten vorgenommen.
Alle Punkte der Route werden als gelber Punkt mit einem bestimmten Durchmesser angezeigt. Der Benutzer müsste jetzt die Namen der Punkte lesen und denjenigen mit der nächsten Nummer in seiner Route ausfindig machen und diesen dann ansteuern. Dort müsste er diesen Vorgang dann mit dem nächsten Punkt wiederholen, bis er schließlich am Ziel ankommt.
Offensichtlich ist dies ein sehr Mühsamer und langwieriger Prozess. Um dem Benutzer das Auffinden des nächsten anzusteuernden Punktes zu erleichtern wird der nächste Punkt immer in Grün anstelle von Gelb angezeigt. Auf diese Weise ist der nächste Kontrollpunkt immer leicht zu erkennen und der Benutzer kann sich immer schnell zum nächsten Ziel bewegen.
Zusätzlich zum jeweils nächsten Kontrollpunkt in der Route wird auch der letzte, endgültige Zielpunkt besonders hervorgehoben. Er wird anstelle von Gelb in Blau im Display eingeblendet, und ermöglicht dem Benutzer das Ende der von ihm angeforderten Route zu sehen und zu erkennen.
Die dritte und wahrscheinlich wichtigste Änderung betrifft den Durchmesser der Kontrollpunkte. Diese Punkte haben in AREA eine feste Größe die sich nie ändert. Dies macht Sinn für die ursprüngliche Version von AREA als AR-Sight-Seeing-App. In diesem Prototypen aber wird es zum Problem. Dadurch dass die Punkte hier nah beieinander liegen kommt es fast immer zu Häufungen und Überlappungen. Da alle Punkte gleich groß angezeigt werden kann man nicht ohne weiteres erkennen welche Punkte näher am Benutzer liegen und welche weiter weg sind. Um zumindest ein grobes Gefühl für die Entfernung der einzelnen Punkte vom Benutzer zu vermitteln wurde das Zeichnen der Punkte so abgeändert, dass der Durchmesser jetzt abhängig von der Entfernung ist. Je größer ein Punkt jetzt angezeigt wird, desto näher liegt er an der Position des Benutzers und umgekehrt, je kleiner ein Punkt dargestellt wird desto weiter entfernt liegt er.
Durch die Einführung dieser drei Änderungen:

- Hervorheben des jeweils nächsten Kontrollpunktes
- Hervorheben des Zielpunktes
- Durchmesser der Punkte von Entfernung abhängig machen

wurde die Benutzung des Prototypen deutlich vereinfacht und der Benutzer ist nicht mehr auf die nummerierten Namen der Punkte angewiesen, sondern bekommt ein einfaches und schnell zu erfassendes visuelles Feedback, dass eine flüssige und komfortable Bedienung der Prototypen ermöglicht.

5.2 Schwierigkeiten bei der Zusammenführung

Im Verlauf der Zusammenführung und Anpassung von AREA und Wifinder gab es auch einige Schwierigkeiten zu überwinden.
Zunächst ist die nicht ganz einfache Installation der beiden Apps zu nennen. Beide Apps benutzen externe Bibliotheken, wie zum Beispiel die Google-Play-Services Bibliothek. Diese muss bei der Installation, aufgrund von Neuerung durch Google durch einen neue Version ersetzt werden.
Auch die Komplexität und Menge an fremdem Code war ein Problem. Beide Apps sind für sich genommen schon recht groß und komplex. Bei der Zusammenführung müssen Aspekte von beiden Apps beachtet werden. Es wurde daher viel Zeit darauf verwendet, den Quellcode, sowohl von AREA als auch von Wifinder zu verstehen. Auch das Herstellen einer funktionierenden Kommunikation zwischen den unterschiedlichen Teilen der Apps war eine Herausorderung. Grund dafür sind die verschiedenen Programmierkonzepte, die zum Einsatz kommen und nicht ohne Weiteres miteinander kompatibel waren.

6

Abgleich der Anforderungen

In diesem Kapitel werden noch einmal die früher aufgestellten Anforderungen betrachtet und bewertet, in welchen Grad diese erfüllt werden konnten.

Die erste Anforderung war, dass der Prototyp eine möglichst hohe Geschwindigkeit hat und nicht zu lange Wartezeiten verursacht. Der hier entwickelte Prototyp benutzt als Augmented-Reality Grundkern die AREA Engine, was einen enormen Geschwindigkeitsvorteil bringt, da diese Engine sehr schnell und leichtgewichtig arbeitet. Das Einzige, was die Geschwindigkeit beeinträchtigt, sind die Anfragen zum Server für die Bestimmung der eigenen Position und für die Ermittlung einer Route. Selbst diese Zeiten halten sich jedoch in einem komfortablen Rahmen, so dass dies nicht störend auffällt.
An zweiter Stelle wurde eine einfache Bedienung verlangt. Auch diese Anforderung wird erfüllt. Der hier entwickelte Prototyp hat in der Kameraansicht nur zwei Bedienelemente. Das Radar und den “Neue Route” Button. Der Button, genau wie der damit verbundene

Dialog zur Routeneingabe, sind durch ihre Beschriftung selbsterklärend. Jeder, der schon einmal etwas mit einer Kartenanwendung zu tun hatte, sollte auf Anhieb damit zurecht kommen. Auch das Radar ist einfach zu verstehen und hat sich bereits in der App AREA bewährt.
Die wichtigste Anforderung ist wohl die Funktion der Routenführung, da dies die Hauptaufgabe dieses Prototypen ist. Diese Anforderung kann ebenfalls als erfüllt betrachtet werden. Zur Routenführung werden in der Kameraansicht die entsprechenden Navigationspunkte angezeigt und es kann ihnen bis zum gewünschten Ziel gefolgt werden.
Die unmittelbar aus der Navigationsfunktion folgende Anforderung ist die Genauigkeit. Es wird verlangt, dass die angezeigten Navigationspunkte immer an der richtigen Stelle platziert sind. Diese Anforderung ist im Allgemeinen erfüllt. Allerdings kann es manchmal zu Störungen der Sensoren oder bei den Wifi-Zugangspunkten kommen. Dies sind jedoch kaum vermeidbare Fälle, die normalerweise eher selten sind.
Auch die Wahl von Start und Ziel ist möglich. Der entsprechende Dialog bietet zwei Textfelder an, in denen Start und Ziel festgelegt werden können. Die einzige Einschränkung hierbei ist, dass nur Räume als Start und Ziel verwendet werden können. Um bei der Auswahl zu helfen, gibt es ein Dropdown-Menü zur Autovervollständigung.
Alle Anforderungen sind in Tabelle 6.1 nocheinmal aufgeführt.

Insgesamt sind also alle Anforderungen erfüllt. Trotzdem gibt es einige Punkte die noch verbessert oder erweitert werden könnten. Mehr dazu wird im Kapitel “Ausblick” erläutert.

Tabelle 6.1: Bewertung der Anforderungen in Tabellenform

Anforderung	Typ der Anforderung	Bewertung
Hohe Geschwindigkeit des Prototyps	nicht-funktional	++
Kurze Wartezeiten	nicht-funktional	+
Einfache Bedienung	nicht-funktional	++
Routenführung mittles AR	funktional	-+
Navsymbole mit nötigen Informationen	funktional	+
Interaktion mit Navsymbolen gibt zusätzliche Informationen	funktional	+
Hohe Präzision bei der Routenführung	nicht-funktional	-
Bei Bewegung wird die Position der Punkte aktualisiert	funktional	0
Wahl des Startpunktes	funktional	++
Wahl des Zielpunktes	funktional	++
Hohe Stabilität	nicht-funktional	0

7

Vorstellung der App

Die in dieser Arbeit entwickelte Hybridapp aus AREA und Wifinder übernimmt zu großen Teilen auch die Benutzeroberflächen der beiden Apps. Bei AREA ist dies im Wesentlichen die Kameraansicht, in der auch die Augmented-Reality Informationen, also die Navigationspunkte, eingeblendet werden (vgl. Abbildung 7.1). Zu dieser Kameraansicht gehört auch ein Radar, das als kleine Übersicht über alle angezeigten Punkte fungiert. Dieses Radar ist auch weiterhin in der Oberfläche vorhanden, da es durchaus hilfreich bei der Navigation ist, eine Übersicht über alle verfügbaren Punkte, und deren relative Position zueinander, zu haben.
Die erste wirkliche Änderung ist das Wegfallen des Menüs für die Einstellung der Sichtweite des Radars. Mit dieser Einstellung kann in AREA festgelegt werden bis zu welcher Entfernung Point-of-Interests in der Augmented-Reality Ansicht dargestellt wurden. Da der hier entwickelte Prototyp nur für den Innenbereich und für Navigationszwecke entwickelt wurde, ist es nicht notwendig, dass diese Einstellung verändert werden kann. Es

Abbildung 7.1: Startansicht im AR-Modus.

reicht aus, wenn es hier einen festen Wert von 500m - 1000m Sichtweite gibt.
Ein weiterer neuer Aspekt der Augmented-Reality Benutzeroberfläche ist das Hinzufügen eines Dialogs zur Auswahl einer Route. Der Dialog kann über einen Button am unteren Bildschirmrand erreicht werden und ermöglicht das Eingeben des gewünschten Start- und Zielpunktes für eine Navigation mit Hilfe von Augmented-Reality. Der Dialog selbst ist identisch mit dem entsprechenden Dialog der Wifinder App (vgl. Abblidung 7.2).
Nachdem ein Routenwunsch eingegeben und die Antwort vom Server verarbeitet wurde, werden die Navigationspunkte, wie in Abbildung 7.3 zu sehen ist, in der Kameraansicht eingeblendet. Dabei werden Navigationspunkte abhängig von ihrer Distanz zum Benutzer in verschiedenen Größen angezeigt. Je weiter ein Punkt vom Benutzer entfernt ist

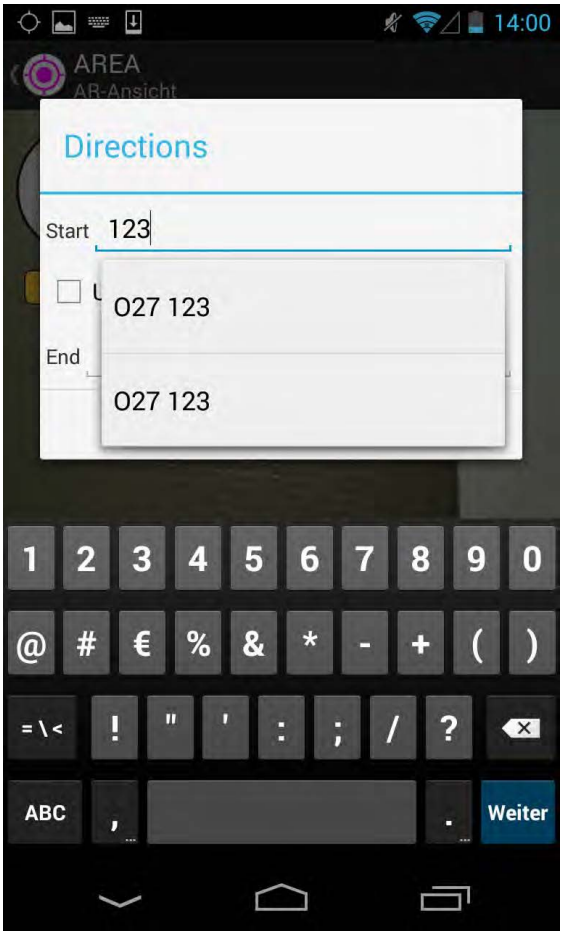

Abbildung 7.2: Dialog zur Auswahl der Route.

desto kleiner wird er dargestellt, und je näher ein Punkt beim Benutzer liegt desto größer wird er dargestellt.

Klickt man auf einen der Wegpunkte, bekommt man zusätzliche Informationen, wie Nummer und Stockwerk angezeigt (vgl. Abbildung 7.4). Dies ist vor allem zur Überprüfung der Richtigkeit der Anzeige nützlich.

Die letzte Änderung ist ein Button oben links in der Actionbar, der den Zugriff auf die Oberfläche der Wifinder-App ermöglicht. Die ursprüngliche Version der AREA-Oberfläche ist in der Arbeit von Philip Geiger [Gei12] zu finden. Dieser Zugang zur Wifinder Oberfläche wird nicht zwingend benötigt und sollte in einer späteren, finalen Version nicht mehr vorhanden sein. Da dies jedoch ein Prototyp ist und es zu Testzwecken nützlich ist,

Zugriff auf die Wifinder Oberfläche und Funktionalitäten zu haben, wurde dieser Zugang beibehalten.
Die Oberfläche von Wifinder ist in drei Tabs unterteilt, die über die Actionbar am oberen Rand des Bildschirm zu erreichen sind. In allen drei Tabs ist am unteren Rand des Bildschirms eine Leiste mit Funktionsbuttons. Einer dieser Buttons ist der Einstellungsbutton, der immer verfügbar ist. Hier können Einstellungen zur URL des Wifinder Servers gemacht werden. Außerdem kann hier der Fingerprinting-Modus ein- und ausgeschaltet werden. Die restlichen Buttons unterscheiden sich in jedem Tab.
Im ersten Tab, der “AP LIST” heißt, kann man sich, nach Berühren der Lupe unten links, eine Liste der empfangbaren Wifi-Zuganspunkte mit ihrer SSID und der zugehörigen Signalstärke anzeigen lassen. Dies sind die Informationen, die beim Fingerprinting gesammelt werden (vgl. Abbildung 7.5).
Im “MAP”-Tab (vgl. Abbildung 7.6) befindet sich die Kartenansicht. Hier kann mit einem Klick auf das Fadenkreuz unten links, die aktuelle Position angezeigt werden oder, durch Klick auf den Pfeil-Button, eine Route von einem beliebigen Startpunkt zu einem beliebigen Zielpunkt berechnet werden. Die Route wird dann auf der Kartenansicht angezeigt. Der nächste Button in der Reihe ermöglicht die Auswahl eines Stockwerkes. Da es in einem Gebäude meistens mehrere Stockwerke gibt, ist dies notwendig, wenn man sich den Verlauf einer gesamten Route oder auch einfach nur den Plan eines bestimmten Stockwerkes ansehen möchte. Der nächste Menüpunkt heißt “AR VIEW” und ermöglicht, wie der Name schon sagt, den Zugriff auf die Augmented-Reality Kameraansicht.
Der dritte Tab enthält die “WIFIMAP” (vgl. Abbildung 7.7). Auch hier wird eine Karte des Gebäudes angezeigt und auch hier kann über ein Menü am unteren Rand ein Stockwerk ausgewählt werden. Diese Karte dient jedoch nicht der Navigation, sondern der Administration. In der Mitte der Karte wird ein blauer Punkt angezeigt. Durch Verschieben der Karte kann der blaue Punkt so platziert werden, dass er der Position des Mobilen Gerätes entspricht. Dann kann über einen Button unten Links der Wifi-Fingerabdruck an den Wifi-Server übermittelt werden. Auf diese Weise kann ein Gebäude nach und nach kartographiert werden. Speziell diese Funktion der Wifinder Oberfläche ist beim testen des neuen Prototyps wichtig, da hier der Navigationsgraph verändert werden kann.
Außer dem zusätzlichen Button für das Aufrufen der Augmented-Reality Kameraansicht

gibt es keine Veränderungen an der Oberfläche, wie sie von Alexander Bachmeier in seiner Arbeit [Bac13] entwickelt wurde.

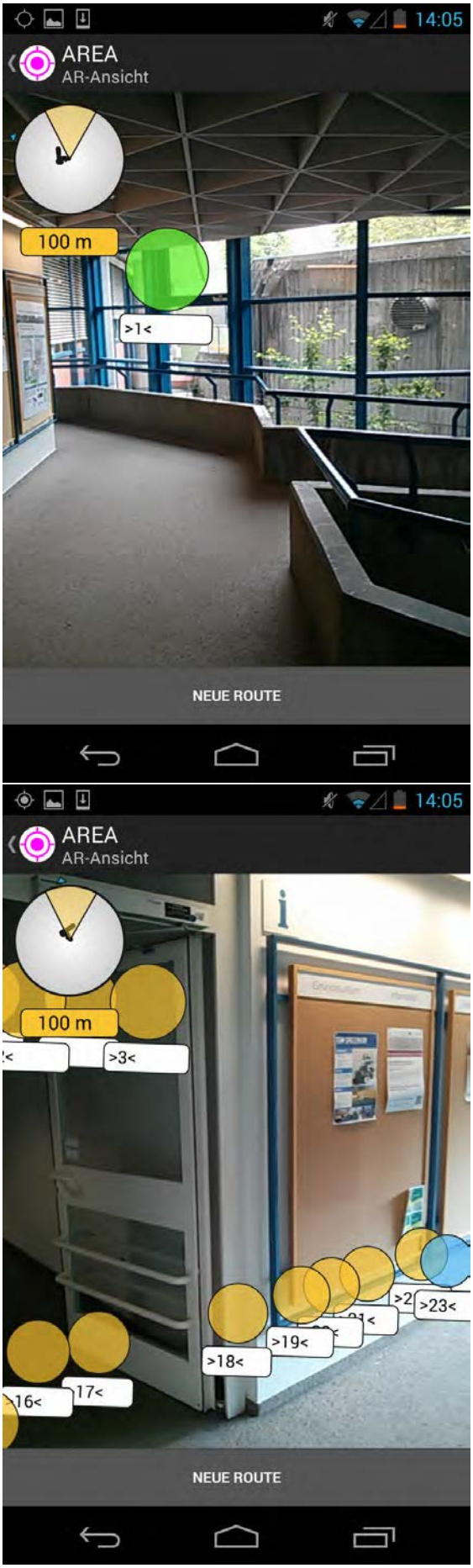

Abbildung 7.3: Erster Wegpunkt(grün), Routenpunkte (gelb) und Zielpunkt (blau) auf einer Route von H20 nach O27 123.

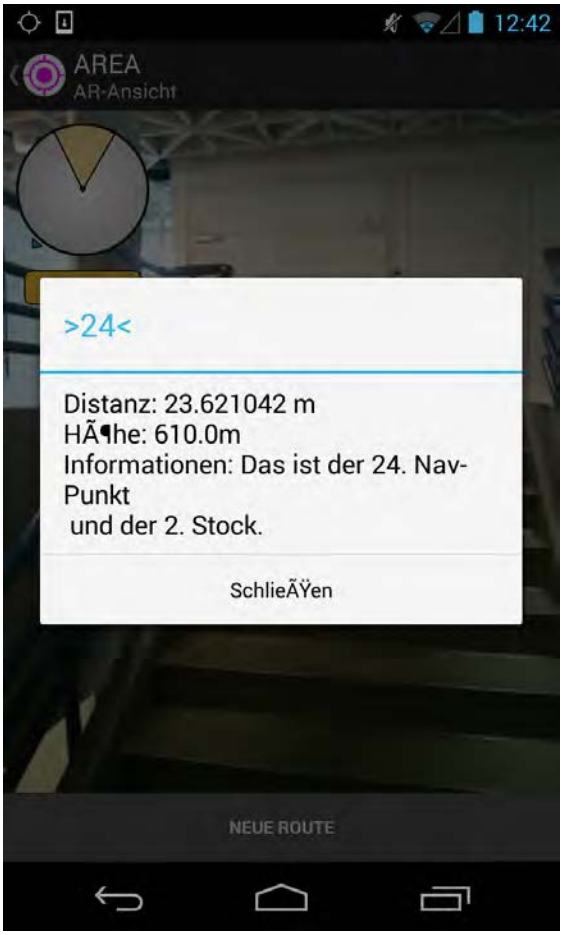

Abbildung 7.4: Nähere Informationen zu einem Wegpunkt.

Abbildung 7.5: Ansicht des “AP List” Tabs.

Abbildung 7.6: Kartenansicht der Wifinder Oberfläche.

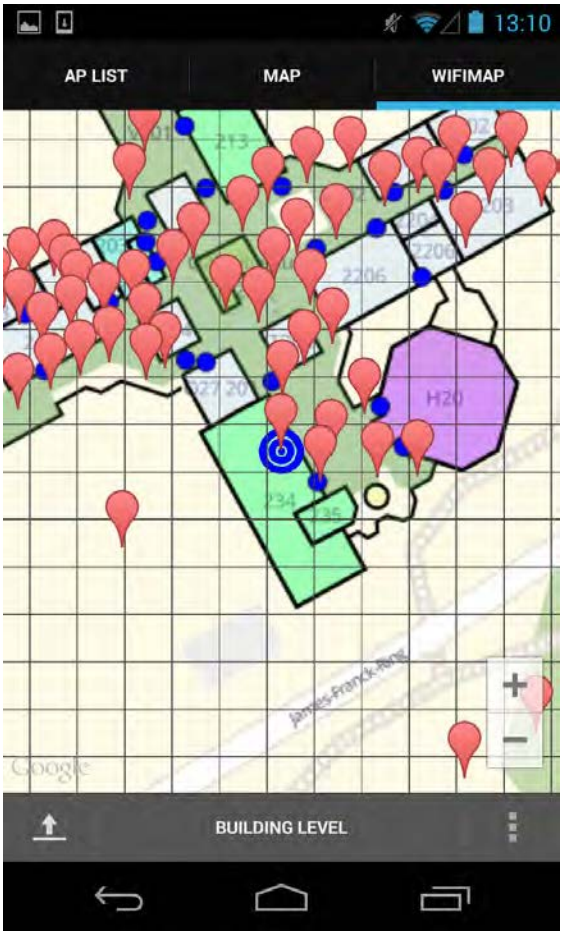

Abbildung 7.7: Wifimap zum anlegen von WiFi-Fingerabdrücken.

8

Zusammenfassung

Das Ziel dieser Arbeit war es, eine In-House-Augmented-Reality- Navigationsanwendung, unter Verwendung der App AREA als Augmented-Reality Engine und der App Wifinder als Positionierungs- und Routing-Engine, zu entwickeln. Dazu wurden zunächst einige allgemeine Grundlagen zu Augmented Reality und zu den beiden verwendeten Apps erläutert. Es wurde klar, dass Wifinder innerhalb von Gebäuden GPS ersetzen kann, und dadurch auch in Bereichen ohne GPS ein funktionierendes standortbasiertes Augmented-Reality Verfahren, wie es von AREA verwendet wird, ermöglicht wird. Wie sich in dieser Arbeit herausgestellt hat, ergänzen sich AREA und Wifinder in dem hier entwickelten Prototypen sehr gut.
Es wurden dann Anforderungen aufgestellt, die möglichst vollständig von dem hier entwickelten Prototypen erfüllt werden sollten. Die Anforderungen leiteten sich teilweise von den Anforderungen an AREA und Wifinder ab. Andererseits gab es auch neue Anforderungen die vor allem an die Augmented-Reality Navigation gestellt wurden.

Hiernach wurde die Architektur und verwendete Technologien erläutert. Es wurde vor allem der verwendete Wifinder-Server vorgestellt und seine Bedeutung für den Prototypen erklärt. Es wurden die verwendeten Technologien, die zur Kommunikation zwischen Client und Server, und die bei den Datenflüssen innerhalb der Clientanwendung zum Einsatz kommen, beschrieben und erklärt. Auch die wichtigsten Aspekte der neuen Architektur des Prototyps wurden gezeigt und erläutert.
Bei der tatsächlichen Implementierung gab es vor allem drei spezielle Bereiche die besonders wichtig sind:

- Die Konvertierung von Datentypen, damit AREA und Wifinder kommunizieren können.
- Das Einschleusen der von Wifinder ermittelten Position in die AREA-Engine.
- Die Feststellung und Anzeige der Route in der Augmented-Reality-Kameraansicht.

Diese drei Bereiche wurden ausführlich erläutert, da sie den Kern des neuen Prototypen bilden. Besondere Schwierigkeiten bei der Implementierung machte vor allem die große Menge an fremdem und komplexem Code. Es wurde viel Zeit darauf verwendet, sowohl die Funktionsweise von AREA als auch die von Wifinder zu verstehen, damit eine Zusammenführung von beiden überhaupt möglich war.
In einer anschließenden Überprüfung und Bewertung der Anforderungen wurde klar, dass der Prototyp die meisten, der an ihn gestellten Anforderungen erfüllen kann. Es wurde jedoch auch schon an früherer Stelle klar, dass dieser Prototyp noch einiges an Verbesserungspotential besitzt. Wie diese Verbesserungen aussehen könnten, soll jetzt im “Ausblick” aufgelistet werden.

9

Ausblick

Die hier entwickelte In-House-Augmented-Reality App hat noch einige Möglichkeiten verbessert zu werden.
Zur Verbesserung der Navigation könnten zum Beispiel zusätzlich zu den Navigationspunkten auch Points of Interest angezeigt werden, um dem Benutzer eine bessere Vorstellung von seiner aktuellen Position zu geben. Außerdem sind Points of Interest auch innerhalb eines Gebäudes meistens leicht zu erkennen und können so dem Benutzer noch mehr Informationen liefern.
Ein weiterer Aspekt ist das GPS. Da die Lokalisierung über Wifinder nicht immer optimal ist und es manchmal auch innerhalb eines Gebäudes möglich ist, ein GPS-Signal zu empfangen, wäre es sinnvoll, wenn auch diese GPS-Signale zur Navigation verwendet werden könnten. Dies ist vor allem dann sinnvoll, wenn es in dem Gebäude viele Außenbereiche, wie zum Beispiel an einer Universität, gibt. In diesen Außenbereichen gibt es normalerweise kein Wifi-Netz, dafür aber GPS, das dann natürlich auch genutzt werden

sollte.
Eine Verbesserung der Augmented-Reality Darstellung und Routenführung könnte erreicht werden, wenn man das gesamte Navigationssystem von Wifinder verändert. Dazu müsste man, statt nur einer “Navigationspunkteklasse”, mehrere Klassen einführen. Es gäbe auch weiterhin normale Navigationspunkte, die aber nie in der Augmented-Reality Ansicht angezeigt würden. Zusätzlich gäbe es spezielle Punkte, die zur Anzeige verwendet würden. Diese Punkte würden Türen, Räume und Treppen bezeichnen. Durch die Beschränkung auf die Anzeige dieser Punkte, kann die Menge der angezeigten Punkte stark reduziert werden, ohne dabei die Navigationsleistung zu vermindern.
Der letzte verbesserungswürdige Punkt ist das Abtrennen der administrativen Wifinder Oberfläche von der eigentlichen Augmented-Reality Navigationsoberfläche. Dies wurde nur deshalb hier noch nicht gemacht, weil diese hybride App noch ein Prototyp ist und es häufig notwendig ist, die Navigationspunkte zu Testzwecken zu verändern.
Ein weiterer, vielversprechender Ansatz in diesem Kontext ist der Einsatz von Prozess-Management-Technologie. Diese muss dann besonders vor dem mobilen Hintergrund betrachtet werden [PMR13], [PTKR10], [PTR10].

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[adt] *Android Developer Tools*. http://developer.android.com/sdk/index.html, . – Accessed: 2014-03-24

[ASP99] AOKI, Hisashi ; SCHIELE, Bernt ; PENTLAND, Alex. Realtime Personal Positioning System for Wearable Computers. In: *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1999 (ISWC '99). – ISBN 0–7695–0428–0, 37–

[Bac13] BACHMEIER, Alexander: *WI-FI based indoor navigation in the context of mobile services*, Universität Ulm, Masterarbeit, 2013

[cal] *Rückruffunktion*. http://de.wikipedia.org/wiki/R%C3%BCckruffunktion, . – Accessed: 2014-04-05

[CNB+13] CROMBACH, Anselm ; NANDI, Corina ; BAMBONYE, Manassé ; LIEBRECHT, Martin ; PRYSS, Rüdiger ; REICHERT, Manfred ; ELBERT, Thomas ; WEIERSTALL, Roland. Screening for mental disorders in post-conflict regions using computer apps - a feasibility study from Burundi. In: *XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference*, 2013, S. 70–70

[Gei12] GEIGER, Philip: *Entwicklung einer Augmented Reality Engine am Beispiel des iOS*, Universität Ulm, Bachelorarbeit, 2012

[GPSR13] GEIGER, Philip ; PRYSS, Rüdiger ; SCHICKLER, Marc ; REICHERT, Manfred. Engineering an Advanced Location-Based Augmented Reality Engine for

Smart Mobile Devices / University of Ulm. Ulm : University of Ulm, October 2013 (UIB-2013-09). – Technical Report

[GSP+14] GEIGER, Philip ; SCHICKLER, Marc ; PRYSS, Rüdiger ; SCHOBEL, Johannes ; REICHERT, Manfred. Location-based Mobile Augmented Reality Applications: Challenges, Examples, Lessons Learned. In: *10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps*, 2014, S. 383–394

[IRLP+13] ISELE, Dorothea ; RUF-LEUSCHNER, Martina ; PRYSS, Rüdiger ; SCHAUER, Maggie ; REICHERT, Manfred ; SCHOBEL, Johannes ; SCHINDLER, Arnim ; ELBERT, Thomas. Detecting adverse childhood experiences with a little help from tablet computers. In: *XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference*, 2013, S. 69–70

[JBKD06] JONG-BAE KIM ; DMITRC, South Korea ; Jeong-Mi Lee ; Hee-Sung J. Ulsan Univ. U. Ulsan Univ. AR-based indoor navigation system for personal locating. In: *Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on*, 2006. – ISBN 0–7803–9459–3, 217–218

[jun] *Junaio*. `http://www.youtube.com/watch?v=WciFx66ojA4`, . – Accessed: 2014-04-05

[LCH11] LIN, Chi-Yi ; CHANG, Chih-Yuan ; HSU, Hui-Huang. A RFID-Based Personal Navigation System for Multi-story Indoor Environments. In: *Proceedings of the 2011 International Conference on Broadband and Wireless Computing, Communication and Applications*. Washington, DC, USA : IEEE Computer Society, 2011 (BWCCA '11). – ISBN 978–0–7695–4532–5, 460–465

[ope] *OpenCV*. `http://opencv.org/`, . – Accessed: 2014-03-26

[osm] *OpenStreetMap*. `http://www.openstreetmap.de/`, . – Accessed: 2014-03-24

[PLRH12] PRYSS, Rüdiger ; LANGER, David ; REICHERT, Manfred ; HALLERBACH, Alena. Mobile Task Management for Medical Ward Rounds - The MEDo

Approach. In: *1st Int'l Workshop on Adaptive Case Management (ACM'12), BPM'12 Workshops*, Springer, September 2012 (LNBIP 132), S. 43–54

[PMLR14] PRYSS, Rüdiger ; MUNDBROD, Nicolas ; LANGER, David ; REICHERT, Manfred. Supporting medical ward rounds through mobile task and process management. In: *Information Systems and e-Business Management* (2014), March

[PMR13] PRYSS, Rüdiger ; MUSIOL, Steffen ; REICHERT, Manfred. Collaboration Support Through Mobile Processes and Entailment Constraints. In: *9th IEEE Int'l Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'13)*, IEEE Computer Society Press, October 2013

[PTKR10] PRYSS, Rüdiger ; TIEDEKEN, Julian ; KREHER, Ulrich ; REICHERT, Manfred. Towards Flexible Process Support on Mobile Devices. In: *Proc. CAiSE'10 Forum - Information Systems Evolution*, Springer, 2010 (LNBIP 72), S. 150–165

[PTR10] PRYSS, Rüdiger ; TIEDEKEN, Julian ; REICHERT, Manfred. Managing Processes on Mobile Devices: The MARPLE Approach. In: *CAiSE'10 Demos*, 2010

[RLPL+13] RUF-LEUSCHNER, Martina ; PRYSS, Rüdiger ; LIEBRECHT, Martin ; SCHOBEL, Johannes ; SPYRIDOU, Andria ; REICHERT, Manfred ; SCHAUER, Maggie. Preventing further trauma: KINDEX mum screen - assessing and reacting towards psychosocial risk factors in pregnant women with the help of smartphone technologies. In: *XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference*, 2013, S. 70–70

[RPR11] ROBECKE, Andreas ; PRYSS, Rüdiger ; REICHERT, Manfred. DBIScholar: An iPhone Application for Performing Citation Analyses. In: *CAiSE Forum-2011*, CEUR Workshop Proceedings, June 2011 (Proceedings of the CAiSE'11 Forum at the 23rd International Conference on Advanced Information Systems Engineering Vol-73)

[RS03] REITMAYR, Gerhard ; SCHMALSTIEG, Dieter. Location Based Applications for Mobile Augmented Reality. In: *Proceedings of the Fourth Australasian User Interface Conference on User Interfaces 2003 - Volume 18*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2003 (AUIC '03). – ISBN 0–909925–96–8, 65–73

[SRLP+13] SCHOBEL, Johannes ; RUF-LEUSCHNER, Martina ; PRYSS, Rüdiger ; REICHERT, Manfred ; SCHICKLER, Marc ; SCHAUER, Maggie ; WEIERSTALL, Roland ; ISELE, Dorothea ; NANDI, Corina ; ELBERT, Thomas. A generic questionnaire framework supporting psychological studies with smartphone technologies. In: *XIII Congress of European Society of Traumatic Stress Studies (ESTSS) Conference*, 2013, S. 69–69

[SSP+13] SCHOBEL, Johannes ; SCHICKLER, Marc ; PRYSS, Rüdiger ; NIENHAUS, Hans ; REICHERT, Manfred. Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps*, 2013, S. 509–518

[SSP+14] SCHOBEL, Johannes ; SCHICKLER, Marc ; PRYSS, Rüdiger ; MAIER, Fabian ; REICHERT, Manfred. Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: *10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps*, 2014, S. 371–382

[vuf] *Vuforia SDK*. `https://developer.qualcomm.com/mobile-development/add-advanced-features/augmented-reality-vuforia`, . – Accessed: 2014-03-26

Name: Tobias Waldenmaier Matrikelnummer: 680544

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Tobias Waldenmaier