ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Datenbanken und Informationssysteme

Konzeption und Entwicklung eines Location based Features zur Unterstützung von Mobile Augmented Reality Applications

Abschlussarbeit an der Universität Ulm

Vorgelegt von:
Lucas Tilmann Gmünder
lucas.gmuender@uni-ulm.de
868101

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Rüdiger Pryss

2019

Fassung 21. März 2019

Satz: PDF-LaTeX 2$_\varepsilon$

Inhaltsverzeichnis

1 Motivation und Zielsetzung **1**

1.1 Motivation 1

1.2 Zielsetzung 1

1.3 Aufbau der Arbeit 2

2 Grundlagen zu Augmented Reality und Augmented Reality Engine Application **3**

2.1 Mobile Augmented Reality 3

2.1.1 Marker-based Augmented Reality 3

2.1.2 Markerless Augmented Reality 4

2.1.3 Location-based Augmented Reality 4

2.2 Global Positioning System und Positionserfassung 4

2.2.1 Global Positioning System 4

2.2.2 Standortermittlung über WLAN oder mobile Netzwerkprovider 5

2.3 Augmented Reality Engine Application Projekt 6

3 Verwandte Arbeiten **7**

4 Anforderungen **9**

4.1 Funktionale Anforderungen 9

4.2 Nicht-funktionale Anforderungen 10

5 Architektur **11**

6 Implementierungsaspekte **15**

6.1 Karte und Zone 15

6.2 Positionsbestimmung 18

7 Vorstellung der Anwendung **21**
7.1 AREA Grundfunktionen 21
7.2 AREA Zone Funktion 27

8 Anforderungsabgleich **33**
8.1 Funktionale Anforderungen 33
8.2 Nicht-funktionale Anforderungen 34

9 Zusammenfassung und Ausblick **35**
9.1 Zusammenfassung 35
9.2 Ausblick 35

Literatur **37**

1 Motivation und Zielsetzung

1.1 Motivation

Sowohl Virtual Reality als auch Augmented Reality sind Technologien, die immer mehr an Bedeutung und Bekanntheit gewinnen. Vor allem im alltäglichen Leben finden sie immer mehr Anwendung. Auch Smartphones finden sich heutzutage in fast jeder Tasche wieder und werden mit jedem neuen Modell immer leistungsstärker. Die Berechnungen, welche Augmented Reality möglich machen, sind somit schon auch auf Mobilen Endgeräten ohne weiteres möglich. Diese Kombination von Mobilität und Technologie bietet zahlreiche Anwendungsfälle. Einige dieser Anwendungsfälle können bereits unter Zuhilfenahme des Augmented Reality Engine Application Frameworks abgedeckt werden [1]. Dieses Framework, um ein location-based Feature zu erweitern, bietet spannende Möglichkeiten und neue Perspektiven im Hinblick auf Mobile Augmented Reality Applications.

1.2 Zielsetzung

Ziel dieser Arbeit ist es ein Feature für Mobile Augmented Reality Applications zu konzeptionieren und zu entwickeln. Unter Verarbeitung positionsbezogener Daten soll die Möglichkeit geschaffen werden, bestimmte Informationen und Augmented Reality Ansichten mit vordefinierten Zonen zu verbinden. Somit kann auf einer Karte in diesen Zonen ganz bestimmtes, genau auf die Lokalität der Zone angepasstes, Augmented Reality Material angezeigt werden. Des weiteren soll eine Möglichkeit implementiert werden, den location-based Augmented Reality Ansatz, welcher bereits in dem oben genannten Framework zum Einsatz kommt, auch in komplexeren Umgebungen nutzen zu können.

1.3 Aufbau der Arbeit

Im Grunde besteht die Arbeit aus acht Teilen, zuerst werden in Kapitel 2 wesentliche Grundlagen zu Mobile Augmented Reality und dem Augmented Reality Engine Application Projekt erläutert. Anschließend beinhaltet Kapitel 3 verwandte Arbeiten und in Kapitel 4 werden die Anforderungen an das System definiert. Im Folgenden geht es um die Architektur und die Implementierungsaspekte des Systems in Kapitel 5 und 6. Darauf aufbauend wird in 7 ein Umfassender Überblick über das System gegeben. In Kapitel 8 werden die Spezifikationen des Systems mit den Anforderungen abgeglichen, bevor in Kapitel 9 eine Zusammenfassung der Arbeit und ein Ausblick auf mögliche zukünftige Arbeiten gegeben wird.

2 Grundlagen zu Augmented Reality und Augmented Reality Engine Application

In diesem Kapitel geht es um die Klärung notwendiger Grundlagen für diese Arbeit.

2.1 Mobile Augmented Reality

Mobile Augmented Reality (MAR) bezeichnet eine Unterordnung der Augmented Reality(AR), also der „erweiterten Realität“. Hierbei wird ein von einer Kamera aufgenommenes Material, um zusätzliche digitale Informationen ergänzt bzw. erweitert. Dies ist nicht zu verwechseln mit Virtual Reality oder auch „virtueller Realität “, bei der die komplette Darstellung der Realität digital erzeugt wird. MAR beschreibt dieses Konzept auf mobilen Geräten, welches bereits 1968 seine Anfänge verzeichnete [2]. MAR lässt sich in drei Unterkategorien aufgliedern und zwar in „marker-based Augmented Reality “ , „markerless Augmented Reality “ und „location-based Augmented Reality “.

2.1.1 Marker-based Augmented Reality

Wenn wir von marker-based-AR sprechen, ist gemeint dass, um digitale Informationen mit einem bestimmten Teil der Realität zu verknüpfen, ein Anker in der echten Welt benötigt wird. Diese Anker gibt es in vielen Formen und können quasi alles sein, was von der Kamera des Systems erkannt werden kann und vorher definiert wird. Beispiele wären Geometrische Formen, Körper oder QR Codes.

Wenn dieser Anker vom System erkannt wird, können die anzuzeigenden Informationen mit diesem Anker verknüpft werden und sind somit automatisch korrekt im Raum platziert.

2.1.2 Markerless Augmented Reality

Bei markerless AR hingegen ist es weniger wichtig, dass die Informationen an einem bestimmten und festen Platz im Raum platziert sind, wenn es beispielsweise nur darum geht dreidimensionale Modelle darzustellen. Hierbei geht es darum, den Nutzer entscheiden zu lassen, wo im Raum sich das Modell befindet, beispielsweise zu Vorschau Zwecken.

2.1.3 Location-based Augmented Reality

Bei location-based AR wird ein komplett anderer Ansatz benutzt. Hier bestimmt die aktuelle Position von Nutzer und Gerät, welche Informationen sich wie im Raum befinden. Dabei ist es von entscheidender Wichtigkeit einen zuverlässigen Standortdienst zu nutzen, um die Genauigkeit der angezeigten Informationen zu gewährleisten. Hierbei sind Positionsdaten von mobilen Netzwerken sowie GPS weit verbreitet. Das dieser Arbeit zugrunde liegende System AREA verwendet diesen Ansatz.

2.2 Global Positioning System und Positionserfassung

2.2.1 Global Positioning System

„Navigational Satellite Timing and Ranging – Global Positioning System“ allgemein bekannt unter dem Kürzel „GPS “ ist ein vom US-Verteidigungsministerium entwickeltes globales Navigationssatellitensystem zur Positionsbestimmung [3]. Dieses System verwendet Signale von GPS-Satelliten in der Erdumlaufbahn, welche konstant ihre Positionsdaten und die genaue Uhrzeit ausstrahlen.

GPS-Empfänger, beispielsweise mobile Endgeräte wie Smartphones, können durch die Laufzeit der Signale, und anschließende Trilateration dieser, die genaue Position bis auf 5-15 Meter genau bestimmen. Trilateration beschreibt ein Vorgehen ähnlich der weitreichend geläufigeren Triangulation. Während bei der Triangulation die Position durch Vermessung von drei Winkeln bestimmt wird, werden bei der Trilateration für den selben Zweck drei Abstände bzw. Entfernungsmessungen zu Fixpunkten verwendet. Noch genauer lässt sich die Position nur mithilfe fester Empfangsstationen ermitteln, welche ein Korrektursignal senden. Um die genaue Position und Zeit bestimmen zu können, werden die Signale von vier Satelliten gleichzeitig benötigt. Die sechs verschiedenen Erdumlaufbahnen, auf denen sich die Satelliten bewegen, sind um 55° gegenüber der Äquatorebene geneigt und so bestückt, dass zu jeder Zeit von jeder Position auf dem Planeten mindestens vier Satelliten zu sehen sind. Im allgemeinen Fall bauen moderne GPS-Empfänger jedoch mit mindestens 6-12 Satelliten eine Verbindung auf um Messfehler und Störungen zu vermeiden, und eine möglichst genaue Positionsberechnung zu gewährleisten [4]. Die Signale der GPS Satelliten, also die elektromagnetische Strahlung im von GPS Sendern verwendeten Frequenzspektrum, breiten sich ähnlich wie sichtbares Licht nahezu geradlinig aus. Um bestmögliche Empfangsergebnisse zu erzielen, ist eine direkte Sichtverbindung vom Empfänger zum Satelliten optimal. Daher ist es in Gebäuden nahezu unmöglich ein korrektes GPS Signal zu empfangen. Sollten solche Signale jedoch aufgrund von Reflektion doch empfangen werden, sind die Daten meist sehr fehlerhaft und unbrauchbar. Auch in der Nähe von höheren Gebäuden können durch Mehrfachreflektion starke Ungenauigkeiten der empfangenen Signale Auftreten.

2.2.2 Standortermittlung über WLAN oder mobile Netzwerkprovider

In aktuellen Smartphones bietet sich noch eine zweite Methode der Standortbestimmung, und zwar die Positionsbestimmung über WLAN Netzwerke in der Umgebung des Nutzers. Viele öffentlich zugängliche WLAN Router sind mit genauen Positionsinformationen bestückt und können diese zusammen mit dem WLAN Signal senden. Mobile Endgeräte wie Smartphones können also auch hier durch das Empfangen mehrerer WLAN Signale die eigene Position bestimmen.

Dies funktioniert ähnlich wie beim GPS über Trilateration der Signale der verschiedenen Router. Sobald also das Gerät mehrere Netzwerke wahrnimmt, es muss keine Verbindung vorhanden sein, kann relativ zu den Signalen der aktuelle Standort bestimmt werden. Im Gegensatz zu GPS funktioniert WLAN basierte Ortung auch in Gebäuden zuverlässig, abhängig davon wie genau der Standort der Router angegeben ist und gesendet wird. Ähnlich wie beim WLAN funktioniert das selbe Prinzip auch mit Informationen von Sendemasten mobiler Netzwerkprovider.

2.3 Augmented Reality Engine Application Projekt

Augmented Reality Engine Application(AREA) ist ein Forschungsprojekt des Instituts für Datenbanken und Informationssysteme der Universität Ulm. Es geht darum, ein zeitgemäßes MAR Kernel für mobile Endgeräte auf IOS- und Android-Basis zur Verfügung zu stellen, mit dem sich flexibel location-based AR Applikationen entwickeln lassen. Bisher ermöglicht AREA es dem Nutzer Interessenspunkte, auch Point of Interests(POI) genannt, in seiner Umgebung, relativ zu seinem Standpunkt, positionsgenau und in korrektem Betrachtungswinkel anzuzeigen. Dieses Prinzip wird durch das Zusammenfassen von Interessenspunkten zu Clustern unterstützt. Ebenso ist es möglich Pfade zu Interessenspunkten anzuzeigen. Der größte Vorteil von AREA ist die modulare Architektur, welche es ermöglicht, auf Basis des AREA Kernels flexibel eigene MAR Applikationen zu entwickeln, welche sich durch effiziente CPU- und möglichst geringfügige Batterie-Nutzung auszeichnen [1, 5].

3 Verwandte Arbeiten

In diesem Kapitel wird ein kurzer Überblick über verwandte Arbeiten gegeben.

Grundsätzlich muss hier das AREA Projekt als Ganzes genannt werden, welches durch Philipp Geiger in [6] begonnen wurde und in [7] veröffentlicht wurde. Erkenntnisse dieser Arbeiten im Bezug auf location-based mobile AR Systeme sind in [8] beschrieben. Verbesserungen des Grundkonzepts und dessen Algorithmen für intelligente mobile AR Systeme sind in [9] thematisiert. Auch [5] ist hier zu erwähnen, in dem die Track Funktion, oder zu Deutsch Pfad Funktion, von AREA behandelt wird. Der aktuelle Stand von AREA vor dieser Arbeit wird in [1] beschrieben.

In [10] geht es darum, geplante Gebäude mithilfe von AR darzustellen. Es wird ein Konzept sehr ähnlich zu der in dieser Arbeit entwickelten Funktion beschrieben. Auch hier werden zur Darstellung der Gebäude Zonen benutzt, in die sich der Benutzer zu begeben hat, um ihm eine optimale Sicht auf das Gebäude in der AR zu ermöglichen. In dieser Arbeit steht der Nutzer im Mittelpunkt, es wird ein User Centered Design Ansatz benutzt und bereits während der Entwicklung des Systems viel mit Testnutzern gearbeitet. Durch zahlreiche Nutzerstudien zur Entwicklungszeit mit Prototypen wurde immer weiter an der Nutzererfahrung gearbeitet. Dabei wurde versucht, dem Nutzer das Thema MAR, welches sehr technologiegesteuert ist, so einsteigerfreundlich wie möglich zu machen. Dadurch sollte es sich von Systemen abheben, die erst im letzten Entwicklungsschritt an Nutzerfreundlichkeit und Bedienbarkeit denken.

Ein sehr ähnliches Prinzip ist auch in dem 2016 erschienenen MAR Spiel Pokemon Go vertreten. In diesem Spiel wird ebenfalls der location-based Ansatz verwendet, um bestimmte AR Inhalte nur in definierten Zonen verfügbar zu machen. Vor allem zeigt die Beliebtheit dieses Spiels, dass Nutzer location-based AR durchaus nicht abgeneigt sind und in dieser Technologie viel Potential steckt [11].

In der mobilen Spielebranche ist der location-based Ansatz aber auch in weniger bekannten Veröffentlichungen durchaus weiter verbreitet und wurde in Spielen wie Ingress [12] und Harry Potter Wizards Unite [13] verwendet.

4 Anforderungen

Im folgenden Kapitel werden funktionale und nicht-funktionale Anforderungen an das zu entwickelnde Feature definiert.

4.1 Funktionale Anforderungen

Nr.	Titel und Beschreibung der funktionalen Anforderung
1	**GPS in komplexen Umgebungen** Sowohl für POI's in der Grunfunktion von AREA, als auch für die Objekt AR Ansicht soll eine Möglichkeit gefunden werden, um auch in komplexen Umgebungen wie innerhalb von Gebäuden oder in der Nähe großer Gebäude korrekt zu funktionieren.
2	**Anzeigen einer Karte** Es muss möglich sein vom AREA Hauptbildschirm auf eine Karte zu wechseln, welche dem Nutzer die aktuelle Umgebung anzeigt.
3	**Nutzerposition auf der Karte** Beim Öffnen der Karte soll der Nutzer seine aktuelle Position in der Mitte des Kartenbildschirms auffinden.
4	**Nutzermanipulation der Karte** Es muss dem Nutzer möglich sein die Karte zu verschieben und heraus oder hinein zu zoomen.
5	**Anzeigen der Zonen auf der Karte** Auf der Karte sollen alle im Umkreis des Nutzers befindlichen Zonen klar unterscheidbar vom Rest der Karte angezeigt werden.

6	**Wechseln in die AR Ansicht** Nur wenn sich der Nutzer in einer der angezeigten Zonen befindet, darf die Möglichkeit bestehen in die AR Ansicht zu wechseln.
7	**Ausblendung entfernter Zonen** Zonen, welche sich nicht im maximalen Radius von AREA befinden, also zwei Kilometern, sollen nicht angezeigt werden.
8	**Zusätzliche Informationen zur Zone** Es soll eine Möglichkeit geben zu der betrachteten Zone zusätzliche Informationen, wie Name der Zone und Objektbeschreibung, zu erhalten.
9	**Anzeige in der AR Ansicht** Sobald sich der Nutzer in der AR Ansicht befindet, soll es ihm möglich sein, das Objekt bei korrekter Positionierung des Smartphones, korrekt platziert im relativen Raum, im Bildschirm angezeigt zu bekommen.
10	**Navigation** Innerhalb des Features soll es ohne Probleme möglich sein zwischen den einzelnen Ansichten zu wechseln und wieder in den AREA Hauptbildschirm zu gelangen.

Tabelle 4.1: Funktionale Anforderungen

4.2 Nicht-funktionale Anforderungen

Nr.	Titel und Beschreibung der Nicht Funktionalen Anforderung
1	**Android Version** Die Applikation soll auf dem mobilen Betriebssystem Android 5.0 (API Level 21) und neuer funktionieren.
2	**Allgemeine Bedienbarkeit** Die Bedienung der neuen Funktion soll intuitiv und einfach möglich sein.
3	**Modularität** Die Möglichkeit, die Funktion für eigene Applikationen zu nutzen, und zu modifizieren, soll gegeben sein.

Tabelle 4.2: Nicht-funktionale Anforderungen

5 Architektur

Im folgenden Kapitel wird die Struktur von AREA mit dem Zusatz der AR-Zones näher beschrieben. Besonderen Wert wird dabei auf die in dieser Arbeit entwickelten Funktion gelegt. Die genauen Architekturbeschreibungen von AREA als Basis sind in [1] genauer gezeigt. Erst wird die generelle Architektur beschrieben und anschließend ein Überblick über die Klassenstruktur gegeben.

In Abbildung 5.1 ist die Architektur von AREA mit der in dieser Arbeit beschriebenen Funktion grafisch aufbereitet. Zu der grundlegenden Architektur AREA's sind einige Komponenten hinzugefügt worden, diese sind in der Abbildung farbig markiert. Beispielsweise spielt in der in dieser Arbeit beschriebenen Funktion die *Google Maps API* eine große Rolle. Die Darstellung der Zonen und auch die Navigation zu den Zonen für den Nutzer werden auf einer Oberfläche beschrieben, die durch Google zur Verfügung gestellt wird. Des weiteren wäre die fundamentalste neue Komponente zu nennen und zwar das Model *Zone*. Hierin wird beschrieben, welche Argumente die Zone und das zugehörige darzustellende Objekt haben soll. Die Darstellung derselben wird vom *Zone Controller* behandelt, welcher dafür zuständig ist, dass die Positionierung des Objekts im Raum korrekt durchgeführt wird. Die dazu benötigten Berechnungen und Umformungen der Koordinaten passieren in der *Locations* Komponente, genauer in *Zone Object Location Konvertierungen*. Die Objekte, von denen jede Zone eines besitzt, um es anzeigen zu können, haben ein eigenes View Objekt und zwar *Zonen Objekt*.

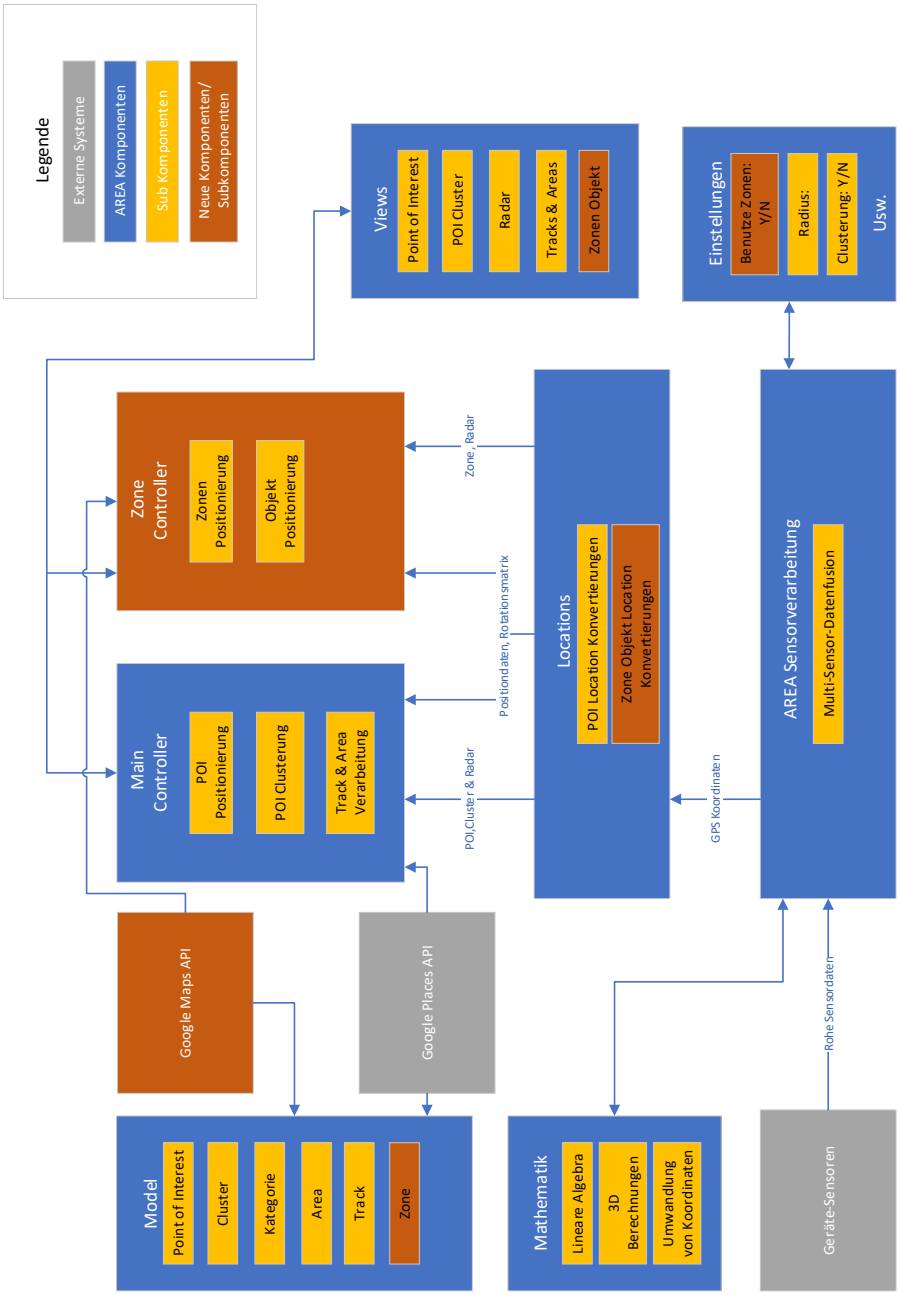

Abbildung 5.1: Architektur von AREA

Um die Klassen, welche für die Funktion der Zonen besonders wichtig sind vorzustellen, werden diese in Abbildung 5.2 dargestellt. Während die *MainActvity* für die grundlegenden AREA Funktionen zuständig ist, sind für die Zonen Funktion *ZoneActivity* und *ShowActivity* von großer Bedeutung. Ähnlich wie bei der *MainActivity*, die durch das *AREAMainFragment* unterstützt wird, ist das *AREAZoneFragment* für die neue Activity zuständig. Das Fragment steuert in beiden Fällen die komplette zugehörige Funktion AREA's, also das Darstellen aller Views in korrekten Positionen. Die dafür notwendigen Daten werden von den AREA eigenen Controllern geliefert.

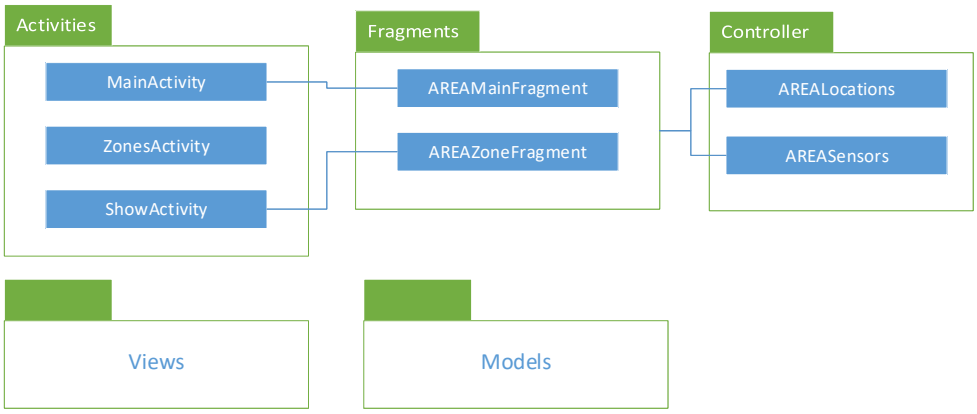

Abbildung 5.2: Klassen von AREA

Anschließend werden in Abbildung 5.3 und 5.4 noch einige für die Zonenfunktion wichtige Klassen vorgestellt.

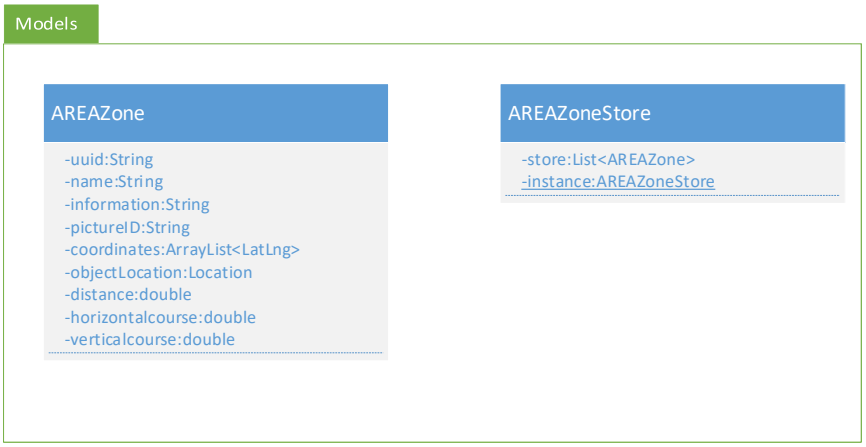

Abbildung 5.3: Neue Models in AREA

Wichtig zu erwähnen ist hier der *AREAZoneStore*. Ähnlich zu den POI's werden auch alle Zonen in einem applikationsweiten Store Objekt gespeichert. Dieses kann zum aktuellen Zeitpunkt bei Inbetriebnahme der Applikation erstellt werden. Eine Verknüpfung zu einer Datenbank wäre in Zukunft durchaus denkbar.

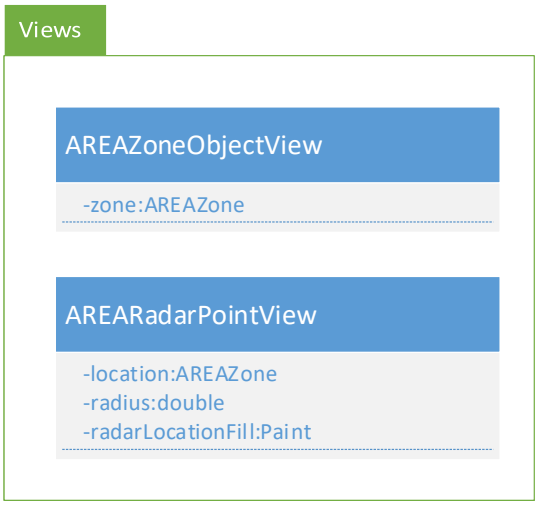

Abbildung 5.4: Neue Views in AREA

Auch die *AREARadarPointView* sei kurz erwähnt, da sie nur einen einzelnen Radarpunkt darstellt, der anzeigt, wo, relativ zum Nutzer, sich das in der AR zu sehende Objekt befindet.

6 Implementierungsaspekte

In diesem Kapitel werden nun einige Aspekte vorgestellt, welche wichtig für das zu entwickelnde Feature sind. Hierbei wird sowohl auf die Karte und die Zonen und deren Manipulation eingegangen, als auch auf die Positionsbestimmung und deren Korrektur bei Erkennung inkorrekter Daten.

6.1 Karte und Zone

Den Grundbaustein der Karte, die für das in dieser Arbeit beschriebene Feature benötigt wird, bildet die API von *Google Maps* für Android. Bei Aufrufen der Karte wird, sobald die Karte geladen wurde, direkt auf die aktuelle Person des Nutzers gezoomt. Dies wird in einem solchen Maßstab getan, dass der Nutzer seine unmittelbare Umgebung sehen und nach Zonen absuchen kann. Die Zonen, welche als „aktiv “ markiert sind, sollen dem Nutzer auf der Karte sichtbar gemacht werden. Zu den aktiven Zonen zählen alle, die sich im geladenen Zonestore Objekt befinden und innerhalb eines zwei Kilometer Radius um die Nutzerposition herum liegen.

Wenn die als aktiv markierten Zonen geladen wurden, wird in einer Schleife durch die Liste der Zonen iteriert und diese auf der Karte als Polygone gezeichnet. Die dafür nötigen Koordinaten sind in jeder Zone als Liste von LatLng Objekten gespeichert. Das Zeichnen der Polygone auf der Karte ist auf Abbildung 6.1, in den Zeilen drei bis fünf beschrieben.

```
1 for (AREAZone zone : activeZones) {// draw the active polygons
2
3   polygontemp = mMap.addPolygon(new PolygonOptions().clickable(true).
      fillColor(GRAY).strokeWidth(5)
4   .add(zone.getCoordinates().get(0), zone.getCoordinates().get(1), zone.
      getCoordinates().get(2), zone.getCoordinates().get(3)));
```

```
5     polygontemp.setTag(zone);
6     //if the current position of the User is inside a zone activate the
        Button to get to the AR view
7     if (isPointInPolygon(new LatLng(latitude, longitude), zone.
        getCoordinates())) {
8       ARButton.setVisibility(View.VISIBLE);
9       zoneCurrentlyIn = zone;
10    } else {
11      ARButton.setVisibility(View.GONE);
12    }
13
14  }
```

Listing 6.1: Zeichnen und Positionbestimmung

Wenn die Zonen nun für den Nutzer auf der Karte sichtbar sind, muss noch bestimmt werden, in welcher Beziehung der Nutzer zu den Zonen steht, also innerhalb oder außerhalb eines Zonenpolygons. Dies wird unter Zuhilfenahme eines Algorithmus bestimmt, welcher als „Ray Casting Algorithmus “ bezeichnet wird. Der Name kommt daher, dass von dem zu untersuchenden Punkt aus ein Strahl, also eine Linie gezogen wird, welche gegen Unendlich in x-Richtung geht. Nun kann aus der Anzahl der Schnittstellen dieses Strahls mit den Seiten des Polygons abgeleitet werden, ob sich der Punkt außerhalb oder innerhalb des Polygons befindet. Ist die Anzahl der Schnittstellen gerade, also durch zwei teilbar, ist der zu untersuchende Punkt außerhalb, ansonsten innerhalb. Der Erste Teil, also die Schleife, welche durch jede Seite des Polygons iteriert, ist in Abbildung 6.2 zu sehen.

```
1   private boolean isPointInPolygon(LatLng tap, ArrayList<LatLng> vertices)
      {
2     int intersectCount = 0;
3     for (int j = 0; j < vertices.size() - 1; j++) {
4       if (rayCastIntersect(tap, vertices.get(j), vertices.get(j + 1))) {
5         intersectCount++;
6       }
7     }
8
9     return ((intersectCount % 2) == 1); // odd = inside, even = outside;
10  }
```

Listing 6.2: Positionsbestimmung

Die einzelnen Schritte, um die Schnittstellen des Strahls mit den verschiedenen Seiten zu bestimmen, ist in Abbildung 6.3 geschildert.

```
1  private boolean rayCastIntersect(LatLng tap, LatLng vertA, LatLng vertB)
       {
2
3    double aY = vertA.latitude;
4    double bY = vertB.latitude;
5    double aX = vertA.longitude;
6    double bX = vertB.longitude;
7    double pY = tap.latitude;
8    double pX = tap.longitude;
9
10   if ((aY > pY && bY > pY) || (aY < pY && bY < pY)
11   || (aX < pX && bX < pX)) {
12     return false; // a and b can't both be above or below pt.y, and a or
13             // b must be east of pt.x
14   }
15
16   double m = (aY - bY) / (aX - bX); // Rise over run
17   double bee = (-aX) * m + aY; // y = mx + b
18   double x = (pY - bee) / m;
19
20   return x > pX;
21 }
```

Listing 6.3: Schnittpunkterkennung

Nun kann, nachdem alle Zonen gezeichnet wurden und bestimmt werden kann ob sich der Nutzer innerhalb einer Zone befindet, dem Nutzer die Möglichkeit der AR Ansicht gegeben werden, falls er sich in einer der Zonen befindet. Dies geschieht in Abbildung 6.1 in den Zeilen sieben bis elf. Mit dem jetzt für den Nutzer zugänglichen Button gelangt er in die Show Activity, welche in Verbindung mit dem zugehörigen Fragment als AR Ansicht dient und ähnlich zum AREA Hauptbildschirm aufgebaut ist. Dabei wird die Zone, in der sich der Nutzer aktuell befindet, als Argument „zoneCurrentlyIn “ mit übergeben, um sicherzustellen, dass auch die richtigen Informationen für die AR Ansicht zur Verfügung stehen. Dieses Argument wird in Zeile neun in Abbildung 6.1 gesetzt, sobald sich der Nutzer in einer Zone befindet.

Diese Methode, welche die Zonen auf der Karte sichtbar macht, die Nutzerposition relativ zu den Zonen evaluiert und den Activitywechsel zur AR Ansicht möglich macht, wird jedes Mal ausgeführt, wenn sich die Nutzerposition ändert. Dabei werden vor jeder Ausführung alle Zonen von der Karte entfernt und das Argument *zoneCurrentlyIn* „null “ gesetzt, um dynamisch Änderungen anzeigen zu können.

6.2 Positionsbestimmung

Ein großes Problem bei der Positionsbestimmung von AREA war bisher die korrekte Bestimmung der Nutzerhöhe in der Nähe von Gebäuden oder in ihnen. Da die für die Daten der POI's genutzte *Google Places API* keine Höhendaten zu den Koordinaten liefert, werden diese bisher mit der Nutzerhöhe initialisiert. Da diese Höhe aber stark variieren kann, aufgrund von inkorrekten GPS Daten, war es bisher der Fall, dass die POI's manchmal stark nach oben oder unten im Raum „gesprungen“ sind. Um dieses Problem zu umgehen, wurde ein Workaround gefunden. Dieser sieht vor, dass die ersten Positionsdaten, welche auch für die Höhendaten der POI's benutzt werden, über WLAN oder mobile Netzwerkprovider initialisiert werden. Dies stellt sicher, dass auch in Gebäuden, oder in deren Nähe, die Initialhöhe korrekt bestimmt werden kann. Dies wird in Abbildung 6.4 veranschaulicht.

```
1 locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 500,
      distanceFilter, this);
2 locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
    500, distanceFilter, this);
3 prevLocation=locationManager.getLastKnownLocation(LocationManager.
    NETWORK_PROVIDER);
4 // first "last know location" is initialized with network because the
    altitude of Gps may very significantly in Buildings or near them
```

Listing 6.4: Positionssensor Datenverarbeitung

Um nun weiterhin sicherzustellen, dass die Location updates nur, mit an Sicherheit grenzender Wahrscheinlichkeit, korrekte Werte annehmen, werden die von GPS und WLAN/Netzwerkprovidern gewonnenen Positionsdaten gefiltert.

Dies äußert sich darin, dass von der initial gesetzten Position *prevLocation* nur solche Updates akzeptiert und verwendet werden, welche sich von der Höhe um weniger als zwei Meter unterscheiden. Diese Grenze wurde gewählt, da die Aktualisierung der Positionsdaten mehrmals pro Sekunde abläuft und somit eine Änderung von mehr als zwei Metern in der Höhe als unwahrscheinlich angenommen wird. Diese beschriebene Datenfilterung ist in Abbildung 6.5 gezeigt.

```
1  public void onLocationChanged(Location location) {
2    if (listener != null && location != null) {
3      if(Math.abs(prevLocation.getAltitude()-location.getAltitude())<2){
4        listener.didUpdateLocation(location);
5        prevLocation=location;
6      }
7      else{
8        listener.didUpdateLocation(prevLocation);
9      }
10   }
11 }
```

Listing 6.5: Positionsdatenfilter

7 Vorstellung der Anwendung

In diesem Kapitel der Arbeit wird die aktuelle Version von AREA mit der neuen Zonenfunktion beschrieben und mit Screenshots grafisch vorgestellt.

7.1 AREA Grundfunktionen

Begonnen wird mit den Funktionen, die bereits in AREA implementiert wurden. Als erstes wird der Fall beschrieben, indem in den AREA Einstellungen die Funktion *Google Loader* gesetzt ist. Nun werden, wenn die Applikation gestartet wird, nachdem der AREA Hauptbildschirm angezeigt wird, eine Verbindung zur *Google Places API* aufgebaut, und damit begonnen, POI's und die dazugehörigen Daten herunterzuladen. Dies wird dem Nutzer mit einer Loading View in der Mitte des AREA Hauptbildschirms signalisiert. Dies ist in Abbildung 7.1 gezeigt.

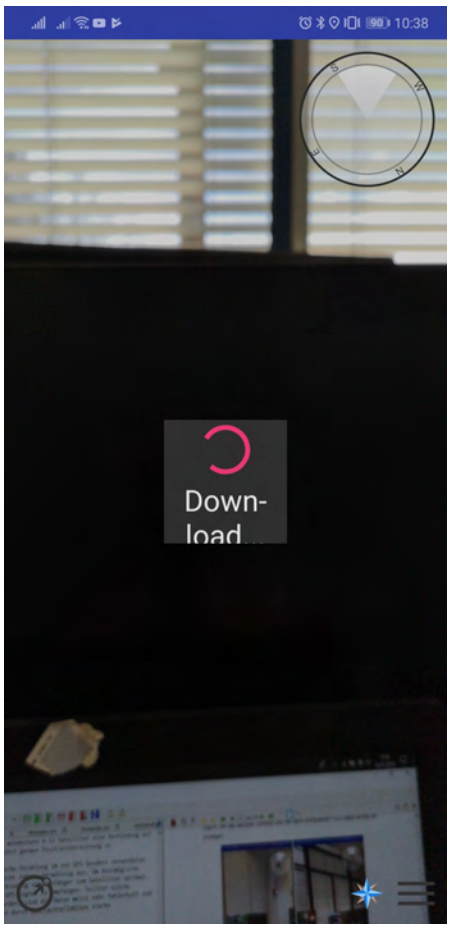

Abbildung 7.1: Laden der POI's von Google

Wenn die POI's nicht von Google geladen werden sollen, müssen sie nativ deklariert oder aus einer anderen Quelle bezogen werden, ansonsten passiert beim Starten der Applikation nichts, bis auf das Anzeigen des AREA Hauptbildschirms.

Wenn die POI's geladen wurden, werden sie auf dem AREA Hauptbildschirm angezeigt, wie in Abbildung 7.2 zu sehen ist. Falls in den Einstellungen das Clustern der POI's gesetzt ist, werden POI's, die im Bildschirm nahe zusammen gelegenen sind, in Clustern angezeigt. Dies ist in Abbildung 7.2 der Fall. Möchte der Nutzer nun wissen, welche POI's sich in diesen Clustern befinden, bringt ein Tipp auf das gewünschte Cluster ihn in die Cluster Ansicht, welche in Abbildung 7.3 dargestellt wird.

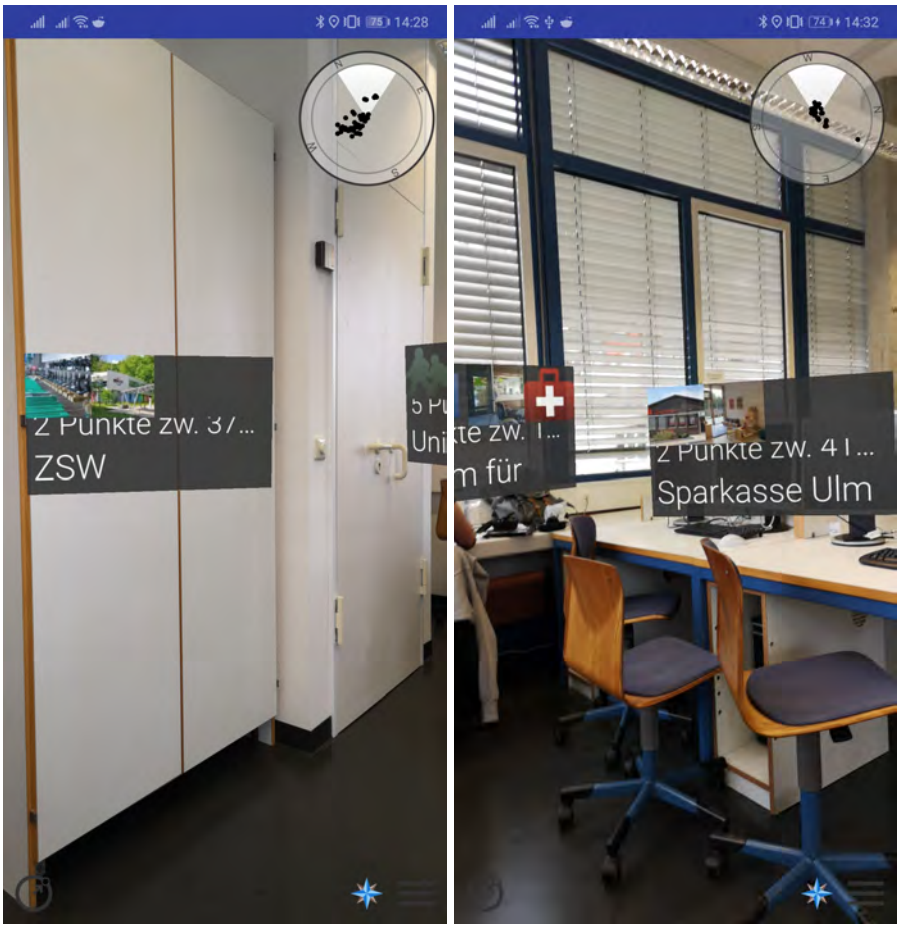

Abbildung 7.2: POI's in AREA

Falls die Cluster Funktion nicht aktiv ist, überlappen sich nahegelegene POI's, können aber durch skalierende Größe der Views in der Entfernung zum Nutzer unterschieden werden. Dies äußert sich darin, dass näher gelegene POI's größer angezeigt werden als weiter entfernte. Zu beachten ist weiterhin das Radar in der rechten oberen Ecke des Bildschirms, zu sehen in Abbildung 7.2. Hier werden sämtliche POI's, welche sich im Radius, der in den Einstellungen gesetzt wurde, befinden, als schwarze Punkte angezeigt. Ihre Position im Radar bezieht sich auf die relative Positionierung zum Nutzer. Der etwas hellere Teil des Radars symbolisiert das aktuelle Sichtfeld des Nutzers im Hauptbildschirm.

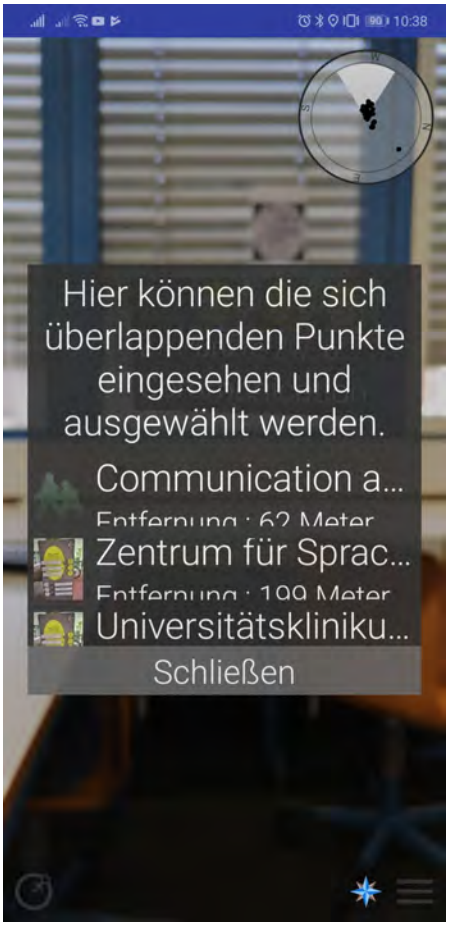

Abbildung 7.3: Clusteransicht

Das Radar dreht sich je nach Änderung der Ausrichtung des Gerätes, also auf allen drei Bewegungsachsen im Raum, um das aktuelle Sichtfeld und die darin enthaltenen POI's stets korrekt anzuzeigen. Von diesem Punkt aus gibt es mehrere Navigationsmöglichkeiten in der Applikation. Dazu zählen die Auswahl des Radius durch Tippen auf das Radar oben rechts oder den Radius-Picker Button unten links, zu sehen in Abbildung 7.2 und das Auswählen von POI Kategorien durch Tippen auf den Listen Button unten rechts in Abbildung 7.2. Die letzte Möglichkeit ist über den Button in Form einer Windrose, ebenso unten rechts im Bildschirm auf die Kartenansicht zu der Zonenfunktion zu gelangen.

Sollte der Nutzer nun eine der zuvor genannten Möglichkeiten zur Änderung des Radius gewählt haben, wird der Radius Picker wie in Abbildung 7.4 zu sehen, angezeigt. Hier wird der aktuelle Radius angezeigt und wie viele POI's innerhalb des selbigen liegen. Nun hat er die Möglichkeit den aktuellen Radius mithilfe des Schiebereglers zu verändern. Falls der Radius verändert wurde und der Picker über den *Schließen* Button geschlossen wurde, befindet sich der Nutzer wieder auf dem AREA Hauptbildschirm auf dem jetzt, falls vorhanden, neue POI's angezeigt werden oder im Umkehrschluss die POI's ausgeblendet werden, welche sich außerhalb des eingestellten Radius befinden. Ebenfalls passt sich die Skalierung des Radars bezüglich des gewählten Radius an.

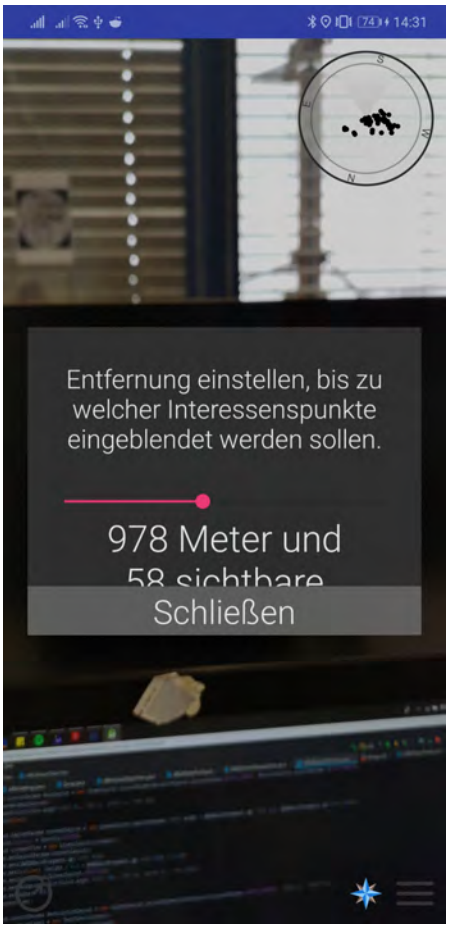

Abbildung 7.4: Radius picker

Wenn der Nutzer, wie zuvor beschrieben, durch das An- und Abwählen von Kategorien, die angezeigten POI's filtern möchte und auf den Kategorie Button getippt hat, wird der Category Picker angezeigt. Dieser ist in der Abbildung 7.5 zu sehen. Dort werden die Kategorien aller sich aktuell im Radius befindenden POI's angezeigt. Nun kann der Nutzer durch einen Tipp auf die Checkbox neben der gewünschten Kategorie diese an- oder abwählen. Nach Schließung der Ansicht über den *Schließen* Button werden die POI's aktualisiert angezeigt und nur jene, welche in die gewählten Kategorien einzuordnen sind, bleiben sichtbar.

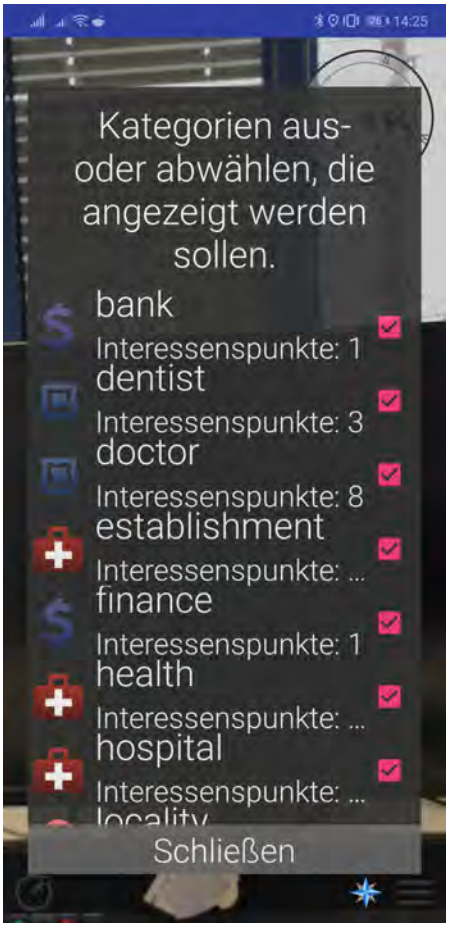

Abbildung 7.5: Category picker

7.2 AREA Zone Funktion

Wenn der Nutzer, wie in Kapitel 7.1 beschrieben, auf den Button in Form einer Windrose auf dem AREA Hauptbildschirm tippt, gelangt er auf die Karte. Dies wird von Abbildung 7.6 veranschaulicht. Hier kann der Nutzer in einem Radius von zwei Kilometern alle zur Verfügung stehenden Zonen in seiner Umgebung sehen. Eine Beispielzone befindet sich in der Mitte der Abbildung 7.6. Beim Aufrufen der Karte wird automatisch auf den aktuellen Standort des Nutzers in einem angemessenen Maßstab fokussiert. Sollte der Nutzer durch Verschiebung der Karte seinen Standort aus den Augen verlieren, kann durch einen Tipp, auf den rechts oben in Abbildung 7.6 dargestellten Standort Button, die Karte wieder auf den eigenen Standort zentriert werden. Außerdem kann über die sich rechts unten im Bild befindenden + und - Buttons, oder mit dem Zusammenziehen bzw. Auseinanderziehen zweier Finger, nach Belieben gezoomt werden. Möchte der Nutzer wieder zurück auf den AREA Hauptbildschirm, kann das über das Benutzen der *Zurück*-Funktion seines Smartphones einfach erreicht werden.

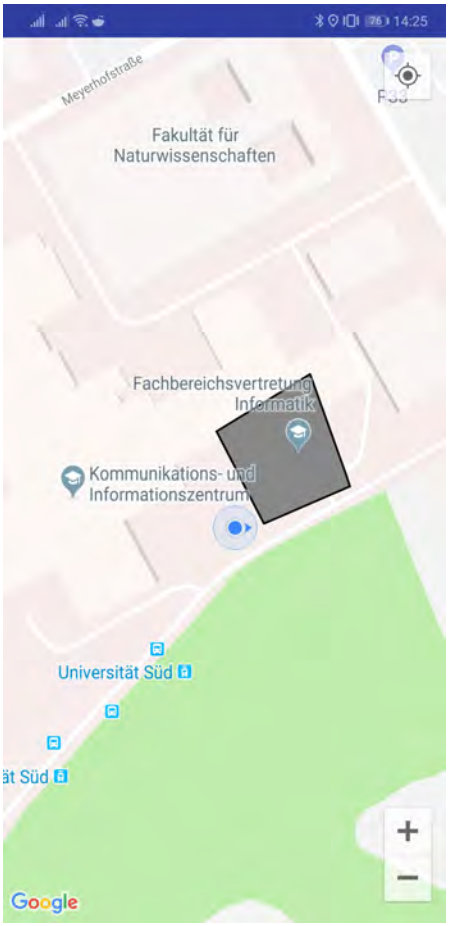

Abbildung 7.6: Karte in AREA

Möchte der Nutzer nun herausfinden, was in den angezeigten Zonen zu sehen ist, genügt ein Tipp auf das Polygon der Zone auf der Karte. Anschließend werden ihm genauere Informationen zu der gewünschten Zone präsentiert. Dieses Feature ist in Abbildung 7.7 zu sehen.

Abbildung 7.7: Zone mit Zusatzinformationen

Begibt sich der Nutzer nun in eine der Zonen, oder befindet sich bereits darin, wird wie in Abbildung 7.8 zu sehen ein Button eingeblendet, über den er in die AR Ansicht der Zone wechseln kann. Bewegt sich der Nutzer wieder aus der Zone heraus, wird der Button ausgeblendet.

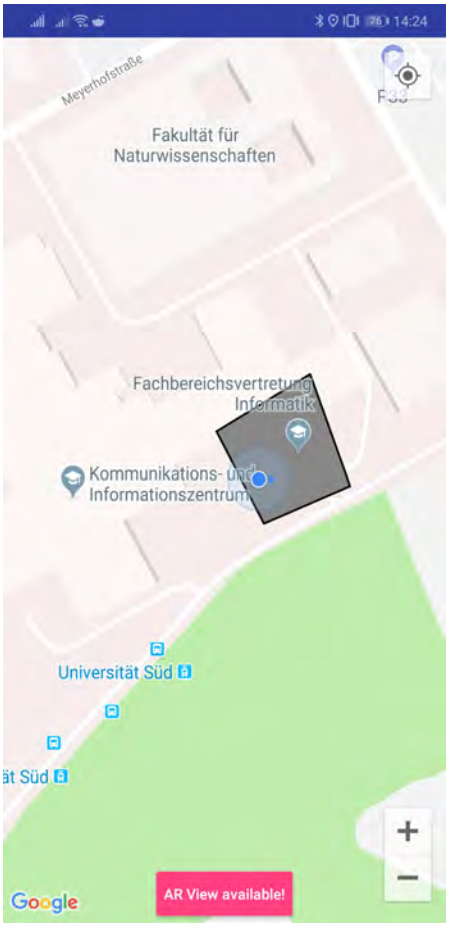

Abbildung 7.8: Nutzer in einer Zone

Wechselt der Nutzer in die AR Ansicht der Zone, welche in Abbildung 7.9 gezeigt wird, hat er die Möglichkeit das zugehörige Objekt auf dem Bildschirm in der korrekten Ausrichtung des Gerätes zu sehen. Wo sich das Objekt relativ zu der Blickrichtung des Nutzers befindet, kann anhand des Radars oben rechts in der Abbildung abgelesen werden. Ähnlich wie im Hauptbildschirm von AREA wird die Position des Objekts als einzelner Punkt, genauso wie das aktuelle Sichtfeld des Nutzers angezeigt. Auch in dieser Ansicht aktualisiert sich die Radar Ansicht je nach Geräteausrichtung, was es dem Nutzer leichter machen soll eine gute Sicht auf das Objekt zu erhalten. Zu erwähnen ist hier, dass es sich bei dem Objekt nicht um ein 3D Modell handelt, sondern lediglich um ein Bild. Das Bild sollte also die Perspektive des Objektes zeigen, welche der Nutzer in der Zone sehen soll.

Somit muss beim Erstellen der Zonen darauf geachtet werden, dass die Zone klein genug ist, sodass die Perspektive des Bildes nicht fehlerhaft dargestellt wird.

Abbildung 7.9: Objekt View

Des weiteren ist wichtig, dass, sobald der Nutzer wieder aus der Ansicht zurück wechseln möchte, er entweder die *Zurück*-Funktion seines Smartphones benutzen kann, oder über den sich wieder rechts unten im Bild befindlichen Button in Form einer Windrose, zurück auf die vorherige Activity gelangt. In diesem Fall wäre das die Karte aus Abbildung 7.6.

8 Anforderungsabgleich

In diesem Kapitel werden die Anforderungen aus Kapitel 4 mit dem implementierten System abgeglichen.

8.1 Funktionale Anforderungen

Nr	Titel und Abgleich der funktionalen Anforderung
1	**GPS in komplexen Umgebungen** *teilweise erfüllt* Die Schwierigkeit besteht darin, dass das GPS wie in Kapitel 2.2 beschrieben, sehr inkorrekte Daten, vor allem in der Höhe, liefern kann sobald man sich nahe oder in Gebäuden befindet. Allerdings wird hier zum Initialisieren der Objekte/POI's WLAN Positionierung benutzt, um konsistente Ergebnisse zu liefern. Korrekte GPS und Höhen Darstellung bleibt jedoch schwierig, da bisher zu POI's keine Höhe von Google mitgeliefert wird.
2	**Anzeigen einer Karte** *erfüllt* Die Nutzung von *Google Maps* funktioniert tadellos und ermöglicht einfache Manipulation der Karte.
3	**Nutzerposition auf der Karte** *erfüllt*
4	**Anzeigen der Zonen auf der Karte** *erfüllt* Dadurch dass Zonen keine maximale Anzahl an Eckpunkten haben, sind sogar komplexere Zonen kein Problem.
5	**Wechseln in die AR Ansicht** *erfüllt* Manchmal benötigt der Algorithmus allerdings ein kurzes Zeitfenster von weniger als einer Sekunde, um korrekt zu Berechnen, ob sich der Nutzer in einem Polygon befindet.

6	**Ausblendung entfernter Zonen** *erfüllt*
7	**Zusätzliche Informationen zur Zone** *erfüllt* Dies ist auf nur der Karte möglich, in der AR Ansicht eine Informations-View einzubauen, wäre aber durchaus denkbar.
8	**Anzeige in der AR Ansicht** *erfüllt* Je nach GPS Empfang ist die Anzeige mehr oder weniger konstant. Korrekte Occlusion, also die Verdeckung von Objekten, welche eigentlich vor dem Objekt liegen, ist allerdings nicht möglich.
9	**Navigation** *erfüllt* Da die Instanzen der Activities als schichtweise aufeinander gestapelt werden ist die Rückführung auf den Start- und Hauptbildschirm von AREA über die Zurück Funktion eines jeden Smartphones sehr intuitiv.

Tabelle 8.1: Abgleich der funktionalen Anforderungen

8.2 Nicht-funktionale Anforderungen

Nr	Titel und Abgleich der Nicht Funktionalen Anforderung
1	**Android Version** *erfüllt*
2	**Allgemeine Bedienbarkeit** *erfüllt* weitestgehend, es wäre noch möglich gewesen, bei Erstbenutzung Benachrichtigungen und Hilfen einzubauen, um dem Nutzer klar zu machen, dass er in die Zonen laufen muss, um in die AR Ansicht zu gelangen oder Ähnliches.
3	**Modularität** *erfüllt* Durch den ZoneStore ist es sehr einfach möglich entweder eigene Zonen zu definieren, oder eine Datenbank einzubinden, aus der die Zonen bezogen werden, ähnlich zu *Google Places*.

Tabelle 8.2: Abgleich der Nicht-funktionalen Anforderungen

9 Zusammenfassung und Ausblick

In diesem Kapitel wird die Arbeit zusammengefasst und ein kurzer Ausblick gegeben.

9.1 Zusammenfassung

Abschließend sei gesagt, dass das Ziel dieser Arbeit, also die Konzeptionierung und Implementierung eines location-based Features zur Unterstützung des AREA Frameworks durchaus erreicht wurde. Zu Beginn der Arbeit wurden wichtige Hintergrundinformationen bereitgestellt, und anschließend verwandte Arbeiten vorgestellt. Darauf folgend wurden die Anforderungen an das Feature definiert und die Architektur AREA's unter Erweiterung des neuen Features beschrieben. Auch im Implementierungsteil werden ausgewählte Aspekte behandelt und im Anschluss das Endprodukt mit den im Vorhinein definierten Anforderungen abgeglichen.

9.2 Ausblick

Eine Möglichkeit diese Arbeit und generell das AREA Projekt weiterzuführen wäre, die Google Places API entweder zu ersetzen oder zusätzlich die korrekten Höhendaten der POI's aus einer anderen Quelle zu beziehen. Dies würde der einer korrekten Darstellung der POI's zugute kommen. Ebenfalls wäre für die Zonen eine Integration in eine Datenbank möglich, um viele verschiedene Zonen speichern zu können.

Eine der offensichtlichsten Verbesserungsmöglichkeiten dieses Features findet sich in der AR Ansicht.

Bisher ist es nur möglich, Bilder des gewünschten Objektes anzuzeigen.
Hier wirklich ein dreidimensionales Objekt anzeigen zu können, würde das Feature ungleich mächtiger machen als es aktuell ist. Dies würde viele neue Möglichkeiten eröffnen und auch die Einschränkung der Zonen was Größe betrifft aufheben, da dann sich die AR Ansicht je nach Standpunkt in der Zone dynamisch anpassen könnte.

Literatur

1. Pryss, R., Geiger, P., Schickler, M., Schobel, J. & Reichert, M. The AREA Framework for Location-Based Smart Mobile Augmented Reality Applications. *International Journal of Ubiquitous Systems and Pervasive Networks (JUSPN)* **9,** 13–21 (Juli 2017).
2. Arth, C. *et al.* The History of Mobile Augmented Reality (Mai 2015).
3. Lipinski, K. GPS (global positioning system). Acessed: 05.03.2019. `https://www.itwissen.info/GPS-global-positioning-system-GPS-System.html` (Feb. 2015).
4. Wendel, J. *Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation* ISBN: 9783486595154. `https://books.google.de/books?id=aaWlCQAAQBAJ` (De Gruyter, 2009).
5. Pryss, R. *et al.* Enabling Tracks in Location-Based Smart Mobile Augmented Reality Applications. *Procedia Computer Science* **110,** 207–214 (2017).
6. Geiger, P. *Entwicklung einer Augmented Reality Engine am Beispiel des iOS* Sep. 2012.
7. Geiger, P., Pryss, R., Schickler, M. & Reichert, M. *Engineering an Advanced Location-Based Augmented Reality Engine for Smart Mobile Devices* Technical Report UIB-2013-09 (Ulm University, Ulm, Okt. 2013). `http://dbis.eprints.uni-ulm.de/972/`.
8. Geiger, P., Schickler, M., Pryss, R., Schobel, J. & Reichert, M. *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* (Apr. 2014), 383–394. `http://dbis.eprints.uni-ulm.de/1028/`.

9. Pryss, R., Geiger, P., Schickler, M., Schobel, J. & Reichert, M. Advanced Algorithms for Location-Based Smart Mobile Augmented Reality Applications. *Procedia Computer Science* **94,** 97–104. http://dbis.eprints.uni-ulm.de/1405/ (Aug. 2016).
10. Lindström, D. *Visualizing future buildings : User-centered design process and evaluation of a sensor-based MAR prototype* Magisterarb. (KTH, School of Computer Science und Communication (CSC), 2017).
11. https://www.pokemongo.com/de-de/. Acessed: 01.03.2019.
12. https://www.ingress.com/de/game/. Acessed: 01.03.2019.
13. https://www.harrypotterwizardsunite.com/de/. Acessed: 01.03.2019.
14. Paladini, M. *3 different types of AR explained: marker-based, markerless & location* https://www.blippar.com/blog/2018/08/14/marker-based-markerless-or-location-based-ar-different-types-of-ar. Accessed: 26.02.2019. Aug. 2014.

Abbildungsverzeichnis

5.1 Architektur von AREA 12
5.2 Klassen von AREA 13
5.3 Neue Models in AREA 14
5.4 Neue Views in AREA 14

7.1 Laden der POI's von Google 22
7.2 POI's in AREA 23
7.3 Clusteransicht 24
7.4 Radius picker 25
7.5 Category picker 26
7.6 Karte in AREA 28
7.7 Zone mit Zusatzinformationen 29
7.8 Nutzer in einer Zone 30
7.9 Objekt View 31

Tabellenverzeichnis

4.1 Funktionale Anforderungen 10
4.2 Nicht-funktionale Anforderungen 10

8.1 Abgleich der funktionalen Anforderungen 34
8.2 Abgleich der Nicht-funktionalen Anforderungen 34

Name: Lucas Tilmann Gmünder Matrikelnummer: 868101

Erklärung

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

Ulm, den .....

Lucas Tilmann Gmünder