ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Optimierung AR-Klassik unter Verwendung von AR-Foundation und generische Erweiterung der AR-Usability in iOS- und Android-Apps

Masterarbeit an der Universität Ulm

Vorgelegt von:
Leoni Holl
leoni.holl@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Rüdiger Pryss

Betreuer:
Jens Winkler

2020

Fassung 25. April 2020

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Augmented Reality (AR) ermöglicht die Gestaltung von neuen Erfahrungen im Bereich Tourismus. Es können neue Dienste auf mobilen Endgeräten zur Verfügung gestellt werden, sodass Städte mit ihren Bürgern und Touristen näher verbunden werden. Dabei können Benutzer Orte aus einem ganz neuen Blickwinkel durch die Gerätekamera erkunden. Innerhalb der AR-Anwendung werden Standorte markiert und zusätzliche Informationen für den Benutzer bereitgestellt.
Die Stadt-App der Firma cm city media setzt diese Idee um. Die Anwendung besteht aus verschiedenen Modulen. Eines der Module ist der Liveguide, welcher einem Augmented Reality-Modul entspricht. Dabei werden interessante Orte durch Points of Interest, die in der Kameraansicht erkundet werden können, markiert. Ziel dieser Masterarbeit ist es, dieses AR-Modul nachzubilden. Das Modul wird mit Hilfe des Frameworks AR-Foundation von Unity entwickelt. Der Fokus liegt auf der Darstellung, Positionierung und Berechnung von Points of Interest und Clustern, welche Punkte zusammenfassen, die sich in der Kameraansicht überlappen würden. Außerdem werden die Points of Interest um Verknüpfungen erweitert, die es dem Benutzer einfacher und übersichtlicher machen Informationen dazu anzufordern.

Inhaltsverzeichnis

1

Einleitung

Die Entwicklung von Smartphones schreitet sehr schnell voran. Jede Verbesserung und jedes neu entwickelte Merkmal bietet neue Möglichkeiten für die Verwendung von mobilen Anwendungen. Der Ausbau und die Bereitstellung der Gerätesensoren ermöglicht es beispielsweise Augmented Reality-Anwendungen zu realisieren. Dabei wird die Umgebung des Benutzers erfasst und mit virtuellen Informationen oder Objekten erweitert. Die Informationen können in verschiedenen Formaten vorliegen, dazu zählen Bilder, Videos, Audio-Ausschnitte, Webseiten und vieles mehr [3].
Dass Augmented Reality, kurz AR, immer attraktiver und für den Benutzer immer greifbarer wird, zeigt die Statistik in Abbildung 1.1. Es wird aufgezeigt, dass die Anzahl an Benutzern von mobilen AR-Anwendungen über die Jahre deutlich steigt. So wurden im Jahr 2019 1,46 Milliarden Benutzer erfasst, während im Jahr 2023 mit 2,4 Milliarden Benutzern in diesem Bereich zu rechnen ist. Da AR-Applikationen einfach auf mobilen Geräten bedienbar sind und in den App-Stores zur Verfügung stehen, sind sie für eine Vielzahl an Benutzern zugänglich. Daraus ist ebenfalls abzuleiten, dass auch die Anzahl an angebotenen Anwendungen im Bereich AR in den letzten Jahren deutlich stieg und auch in Zukunft weiter steigen wird. In vielen Bereichen unterstützen AR-Anwendungen Benutzer und Anwender. Ein Beispiel dafür ist der Automobilbereich, in welchem den Kunden neue Erfahrungen beim Kauf ermöglicht werden können. So kann der Innenbereich eines Fahrzeugs durch die Kameraansicht betrachtet werden, in der weitere Informationen für bestimmte Teile der Innenausstattung zur Verfügung stehen [1].

Auch im Bereich Tourismus bietet der Einsatz von AR neue Möglichkeiten. Touristen können Orte mit deren Sehenswürdigkeiten und Plätzen aus einem anderen Blickwinkel

erleben, wenn AR-Anwendungen schnell und einfach erweiterte Informationen dafür zugänglich machen [1]. Auch der Zugriff auf verschiedene Dienste und Angebote kann

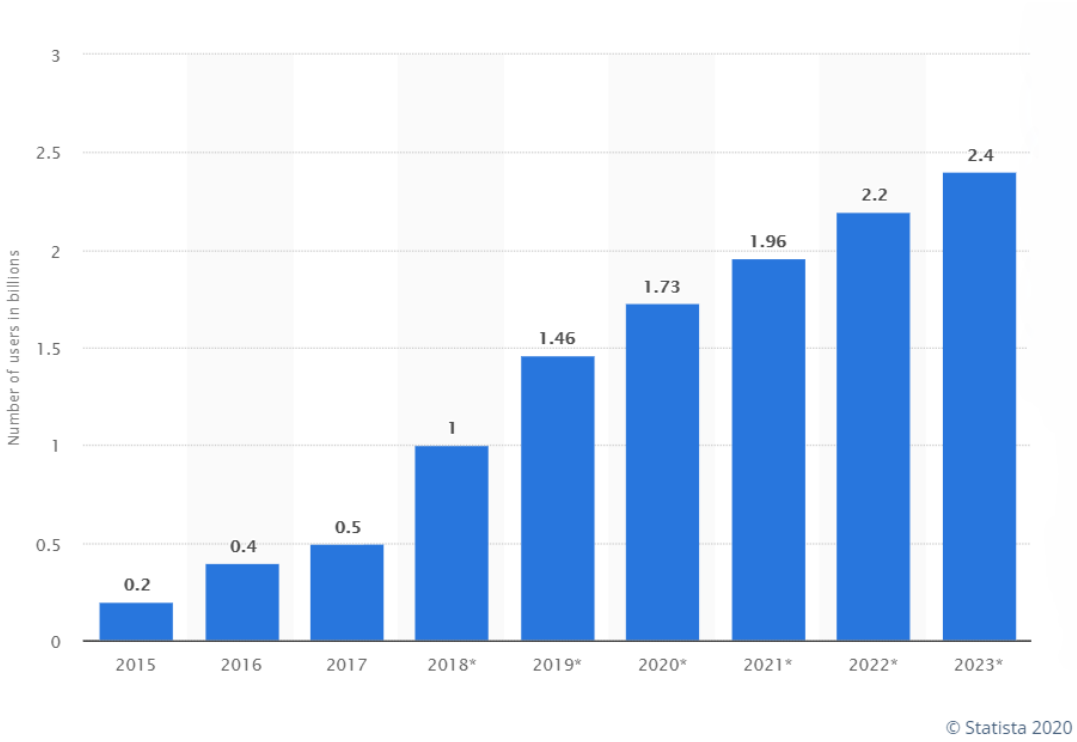


Abbildung 1.1: Anzahl der mobilen AR-Benutzer weltweit von 2015 bis 2023 (in Milliarden), * Prognose [2]

dadurch vereinfacht werden. Dazu zählen die Buchung von Tickets und Veranstaltungen, die Tischreservierung in einem Restaurant oder auch das Hinterlassen und Abrufen von Bewertungen an bestimmten Orten. Zusätzlich können AR-Anwendungen ganze Stadt-Führungen übernehmen, indem eine Navigation zwischen ausgewählten Punkten für Benutzer zur Verfügung steht. Des Weiteren besteht die Möglichkeit, dass Benutzer selbst bestimmen können, welcher Inhalt für sie interessant ist. Durch Filtereinstellungen kann der AR-Inhalt personalisiert werden und irrelevante Informationen ausgeblendet werden [3].

Für diesen Zweck steht auch die Stadt-App von cm city media zur Verfügung. Die Idee dahinter ist, Gemeinden, Städte und Landkreise mit ihren Bürgern und Touristen zu verbinden. So können News, Veranstaltungen und Dienste in der App für alle Interessierten

zur Verfügung gestellt werden [4]. Unter anderem kann die Umgebung mit Hilfe des Liveguides durch die Gerätekamera erkundet werden. Dabei gehören Points of Interest, kurz POIs, die relevante Plätze markieren, zum eingeblendeten AR-Inhalt. Nähere Informationen können durch das Anklicken eines solchen Punktes angefordert werden. Für die Einbindung von AR wurde das Framework AREA, Augmented Reality Engine Application, entwickelt. Der Liveguide soll mit dieser Arbeit neu aufgesetzt werden, um die Entwicklung von erweiterten Funktionen sicherzustellen und die Probleme von AREA zu umgehen. Für die Umsetzung kommt das Framework AR-Foundation von Unity zum Einsatz. Die Entwicklung des Unity-Moduls und die Verwendung von AR-Foundation wird in dieser Arbeit näher vorgestellt.

1.1 Problemstellung

Für die Entwicklung des Liveguides in der Stadt-App wurde das AREA-Framework implementiert, siehe dazu Kapitel 3.1. Dies bringt einige Probleme mit sich. AREA implementiert einen Kern für standortbezogene AR-Anwendungen ohne Zuhilfenahme von weiteren, aktuellen Frameworks oder Technologien. Durch die schnelle technische Weiterentwicklung der Smartphones und Tablets verbessern sich kontinuierlich die Möglichkeiten neue Funktionalitäten für Anwendungen zu entwickeln oder bestehende Funktionalitäten anzupassen. Verbesserungen sind dabei zum Beispiel das Hinzufügen neuer Sensoren. Da AREA auf die Sensordaten zurückgreift, muss das Framework ständig an Änderungen angepasst werden, über welche sich Entwickler regelmäßig informieren müssen [1]. Da hinter der Entwicklung des AREA-Frameworks nur wenige Entwickler stehen, ist es schwerer möglichst flexibel auf Änderungen und Anpassungen zu reagieren, als es in einem großen Unternehmen möglich wäre. Außerdem steht AREA für Android- und iOS-Geräte jeweils in einer nativen Entwicklung zur Verfügung, das heißt Anpassungen müssen in zwei Projekten vorgenommen werden. Hinzu kommen Problematiken bei der Entwicklung von Erweiterungen, die in dem Framework vorgenommen werden sollen.
Da alle Grundfunktionalitäten im Kern selbst implementiert werden müssen, entsteht ein hoher Aufwand und Komplikationen können leicht auftreten. Eine Grundfunktionalität ist

zum Beispiel das Rendering. Dies wird in üblichen AR-Frameworks komplett übernommen, muss in AREA allerdings selbst implementiert werden, wodurch der Fokus nicht auf der eigentlichen Funktionalität konzentriert sein kann.
Ein weiterer Punkt ist das Thema Stabilität. Anwendungen, die auf AREA aufgebaut sind, haben oft das Problem, dass POIs nicht stabil angezeigt werden. Durch eine ungenaue Verarbeitung der Positions-Daten, die das Gerät zur Verfügung stellt, wechseln AR-Objekte und der Benutzer selbst in kurzen Abständen die Position. Dies führt zu einem Springen der AR-Objekten in der Kamera-Ansicht.

1.2 Zielsetzung

Ziel der Arbeit ist es ein neues Modul zu implementieren, das den bisherigen AREA-Liveguide nachbaut und ersetzt. Die Probleme des AREA-Frameworks sollen damit größtenteils behoben werden. Das Modul soll die Funktionalitäten und auch das Aussehen des AREA-Liveguides übernehmen. Als Unterstützung soll das Framework AR-Foundation von Unity verwendet werden. Im Folgenden wird das neue Modul als Unity-Modul oder Unity-Liveguide betitelt. AR-Foundation übernimmt viele Implementierungen bezüglich der AR-Grundfunktionalitäten, wodurch neue Erweiterungen mit einem deutlich geringeren Aufwand umgesetzt und in das Unity-Modul aufgenommen werden können. Das Framework soll außerdem aktuelle Technologien anbieten und den Entwicklern durch regelmäßige Updates Anpassungen und neue Features zur Verfügung stellen. Diese Anforderungen erfüllt AR-Foundation. Außerdem soll durch das Framework erreicht werden, dass eine Entwicklung auf beiden Plattformen, Android sowie iOS, lauffähig ist. Darüber hinaus soll das neue Modul ein stabiles und präzises AR-Erlebnis anbieten können. Dafür soll ein Unity-Plugin, AR+GPS Location, zum Einsatz kommen. Der Unity-Liveguide soll letztendlich in die native Android- und iOS-Applikation eingebunden werden. Dort soll das Modul auf allen Geräten aktiv werden, welche die Anforderungen der neuen Technologien erfüllen. Auf den anderen Geräten soll weiterhin der AREA-Liveguide zur Verfügung stehen. Abbildung 1.2 zeigt dazu die Architektur der Stadt-App mit Fokus auf den Liveguide. Auf der linken Seite ist der bisherige Aufbau, bei welchem der AREA-Liveguide zum Einsatz kommt. Auf der rechten Seite ist der neue Aufbau,

bei welchem, wenn möglich, der Unity-Liveguide aufgerufen wird. Ziel ist es, dass in möglichst vielen Fällen der Unity-Liveguide zum Einsatz kommt.

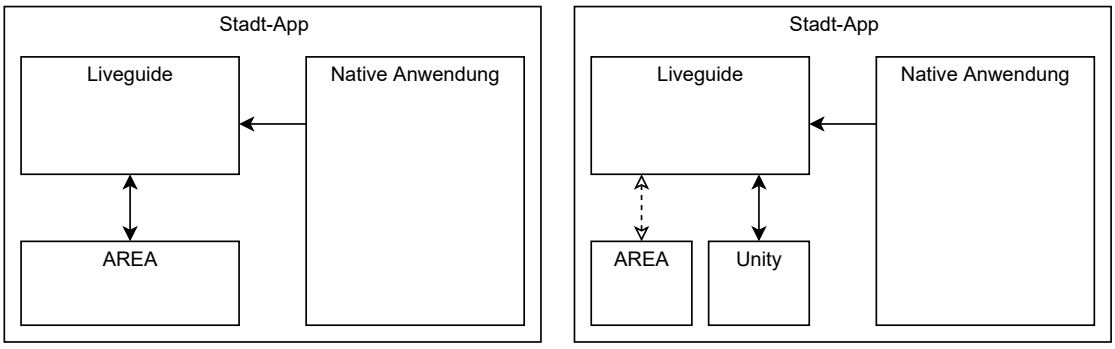


Abbildung 1.2: Architektur des Liveguides in der Stadt-App bisher (links) und neu (rechts)

Zusätzlich zu allen Funktionen, die von dem AREA-Liveguide in das Unity-Modul übernommen werden, soll eine AR-Erweiterung implementiert werden. Diese soll dem Benutzer mehr Interaktionsmöglichkeiten mit den AR-Objekten, POIs und Cluster, bieten.

1.3 Struktur der Arbeit

Im Folgenden wird der Aufbau sowie eine kurze inhaltliche Angabe der Kapitel gegeben. Das Kapitel 2 erläutert zunächst einige Begriffe, die dem Verständnis der Arbeit dienen. Dabei wird zunächst auf die Technologie Augmented Reality eingegangen. Anschließend wird das Projekt *Stadt sind wir* vorgestellt, das den integrierten Liveguide anbietet, der in dieser Arbeit nachgebildet wird. In Kapitel 3 wird das AREA-Framework behandelt, welches als Grundlage des Liveguides diente. Es wird auf die Herausforderungen und Ziele des Frameworks eingegangen. Außerdem wird dessen Architektur kurz vorgestellt. Die Anforderungsanalyse wird in Kapitel 4 erläutert. Diese wird in funktionale und nicht-funktionale Anforderungen unterteilt. Dabei handelt es sich bei den funktionalen Anforderungen um Funktionen, die das Modul erfüllen soll und die nicht-funktionalen Anforderungen definieren Qualitätsmerkmale, die erfüllt werden sollen. Kapitel 5 geht auf alle wichtigen Aspekte der Realisierung ein. Zunächst werden die verwendeten Tools

und Frameworks vorgestellt und eingeordnet. Anschließend wird auf die Architektur des Moduls eingegangen. Dabei wird diese durch die Darstellung von vereinfachten UML-Klassendiagrammen aufgezeigt. Des Weiteren wird auf die wichtigsten Methoden der Skripte und Klassen eingegangen. Der Abschnitt Implementierung 5.3 erläutert vertieft die technische Umsetzung einzelner Objekte und Funktionalitäten. Zuletzt wird die Arbeit in Kapitel 6 zusammengefasst. Dabei wird ebenfalls auf Möglichkeiten eingegangen wie das Modul in Zukunft erweitert und verbessert werden kann.

2

Grundlagen

In diesem Kapitel werden einige Grundlagen und Begriffe erklärt, die zum Verständnis der Arbeit beitragen. Es wird zunächst geklärt was man unter Augmented Reality versteht und welche Rolle ein Point of Interest dabei spielt. Außerdem wird die Stadt-App der Firma cm city media vorgestellt. Dabei wird näher auf den integrierten Liveguide eingegangen, der in dieser Arbeit überarbeitet wird und als Orientierung bezüglich der Funktionsweise und der Oberflächengestaltung dient. Zum Schluss wird erklärt was man unter der AR-Erweiterung versteht, die als zusätzliche Funktion in die Anwendung eingebaut wird.

2.1 Augmented Reality

Unter Augmented Reality, auf Deutsch erweiterte Realität, versteht man eine Erweiterung der realen Welt, indem digitale Informationen oder interaktive Elemente in das Kamerabild der Umgebung integriert werden. Es entsteht also eine Kombination aus realen und virtuellen Objekten [5, 6].
Für viele Bereiche öffnet das neue Wege, um den Benutzern Informationen näher zu bringen. So dienen AR-Anwendungen zum Beispiel in Bildung, Industrie oder Medizin als Hilfestellung oder interaktive Anleitung für bestimmte Vorgänge [7]. Auch für den Alltag, vor allem im Bereich Gaming, gibt es eine große Auswahl an Anwendungen. Das 2016 erschienene Pokèmon GO ist eines der bekanntesten AR-Spiele. Der Benutzer geht dabei auf die Suche nach Pokèmons, indem er sich mit seinem Smartphone an bestimmte reale Orte begibt. Dort erscheinen die Pokèmons auf dem Bildschirm, als würden sie neben dem Benutzer stehen [6].

Für den Bereich Shopping gibt es als Beispiel die Anwendung IKEA Place. Dabei kann der Benutzer virtuelle Möbel, zum Beispiel einen Sessel wie in Abbildung 2.1, in seinem Zuhause platzieren und verschieben [5, 8].

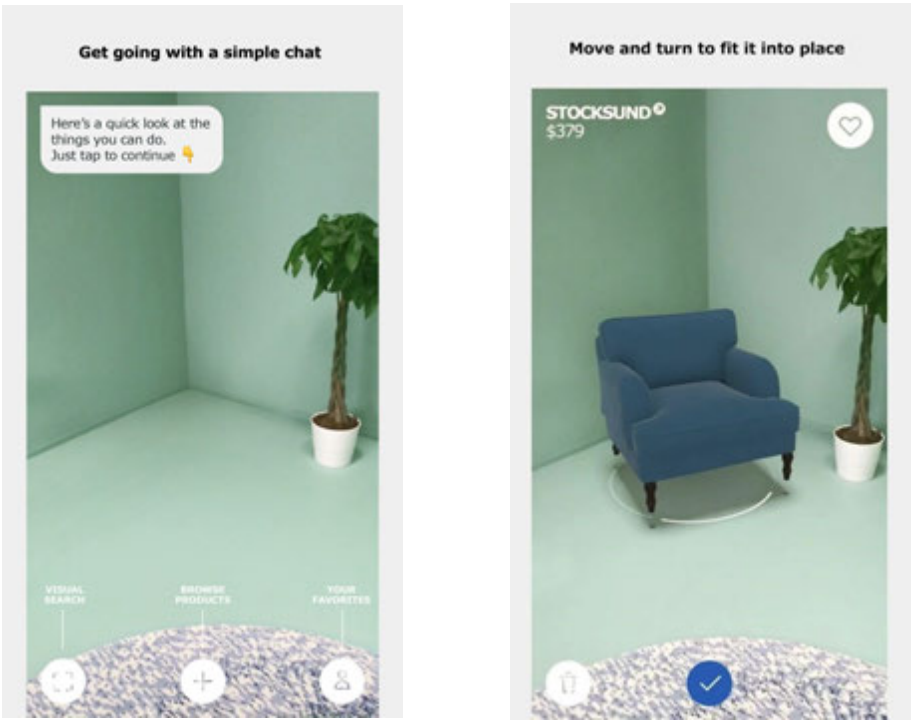


Abbildung 2.1: IKEA Place [8]

Im Bereich Tourismus können AR-Anwendungen den Benutzer durch Orte und Städte führen, indem virtuelle Informationen an wichtigen Punkten bereitgelegt werden [9]. Solch ein Punkt wird auch Point of Interest genannt und in Abschnitt 2.2 behandelt.

2.2 Points of Interest

Ein Point of Interest, kurz POI, markiert einen Ort, der für den Benutzer wichtig oder interessant sein kann. Zusätzlich können zu einem POI weitere Informationen hinterlegt sein, die dem Benutzer zur Verfügung stehen. Beispiele für POIs sind Sehenswürdigkeiten, Restaurants, Museen, Rathäuser oder Spielplätze.
In der AR-Ansicht sind POIs virtuelle Objekte, die durch die Kamera in der realen Welt

platziert werden. Die virtuellen POIs sind an dem Standort zu sehen, welchen sie repräsentieren. Sie sind also positionsabhängig und erscheinen dem Benutzer in der Kamera, wenn dieser sich in der Umgebung des Standortes befindet und in die entsprechende Richtung blickt.
Im Folgenden wird der Aufbau eines POIs beschrieben, wie er in dieser Arbeit benötigt wird. Ein POI besteht aus einem Icon, einem Namen und der Entfernung. Das Icon zeigt welcher Kategorie der Ort zugehörig ist, zum Beispiel gehört ein Krankenhaus der Kategorie Gesundheit an. Die Entfernung gibt an wie weit sich der Ort vom Benutzer entfernt befindet. Das POI in Abbildung 2.2 repräsentiert ein Krankenhaus, das in 200 Metern Entfernung lokalisiert ist.

Abbildung 2.2: Darstellung eines POIs

Befinden sich viele POIs nah aufeinander, überlappen diese sich in der Ansicht des Benutzers. Das erschwert die Identifikation und Interaktion von einzelnen POIs. Um das zu verhindern, werden Cluster erstellt, die diese POIs bündeln [10, 1]. Wie in Abbildung 2.3 verdeutlicht, werden bei einem Cluster die Icons der ersten vier POIs dargestellt. Sind dem Cluster weniger POIs zugehörig, wird die Anzahl der Icons entsprechend angepasst. Außerdem wird angegeben, wie viele Punkte in dem Cluster enthalten sind und welchen Abstand der Benutzer zu dem entferntesten und dem nächsten POI hat. Zuletzt werden die Namen aller POIs aufgelistet.
Der Benutzer hat die Möglichkeit mit einem Cluster zu interagieren, indem dieses ausgewählt wird. Anschließend werden alle POIs, die diesem Cluster zugehörig sind, in einer Übersicht aufgelistet.

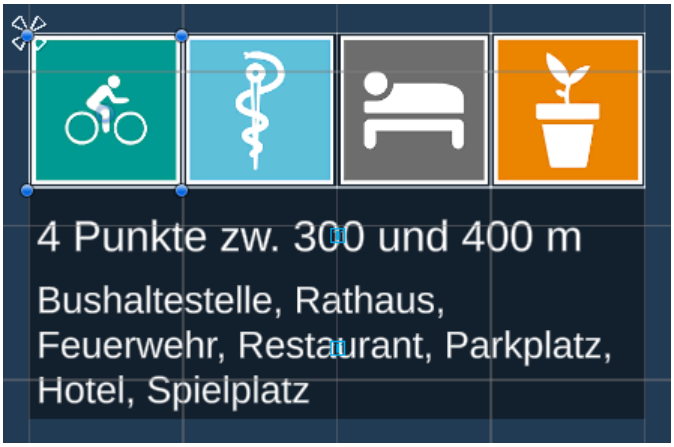


Abbildung 2.3: Darstellung eines Clusters mit vier zugehörigen POIs

Auch mit einem POI ist eine Interaktion möglich, wodurch nähere Informationen zu einem Ort abrufbar sind [9, 10]. Dabei kann ein einzelner POI oder ein POI, das durch ein Cluster aufgelistet ist, ausgewählt werden. In der aktuellen Version des Liveguides wird der Benutzer auf die Detailansicht des Ortes weitergeleitet, die alle zugehörigen Informationen aufbereitet. Hinzugefügt werden soll eine AR-Erweiterung, welche im folgenden Kapitel näher erklärt wird.

2.3 AR-Erweiterung

Die AR-Erweiterung ist eine neue Funktion, welche die Interaktion mit einzelnen POIs benutzerfreundlicher macht. Dabei bekommt der Benutzer eine Auswahl an Navigationsmöglichkeiten aufgelistet, wenn ein POI angeklickt wird. Neben der Detailansicht sollen dem Benutzer weitere Möglichkeiten gegeben werden, um Informationen zu einem Ort aufrufen zu können. Um dies zu erreichen, wird ein POI mit Verknüpfungen verlinkt. Verknüpfungen können sein: Allgemeines, Aktuelles, 360°-Bilder, Selfies, Videos, Dateien, Webseite und Kalender. Alle zugehörigen Verknüpfungen werden dem Benutzer aufgelistet, wenn dieser das POI auswählt. Dadurch kann gezielt ausgewählt werden in welcher Form nähere Informationen oder Materialien angefordert werden. Eine Verknüpfung kann den Benutzer auf eine Webseite weiterleiten, Dateien und Bilder

zur Verfügung stellen oder ein Modul der Stadt-App öffnen.
Um POIs mit Verknüpfungen in der AR-Ansicht hervorzuheben, werden diese wie in Abbildung 2.4 mit einem Plus in der rechten oberen Ecke markiert.

Abbildung 2.4: Darstellung eines POIs mit Verknüpfungen

2.4 App: Stadt sind wir

Die Stadt-App der Firma cm city media [11] für über 60 Gemeinden und Städte stehen den Nutzern von Android- und iOS-Geräten in den jeweiligen App-Stores zur Verfügung. Die Anwendung wurde dafür entwickelt Gemeinden mit ihren Bürgern und Touristen zu verbinden. Dabei bekommen die Nutzer Zugriff auf aktuelle Angebote und Informationen. Touristen und Bürger haben die Möglichkeit sich über Sehenswürdigkeiten, Veranstaltungen und vieles mehr zu informieren. Jede Stadt-App setzt sich aus den individuellen Bereichen und Angeboten der jeweiligen Gemeinde zusammen. Abbildung 2.5 zeigt dazu Beispiele der Gemeinden Bühlerzell und Lonsee. Im Folgenden wird eine Auswahl an Modulen aufgezählt, die den Nutzern auf der Startseite der Anwendung zur Verfügung gestellt werden können [4]:

- News
- Kalender
- Öffnungszeiten
- Verkehr und öffentlicher Personennahverkehr
- Karte, Orientierung und virtueller Liveguide

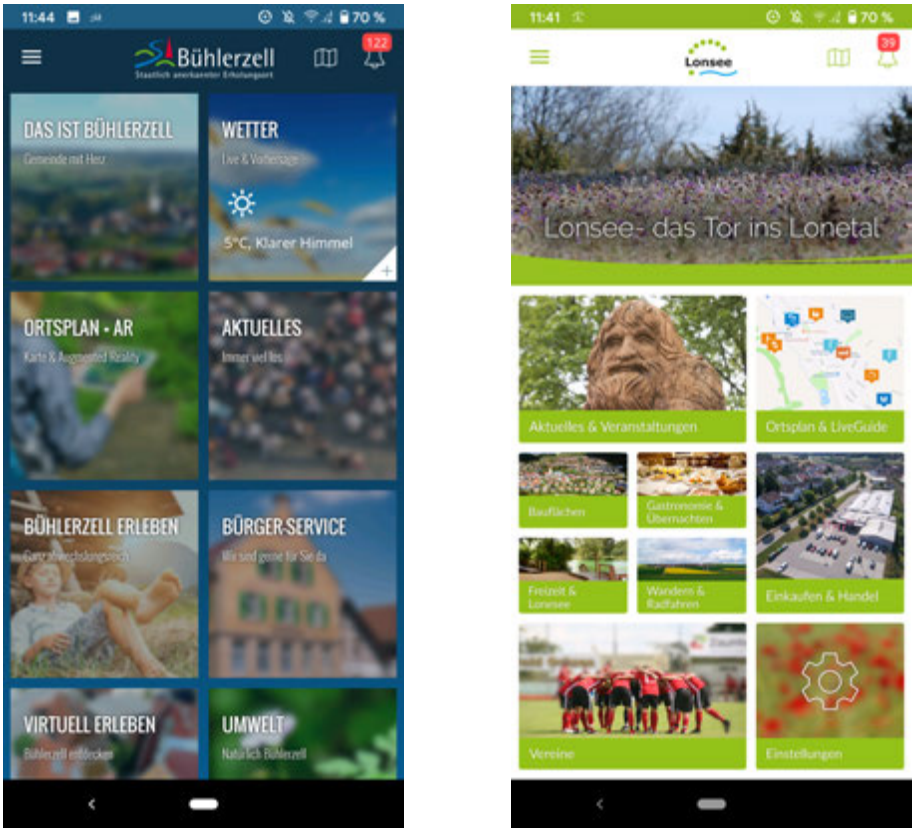


Abbildung 2.5: Startseite Stadt-App der Gemeinden Bühlerzell [12] und Lonsee [13]

Neben einer einfachen Karte, die alle wichtigen Punkte markiert, haben Benutzer auch die Möglichkeit die Gemeinde mit Hilfe des Liveguides zu erkunden. Der Liveguide zeigt eine AR-Ansicht, in der standortbezogene POIs sichtbar werden, siehe dazu Abschnitt 2.1 und 2.2. Vor dem Start des Liveguides kann der Benutzer in der Kategorieauswahl bestimmen, dass nur POIs ausgewählter Kategorien angezeigt werden sollen.
Abbildung 2.6 zeigt wie ein POI und ein Cluster dem Benutzer in der AR-Ansicht präsentiert wird.
Im Liveguide können verschiedene Interaktionen ausgeführt werden, die im Folgenden erklärt werden. Wie in Abschnitt 2.2 erwähnt, kann durch die Auswahl des POIs eine Informationsseite angefordert werden. Durch die Auswahl eines Clusters öffnet sich eine Übersicht aller zugehöriger POIs, zu sehen in Abbildung 2.7 linke Seite. Auch in dieser Übersicht kann ein einzelnes POI ausgewählt werden. Das Radar in der Ecke

rechts oben gibt die Orientierung an und zeigt außerdem eine Übersicht aller POIs in der Umgebung auf.

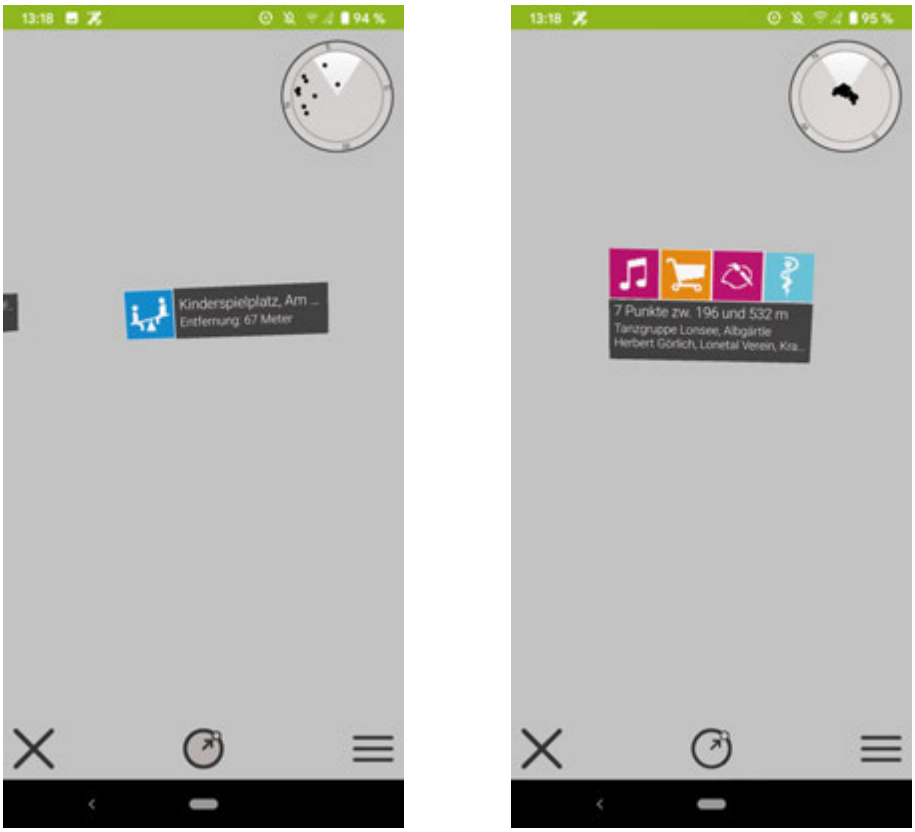


Abbildung 2.6: Liveguide Stadt-App POI und Cluster [13]

Durch den Button in der linken unteren Ecke hat der Benutzer die Möglichkeit den Liveguide zu schließen und an die Stelle zurückzukehren von wo aus er diesen geöffnet hat. Der Button in der rechten unteren Ecke öffnet die Kategorieauswahl. Um zu den Einstellungen, siehe Abbildung 2.7, zu gelangen kann der Button in der Mitte oder das Radar ausgewählt werden. Dort kann unter anderem die maximale Entfernung eingestellt werden, in der die POIs angezeigt werden sollen. Alle POIs außerhalb dieser Entfernung sind in der AR-Ansicht nicht sichtbar. Weitere Einstellungen können bezüglich der Funktion Strecken und Flächen ausgewählt werden, welche in dieser Arbeit keine Rolle spielen.

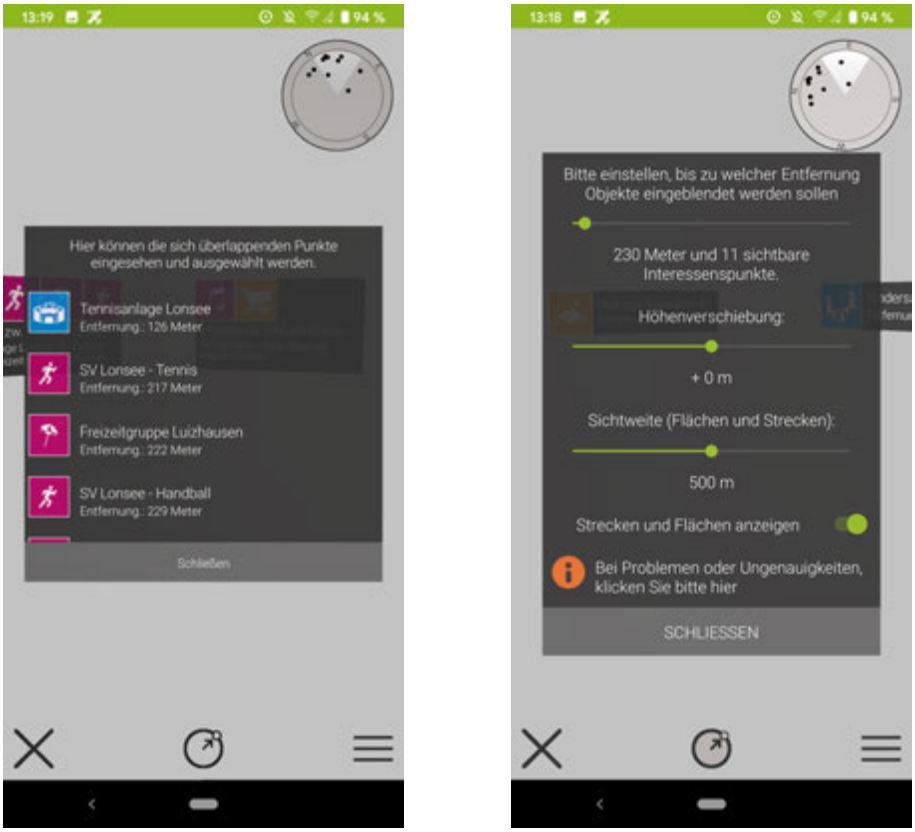


Abbildung 2.7: Liveguide Stadt-App Übersicht POIs und Einstellungen [13]

3

Verwandte Arbeiten

In diesem Kapitel wird das Framework Augmented Reality Engine Application, kurz AREA, vorgestellt. Es wurde zur Entwicklung des Liveguides der Stadt-App, welche in Kapitel 2.4 vorgestellt wurde, eingesetzt. Das Kapitel geht darauf ein, welche Funktionen darin implementiert wurden. Außerdem werden die Herausforderungen, die bei der Entwicklung bewältigt werden mussten, erläutert. Die Architektur von AREA wird ebenfalls näher vorgestellt, allerdings wird auf den Aufbau der Algorithmen nicht weiter eingegangen.

3.1 Augmented Reality Engine Application

Der Liveguide der Stadt-App, siehe Kapitel 2.4, basiert auf dem Framework AREA. Es ist in über 60 Applikationen für Gemeinden im Einsatz und steht für iOS- und Android-Geräte zur Verfügung [1]. Aufgrund der steigenden Nachfrage für Anwendungen, die Benutzer in ihrer Freizeit unterstützen und begleiten und durch die steigende Leistung der Endgeräte, entstehen neue Möglichkeiten. Das Framework AREA wurde entwickelt, um eine Lösung für solch ein Szenario zu bieten. Dabei wird der Benutzer bei der Erkundung einer Stadt oder Gemeinde unterstützt. Richtet der Benutzer die Kamera beispielsweise auf ein Gebäude, wird dieses gekennzeichnet und es können weitere Informationen dazu abgerufen werden. AREA stellt dafür die Kernfunktionen für standortbezogene, mobile AR-Anwendungen dar [10, 9]. Dazu zählen die Erkennung, Darstellung und richtige Positionierung der POIs. Außerdem werden Cluster erstellt, wenn eine Überlappung zwischen POIs stattfindet [10, 9]. Zusätzliche Funktionen wurden nach und nach implementiert. Diese umfassen Strecken und Flächen, welche in Abbildung 3.1 zu sehen sind.

Strecken sind Verbindungen von POIs durch Linien, die zur Navigation dienen. Flächen sind Markierungen von Bereichen wie zum Beispiel Fußballfeldern [9, 1].

Abbildung 3.1: AREA Strecken und Flächen [9]

Bei der Entwicklung mussten technische Herausforderungen bearbeitet werden. Darunter fällt die korrekte Positionierung und Ausrichtung der POIs in Fällen, in welchen das Gerät schräg gehalten wird. Des Weiteren zählt dazu die Erkennung von überlappenden POIs, um diese zu clustern. Eine weitere Herausforderung stellte das Ziel, dass die Anwendung auf verschiedenen Plattformen lauffähig ist [10, 9].
Für die Positionierung der POIs wird ein Koordinatensystem berechnet, das eine virtuelle 3D-Welt repräsentiert. Der Benutzer sowie die virtuelle 3D-Kamera bilden den Mittelpunkt dieser Welt. Für die Berechnung des Koordinatensystems werden die Sensordaten des Geräts herangezogen [10, 9, 1].
Die Architektur von AREA ist in Abbildung 3.2 schematisch dargestellt. Das Framework funktioniert durch ein Zusammenspiel von neun Komponenten, die im Folgenden kurz erklärt werden. Das Model verwaltet POIs, Kategorien, Cluster, Strecken und Flächen.

Funktionen, die Berechnungen innerhalb des Koordinatensystems anstellen, werden in der Komponente Math bereitgestellt. Um Zugriff auf die Sensordaten des Geräts zu haben, wird die Komponente Sensors benötigt. Die Komponente Locations ist dafür zuständig Algorithmen für die verschiedenen Koordinatensysteme bereitzustellen. Die Daten der Locations- und der Sensor-Komponente werden im Main-Controller verwaltet. Diese Daten werden benötigt, um die Funktionalitäten der Objekte des Models auszuführen. Darunter fallen die POI-Positionierung, das Clustering und das Bestimmen von Strecken und Flächen. Die View-Komponente ist anschließend zuständig für die Visualisierung der verschiedenen Objekte. Die Komponente Settings verwaltet alle Funktionen, die der Benutzer bezüglich der Visualisierung anpassen kann [10].

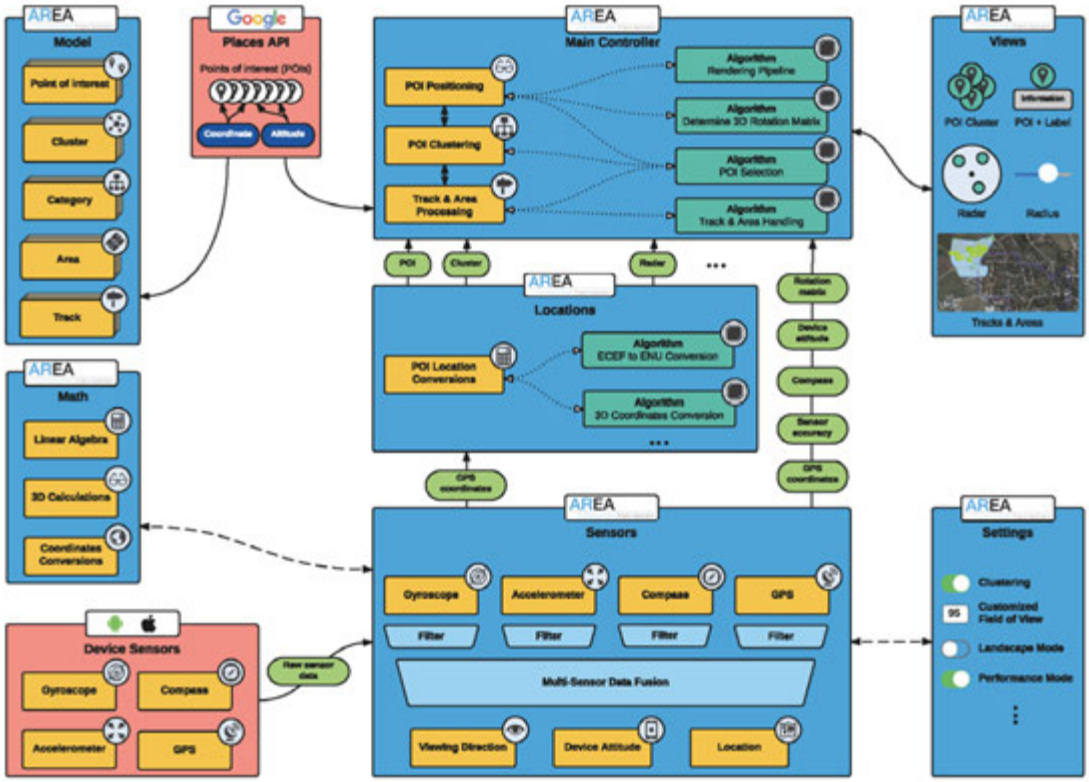


Abbildung 3.2: AREAv2 Architektur [10]

4

Anforderungsanalyse

In diesem Kapitel werden alle notwendigen Anforderungen definiert, welche als Leitfaden während der Implementierungsphase dienen. Dabei werden funktionale und nicht-funktionale Anforderungen unterschieden. Funktionale Anforderungen beschreiben Funktionen, die das Modul erfüllen sollen. Nicht-funktionale Anforderungen beschäftigen sich mit der Qualität und den Randbedingungen des Moduls.

4.1 Funktionale Anforderungen

Die funktionalen Anforderungen werden in der Tabelle 4.1 definiert.

FA#01 AR:	Der Benutzer kann seine Umgebung über die AR-Kamera erkunden.
FA#02 Darstellung von POIs:	Wichtige Orte werden durch POIs markiert. Ein POI wird durch ein Icon, einen Namen und der Entfernung zum Benutzer dargestellt.
FA#03 Darstellung von Clustern:	Befinden sich zu viele POIs an derselben Stelle, werden diese zu einer kompakten Ansicht zusammengefügt.
FA#04 Positionsabhängigkeit:	POIs und Cluster werden abhängig von deren Position dargestellt.

FA#05 Kategorien:	Jedes POI ist einer oder mehreren Kategorien zugeordnet. Jede Kategorie wird durch ein Icon repräsentiert. Ein POI zeigt das Icon einer Kategorie an.
FA#06 Radar:	Das Radar zeigt eine Übersicht aller POIs in der Umgebung des Benutzers. Dabei wird außerdem verdeutlicht welche POIs in dem Sichtfeld des Benutzers liegen.
FA#07 Beenden:	Der Benutzer kann die AR-Ansicht jederzeit beenden und zu der nativen App zurückkehren.
FA#08 Kategorieauswahl:	Ein Benutzer hat die Möglichkeit nach Kategorien zu filtern. Nur POIs aus den ausgewählten Kategorien werden sichtbar.
FA#09 Maximale Entfernung:	Der Benutzer hat die Möglichkeit in den Einstellungen anzupassen bis zu welcher Entfernung POIs angezeigt werden soll.
FA#10 Zurücksetzen:	Der Benutzer hat die Möglichkeit in den Einstellungen die AR-Werte neu zu laden.
FA#11 Clusteransicht:	Wird ein Cluster ausgewählt, werden alle zugehörigen POIs aufgelistet.
FA#12 Detailansicht:	Wird ein POI ohne Verknüpfungen ausgewählt, wird dem Benutzer dessen Detailansicht angezeigt. Diese zeigt dem Benutzer zusätzliche Informationen an.

FA#13 Verknüpfungen:	Wird ein POI mit Verknüpfungen ausgewählt, wird eine Übersicht aller Verknüpfungen angezeigt. Ein POI kann mindestens zwei und maximal acht Verknüpfungen besitzen.
FA#14 Auswahl einer Verknüpfung:	Bei der Auswahl einer Verknüpfung wird die AR-Ansicht geschlossen und die dazu definierte Aktion ausgeführt. Dabei kann der Benutzer zu der nativen App, zu einer Webseite oder einer anderen Anwendung weitergeleitet werden.

Tabelle 4.1: Funktionale Anforderungen

4.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen werden in der Tabelle 4.2 definiert.

NFA#01 Betriebssystem:	Das Unity-Modul kann in die native Android- sowie in die native iOS-App exportiert werden.
NFA#02 Stabilität:	Die AR-Objekte werden stabil angezeigt und ein Springen der Objekte durch ständige Positionsänderungen wird vermieden.
NFA#03 Präzision:	Die Position der AR-Objekte wird so präzise wie möglich bestimmt.
NFA#04 Verfügbarkeit:	Das Unity-Modul ist zu jeder Zeit abrufbar, wenn die Standorterkennung aktiviert ist und das Gerät mit ARCore oder ARKit kompatibel ist .

NFA#05 Effizienz:	Die Darstellung und Berechnung der AR-Objekte erfolgt möglichst effizient.
NFA#06 Design:	Das Design gleicht dem AREA-Liveguide.

Tabelle 4.2: Nicht-funktionale Anforderungen

5

Realisierung

In diesem Kapitel liegt der Fokus auf der Umsetzung des Unity-Moduls. Es werden verwendete Tools wie die Entwicklungsumgebung Unity und das Framework AR-Foundation vorgestellt. Die Architektur des Moduls wird anhand von Klassendiagrammen dargestellt und erklärt. Anschließend wird die Umsetzung einzelner Komponenten und Algorithmen im Abschnitt 5.3 erläutert.

5.1 Verwendete Tools

Dieses Kapitel stellt die Arbeitsmittel vor, mit welchen die Implementierung umgesetzt wurde. Als Entwicklungsumgebung wurde die Echtzeit-Entwicklungsplattform Unity zusammen mit Microsoft Visual Studio verwendet. Für die Einbindung von Augmented Reality kam das Framework AR-Foundation zum Einsatz, das die AR-Frameworks ARCore von Google und ARKit von Apple vereint. Das Unity Asset AR+GPS Location vereinfacht das Setzen von geografischen Positionen der AR-Objekte.

5.1.1 Entwicklungsumgebung: Unity

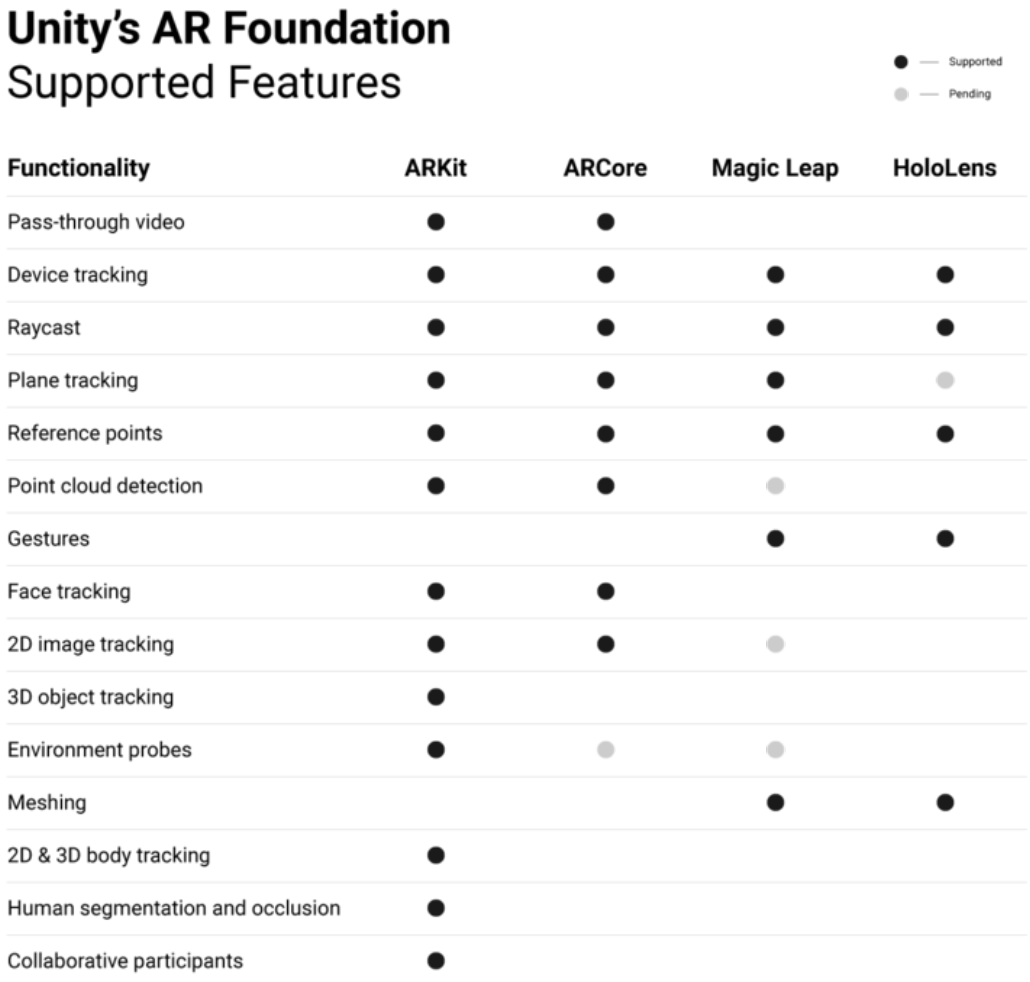
Die Umsetzung erfolgte mit Unity 2019.3 Personal. Unity ist eine Echtzeit-Entwicklungsplattform, die unter anderem für die 2D- und 3D-Spielentwicklung zur Anwendung kommt. Darunter fallen auch Anwendungen im Bereich virtuelle oder erweiterte Realität. Unity bietet eine Reihe an Entwicklungsplattformen an, welche alle mit demselben Projekt bedient werden können. Solche Plattformen sind Android, iOS, Windows, die Spielkonsolen PS4 und XBOX One, aber auch das Web [14].

Durch die integrierte Physik-Engine wird die Entwicklung von Schwerkraft, Bewegung, Licht und vieles mehr unterstützt. Anwendungen in Unity setzen sich aus beliebig vielen, aber mindestens einer Szene zusammen. Dabei enthält jede Szene eine eigene Umgebung und eigenen Menüelemente. Die Entwicklung der Objekte basiert auf der Beziehung zwischen Gameobjects und Komponenten. Ein Gameobject ist die Basisklasse für alle Entitäten und hat selbst keine Funktionalität. Diese sowie weitere Modifikationen werden dem Gameobject mit Hilfe von Komponenten hinzugefügt. Die Transform-Komponente, zum Beispiel, bestimmt Position und Skalierung des Objekts. Komponenten werden nicht hierarchisch angelegt, sondern sind als Liste in dem jeweiligen Gameobject gespeichert. Um dem Gameobject ein individuelles Verhalten zuzuweisen, werden Skripte erstellt. Diese sind in Unity in C# programmiert und können mit einer Entwicklungsumgebung wie zum Beispiel Microsoft Visual Studio bearbeitet werden. Mit Hilfe von Skripten können ebenfalls Gameobjects erstellt und Komponenten hinzugefügt oder gelöscht werden. Bei einem entsprechenden Zugriff auf Objekte können beliebige Werte über Skripte manipuliert werden [15, 14].
Für die Verwendung von eigenen Bildern oder Icons werden sogenannte Sprites verwendet. Einem Bild in JPG- oder PNG-Format kann im Unity-Editor der Texture Type Sprite zugewiesen werden. Dadurch kann das Bild einer Image- oder SpriteRenderer-Komponente hinzugefügt werden. Dieses Vorgehen kann für UI-Elemente oder 2D-Gameobjects genutzt werden [14].

5.1.2 Framework: AR-Foundation

Für die Implementierung der erweiterten Realität wird das Framework AR-Foundation von Unity eingesetzt. Das Framework bietet die Möglichkeit AR-Szenarien für verschiedene Endgeräte zu entwickeln. AR-Foundation greift dazu auf die Kernfunktionen von ARKit und ARCore zurück und implementiert selbst keine AR-Funktionen. ARCore ist eine Plattform von Google, die es ermöglicht AR-Anwendungen für Android-Geräte zu implementieren [16]. ARKit erlaubt die Implementierung von AR-Erlebnissen auf iOS-Geräten [17]. Um die AR-Foundation Anwendung auf den Endgeräten nutzen zu können, müssen ARCore oder ARKit auf dem entsprechendem Gerät verfügbar und

installiert sein.
AR-Foundation unterstützt verschiedene Konzepte, auf die der Entwickler zugreifen kann. Beispiele dafür sind das World Tracking, das die Position und Orientierung des Geräts im physischen Raum erkennt, außerdem die Ebenenerkennung horizontal sowie vertikal oder die Erkennung von Gesichtern, 2D Bildern und 3D Objekten [18, 19]. Die Tabelle in Abbildung 5.1 verdeutlicht welche Konzepte für welche Plattformen zur Verfügung stehen.



Unity's AR Foundation
Supported Features

● Supported
● Pending

Functionality	ARKit	ARCore	Magic Leap	HoloLens
Pass-through video	●	●		
Device tracking	●	●	●	●
Raycast	●	●	●	●
Plane tracking	●	●	●	○
Reference points	●	●	●	●
Point cloud detection	●	●	○	
Gestures			●	●
Face tracking	●	●		
2D image tracking	●	●	○	
3D object tracking	●			
Environment probes	●	○	○	
Meshing			●	●
2D & 3D body tracking	●			
Human segmentation and occlusion	●			
Collaborative participants	●			

Abbildung 5.1: AR-Foundation Funktionen [18]

In Unity müssen neben dem Package AR-Foundation auch ARKit XR Plugin und ARCore XR Plugin installiert werden.
Eine AR-Szene mit AR-Foundation besteht aus drei Basiselementen. Der hierarchische Aufbau dieser ist in Abbildung 5.2 zu sehen. Die Objekte sind AR-Session, AR-Session Origin und AR-Kamera. Welche Komponenten zu den jeweiligen Objekten implementiert sind, verdeutlicht Abbildung 5.3 in einem Klassendiagramm. Die wichtigsten Funktionalitäten werden im Folgenden für jede Klasse erklärt.

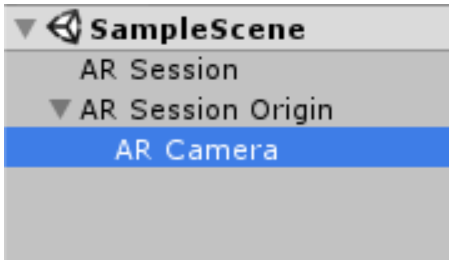


Abbildung 5.2: AR-Foundation Szene [19]

AR-Session: Die AR-Session bildet die AR-Instanz und kontrolliert den Lebenszyklus auf dem Gerät. Ist die benötigte AR-Software auf dem Gerät nicht kompatibel oder installiert, weist das Gerät den Nutzer darauf hin und installiert, wenn möglich, benötigte Komponenten [19].

AR-Session Origin: Das Objekt AR-Session Origin ist für die korrekte Position, Orientierung und Skalierung der AR-Objekte in der realen Welt zuständig [19]. AR-Objekte sind in diesem Fall POIs und Cluster.

AR-Kamera: Die AR-Kamera rendert den virtuellen Inhalt, also die AR-Objekte, in die reale Welt, die durch die Geräte-Kamera sichtbar ist. Die Komponente ARPoseDriver steuert die korrekte Position und Ausrichtung der Kamera in der AR-Szene abhängig von den Trackinginformationen des Geräts [19].

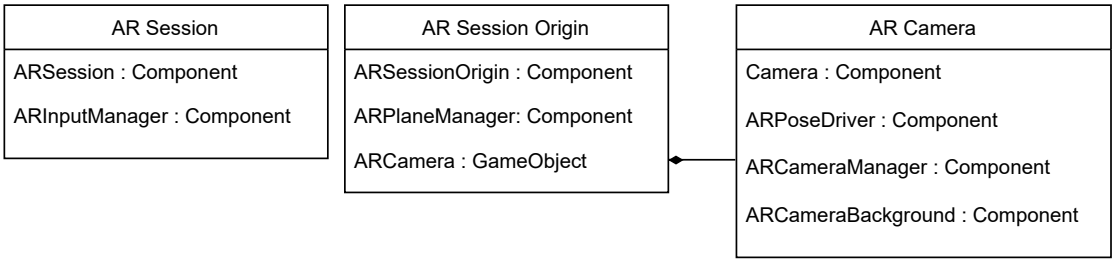


Abbildung 5.3: UML-Klassendiagramm einer einfachen AR-Foundation Szene

5.1.3 Plugin: AR+GPS Location

AR+GPS Location ist ein Asset, das über den Unity Asset Store in das Projekt eingebunden werden kann. Das Asset bietet die Funktion AR-Objekte mit GPS-Koordinaten bestehend aus der Höhe, Längen- und Breitengrade auszustatten. Die Objekte können somit an einer bestimmten Position in der echten Welt platziert werden. Weitere Funktionalitäten sind unter anderem Hotspots, welche die Objekte erst aktivieren, wenn der Benutzer sich in einem bestimmten Radius dazu befindet. Außerdem ist es möglich Pfade für AR-Objekte zu definieren, an welchen diese sich entlang bewegen. Für diese Arbeit werden den AR-Objekten, POIs und Clustern, eine feste Position zugeteilt.
Das Plugin berechnet die Position und Orientierung der AR-Objekte relativ zu der AR-Kamera, also zu dem Benutzer. In Abbildung 5.4 wurde die Struktur der AR-Foundation Szene erweitert.

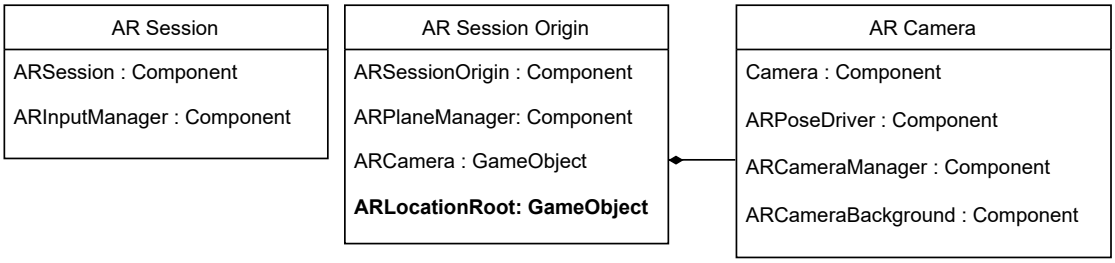


Abbildung 5.4: UML-Klassendiagramm mit der Erweiterung AR+GPS Location

Auf gleicher Ebene zu der AR-Kamera wurde das Objekt ARLocationRoot hinzugefügt. Dieses Objekt ist zuständig für die Berechnung der geografischen Richtung der AR-Kamera. Sie ist notwendig, um die AR-Objekte korrekt positionieren zu können [20].
Bei der Verwendung des Plugins müssen dessen Limitationen berücksichtigt werden. Die Angabe der Höhe für AR-Objekte ist ungenau und sollte relativ zu dem Gerät angegeben werden, wenn es benötigt wird. Außerdem sollte die Anwendung nur im Portrait-Modus verwendet werden, da die Sensordaten im Landscape-Modus ungenau sind. Die GPS-Daten werden über die Sensoren des Gerätes ermittelt und können ungenaue Werte aufweisen, wodurch es passieren kann, dass Objekte nicht exakt an der angegebenen Position erscheinen. Dies ist vor allem der Fall, wenn Objekte in Gebäuden platziert werden. Es wird empfohlen dies zu vermeiden [20].
Das Ziel ist es die AR-Objekte möglichst präzise und möglichst stabil zu positionieren. Für eine hohe Präzision werden allerdings viele Messungen benötigt, wodurch die Position der Objekte bei jeder neuen Messung angepasst wird. Durch die vielen Veränderungen erscheint die Positionierung als instabil. Um eine möglichst hohe Stabilität zu erreichen, wird die Anzahl der Positionsupdates so gering wie möglich gehalten. Es werden bestimmte Intervalle festgelegt, nach welchen eine Positionsänderung stattfinden soll. Dadurch wird verhindert, dass Anpassungen nach jeder Positions- und Orientierungsänderung durchgeführt werden [20]. Festgelegt wird, dass Änderungen entweder nach 60 Sekunden oder nach zehn Metern zurückgelegter Strecke überprüft werden.

5.2 Architektur der Anwendung

Das Kapitel 5.2 stellt die verwendeten Klassen in Form von UML-Klassendiagrammen vor. Abbildung 5.5 und Abbildung 5.6 zeigen die Beziehungen zwischen einem POI-Gameobject beziehungsweise einem Cluster-Gameobject und deren zugehörigen Komponenten. Wie Gameobjects und Komponenten miteinander in Verbindung stehen, wurde in dem Abschnitt 5.1.1 behandelt. In Abbildung 5.7 werden die Klassen vorgestellt, welche Aktionen, Funktionen und Berechnungen implementieren. Das Klassendiagramm in Abbildung 5.8 verdeutlicht den Aufbau der POI-Verknüpfungen.
Alle Diagramme wurden zur besseren Verständlichkeit vereinfacht und bilden nicht alle Variablen oder Methoden ab.

Point of Interest

Ein POI wird in Unity als ein Gameobject repräsentiert. Dieses besteht aus zwei weiteren Gameobjects von welchen eines den Punkt des POIs im Radar darstellt und eines den Marker in der AR-Ansicht. Wie in Abbildung 5.5 abgebildet, sind die drei wichtigsten Komponenten die POI-Komponente, die PlaceAtLocation-Komponente und die AlignToCamera-Komponente. Letztere wird benötigt, um das einzelne POI zum Benutzer auszurichten, sodass diesem die Vorderansicht des POIs angezeigt wird. Die Daten, die das POI definieren wie zum Beispiel den Namen, die Distanz zum Benutzer und die Position, sind in der POI-Komponente abrufbar. Die PlaceAtLocation-Komponente wird von dem Plugin AR+GPS Location bereitgestellt und setzt die geografische Position des POIs.

Cluster

Jedes Cluster wird in Unity als ein Gameobject repräsentiert. Dieses besteht unter anderem aus einer Liste an weiteren Gameobjects, welche bis zu vier Icons der zugehörigen POIs anzeigen. Die Komponenten des Cluster-Gameobjects sind ähnlich wie die des POI-Gameobjects. Die Cluster-Komponente bildet die Entität des Clusters und

speichert alle notwendigen Informationen, die zur Bildung eines Cluster-Gameobjects benötigt werden. Abbildung 5.6 zeigt, dass die Cluster-Komponente unter anderem alle zugehörigen POI-Gameobjects, die Position des Clusters und die maximale sowie die minimale Distanz der POIs zum Benutzer definiert.

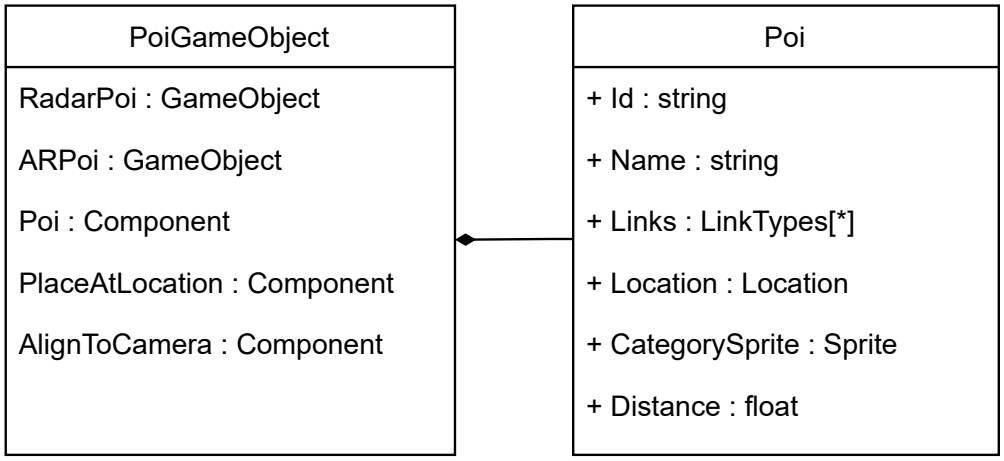


Abbildung 5.5: UML-Klassendiagramm Beziehung zwischen dem POI-Gameobject und der POI-Komponente

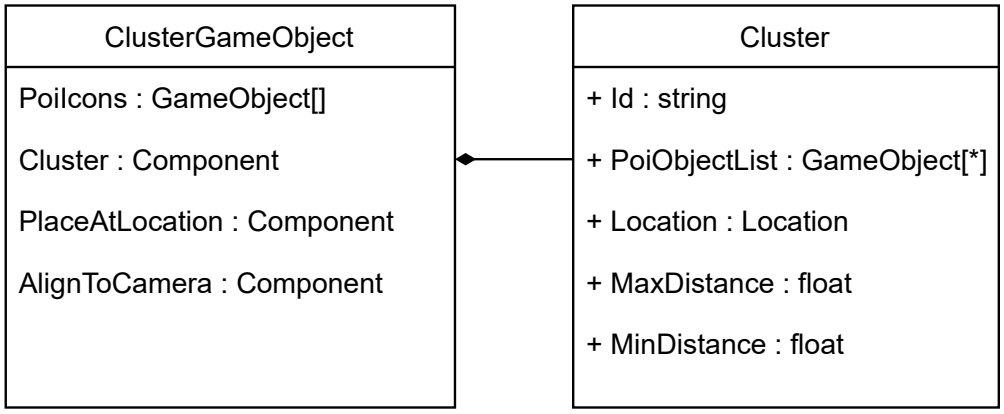


Abbildung 5.6: UML-Klassendiagramm Beziehung zwischen dem Cluster-Gameobject und der Cluster-Komponente

Handler

Die Handler-Klassen implementieren Aktionen, Funktionen und Berechnungen, sodass POIs und Cluster korrekt dargestellt werden. Außerdem werden Interaktionen mit Buttons, den Einstellungen oder den AR-Objekten behandelt. Die Handler-Klassen sind zusammengesetzt aus PoiHandler, SettingsHandler, SelectionHandler, ButtonHandler und RadarHandler, siehe Abbildung 5.7. Im Folgenden wird die Aufgabe der einzelnen Klassen und deren Methoden genauer beschrieben.

PoiHandler: Der PoiHandler steuert die Darstellung, Erstellung und Berechnung der POIs und der Cluster.

InitializePoiGameObject: Das native Liveguide-POI wird zu einem Unity-POI umgewandelt und einem POI-Gameobject zugeordnet. Das POI-Gameobject wird gesetzt und aktiviert.

FindClusters: Für jedes POI wird überprüft, ob es sich mit einem anderen Cluster oder POI überschneidet. Überschneidet es sich mit einem Cluster, wird es diesem hinzugefügt. Überschneidet es sich mit einem anderen POI entsteht ein neues Cluster, siehe dazu Abschnitt 5.3.2.

CreateClusterUI: Wenn ein neues Cluster aktiviert wird, wird die minimale und maximale Distanz der POIs berechnet. Die UI des Clusters wird den Werten entsprechend angepasst.

UpdateClusterUI: Wenn ein POI einem Cluster hinzugefügt wird, werden die Werte, wie Anzahl an POIs, maximale und minimale Entfernung der POIs, in der UI aktualisiert.

CalculateClusterPosition: Wenn ein Cluster aktiviert oder aktualisiert wird, wird die neue Position des Clusters berechnet, indem der Mittelpunkt aller zugehöriger POIs bestimmt wird.

CalculateAngle: Für die Überprüfung, ob sich ein POI mit einem Cluster oder einem anderen POI überlappt, wird der Winkel zwischen dem linken und dem rechten Ende des POIs oder des Clusters zum Benutzer bestimmt. Liegt ein anderes Objekt in diesem Bereich, findet eine Überlappung statt, siehe dazu Abschnitt 5.3.2.

SettingsHandler: Der SettingsHandler reagiert auf Änderungen der Einstellungen.

OnDistanceSliderValueChanged: Wird in den Einstellungen die maximale Distanz, in der POIs angezeigt werden sollen, geändert, wird die Anzahl aller sichtbaren POIs berechnet.

CloseSettingsPanel: Werden die Einstellungen geschlossen, wird überprüft, ob Änderungen vorgenommen worden sind. Ist dies der Fall, wird die Berechnung der Darstellung von POIs und Cluster neu durchgeführt.

SelectionHandler: Der SelectionHandler reagiert auf die Touch-Auswahl von POIs und Cluster.

ClusterClicked: Wenn ein Cluster ausgewählt wird, öffnet sich die Cluster-Übersicht, in der alle zugehörigen POIs aufgelistet sind.

PoiClicked: Wenn ein POI ausgewählt wurde und dieses Verknüpfungen besitzt, werden diese aufgelistet. Besitzt das POI keine Verknüpfungen wird das Unity-Modul geschlossen und die Detailansicht des POIs angezeigt.

SetLinkUI: Werden die Verknüpfungen des POIs angezeigt, wird für jede Verknüpfung ein Button aktiv, der die jeweilige Aktion der Verknüpfung auslösen kann.

ButtonHandler: Der ButtonHandler verwaltet die Aktionen, die ausgelöst werden, wenn ein Button geklickt wird.

Reset: Der LocationProvider und der LocationManager werden neu gestartet, sodass die Position des Benutzers neu gesetzt wird.

Close: Das Unity-Modul wird geschlossen.

ShowCategoryFilter: Das Unity-Modul wird geschlossen und die Kategorieauswahl der nativen App wird angezeigt.

OpenSettingsPanel: Die Einstellungen werden aktiviert und alle notwendigen Werte dafür gesetzt.

RadarHandler: Der RadarHandler steuert die Radar-Kamera und die Radar-Ansicht.

SetRadarUI: Die Radar-Kamera wird ausgerichtet und orientiert sich an der AR-Kamera. Dabei positioniert sich die Radar-Kamera immer über der AR-Kamera und blickt in deren Richtung. Die Sichtweite und die Höhe der Radar-Kamera wird abhängig von der maximalen, eingestellten Distanz berechnet. Die Größe der POI-Punkte wird ebenfalls anhand dieser Werte berechnet, siehe dazu Abschnitt 5.3.4.

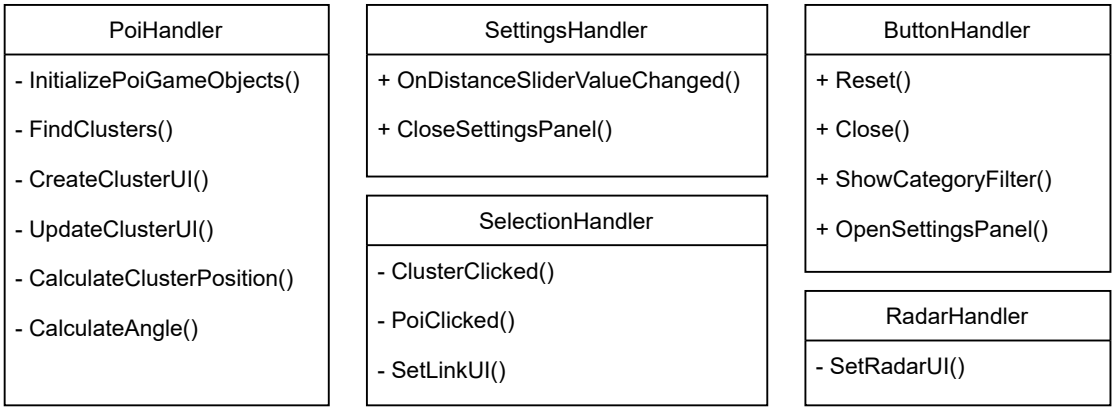


Abbildung 5.7: UML-Klassendiagramm der Handler-Klassen

Verknüpfung

Jedes POI besitzt eine Liste, die definiert welche Verknüpfungen mit diesem verlinkt sind, siehe dazu Abbildung 5.5. In der Liste ist gespeichert, welcher Verknüpfungstyp dem POI zugehörig ist. Das Interface ILink gibt vor, dass jede Verknüpfung die ID des POIs sowie das Icon und die Bezeichnung der Verknüpfung festlegt. Auch die Funktion, die bei der Auswahl der Verknüpfung ausgelöst wird, wird von dieser definiert. Für diese Aktion wird die ID des POIs benötigt. Eine mögliche Aktion kann das Aufrufen einer Webseite abhängig des POIs sein.
Das Klassendiagramm in Abbildung 5.8 zeigt den Aufbau der Verknüpfungen. Es ist möglich, diesen beliebig zu erweitern.

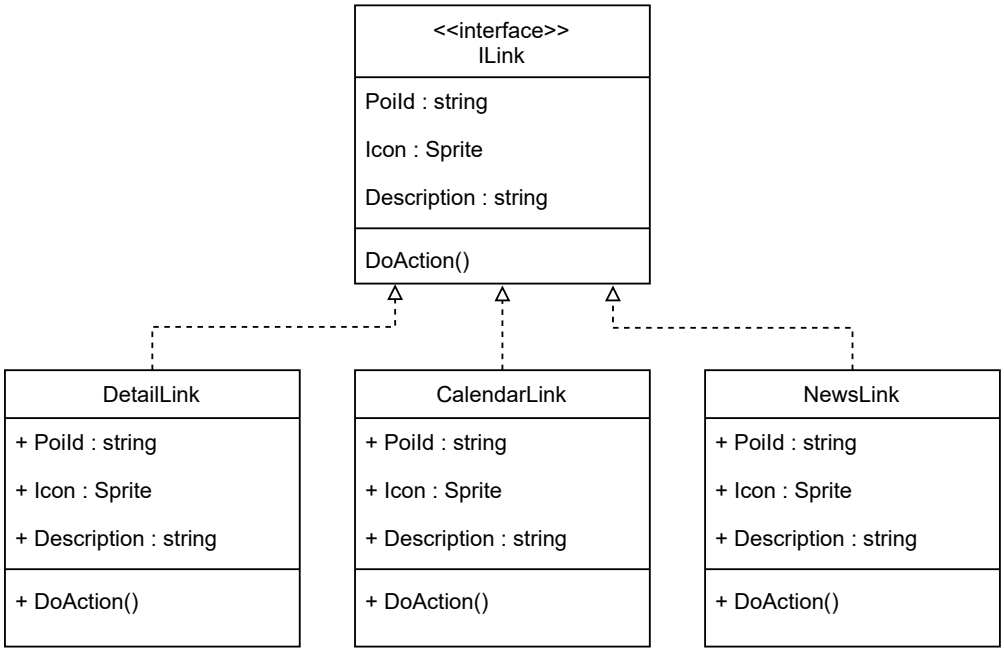


Abbildung 5.8: UML-Klassendiagramm der Verknüpfungen eines POIs

5.3 Implementierung

Im Folgenden wird die technische Umsetzung des Unity-Moduls genauer erläutert. Dabei werden die Algorithmen einiger Funktionen vorgestellt, welche in Abschnitt 5.2 bereits erwähnt wurden. Das Kapitel dient dazu, ein tieferes Verständnis über die Implementierung des Unity-Moduls zu vermitteln.

5.3.1 Points of Interest

Die Implementierung der Points of Interest wurde in zwei Schritten ausgeführt. Als Erstes wurden die POIs ohne Angabe einer Position der realen Welt in dem Koordinatensystem platziert. Als Zweites wurde den POIs jeweils einen Standort zugeteilt, sodass diese abhängig davon platziert werden.

Positionsunabhängige Darstellung eines POIs

Ein POI ist im Kontext des Unity-Moduls ein 2D-Objekt, das durch ein Icon definiert ist. Dafür wird ein Gameobject mit der Komponente SpriteRenderer benutzt. Das Icon wird als Sprite in das Projekt importiert und mit Hilfe des SpriteRenderers dem Gameobject zugeordnet. Für eine positionsunabhängige Darstellung der POIs werden die POI-Gameobjects in der Umgebung des Kamera-Objekts platziert, indem sie in dem Koordinatensystem an eine beliebige Position gesetzt werden, siehe Abbildung 5.9. Für die richtige Ausrichtung der einzelnen POIs werden diese in die Richtung der AR-Kamera und somit in die des Benutzers rotiert.

Positionsabhängige Darstellung eines POIs

Um die POIs positionsabhängig darzustellen, wird jedem POI eine Position in Form von Längen- und Breitengrade zugeordnet. Diese Daten werden beim Start des Unity-

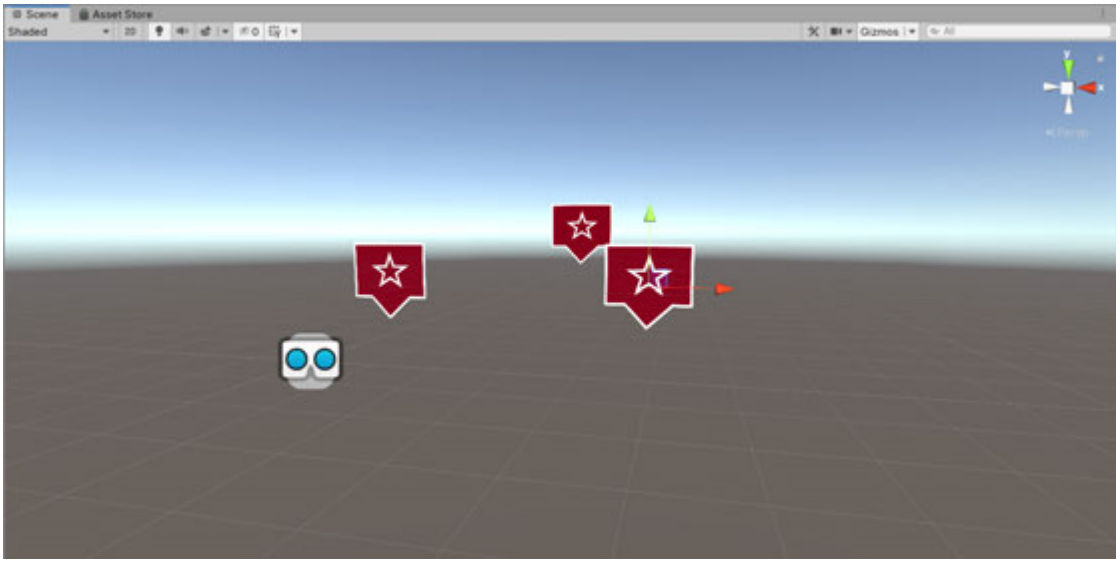


Abbildung 5.9: Darstellung positionsunabhängiger POIs

Moduls von der nativen App an das Modul in JSON[1]-Format übertragen. Somit hat das Unity-Modul Zugriff auf die relevanten Daten der POIs.
Für jeden Datensatz wird anschließend ein POI-Gameobject aktiviert. Die Position sowie die UI werden den POI-Daten angepasst. Jedem POI-Gameobject wird eine POI-Komponente zugeteilt, welche die Verfügbarkeit aller Attribute sicherstellt.

Nachdem alle POI-Gameobjects initialisiert wurden, wird im nächsten Schritt untersucht, welche Objekte geclustert werden müssen.

5.3.2 Cluster

Ein Cluster ist ebenfalls, wie ein POI, ein 2D-Objekt und wird eingesetzt, sobald sich zwei POIs in der AR-Ansicht überschneiden. Um dies zu bestimmen, wird der Cluster-Algorithmus angewendet, der in Abbildung 1 zu sehen ist.
Es wird für jedes POI zunächst überprüft, ob es sich innerhalb des maximal eingestellten Radius um den Benutzer befindet, in welchem POIs angezeigt werden sollen. Ist

[1] JavaScript Object Notation

dies nicht der Fall, wird das entsprechende Gameobject deaktiviert, das somit für den Benutzer nicht mehr sichtbar ist. Liegt das POI in diesem Radius wird es mit jedem Cluster verglichen, das bereits existiert. Dafür wird für beide Objekte der Bereich bestimmt, in dem ein anderes Objekt ganz oder nur zum Teil liegen muss, sodass es zu einer Überschneidung kommt. In Abbildung 5.10 ist das der grau-markierte Bereich. Um dies zu Bestimmen wird der Winkel α berechnet, siehe 5.2. Dazu benötigt wird die Distanz d zwischen dem Benutzer und dem POI, die Länge der Seiten s1 und s2 und die Breite w des Objekts, siehe 5.1. Es gilt s1 = s2, da das POI-Objekt immer zur Kamera ausgerichtet wird.

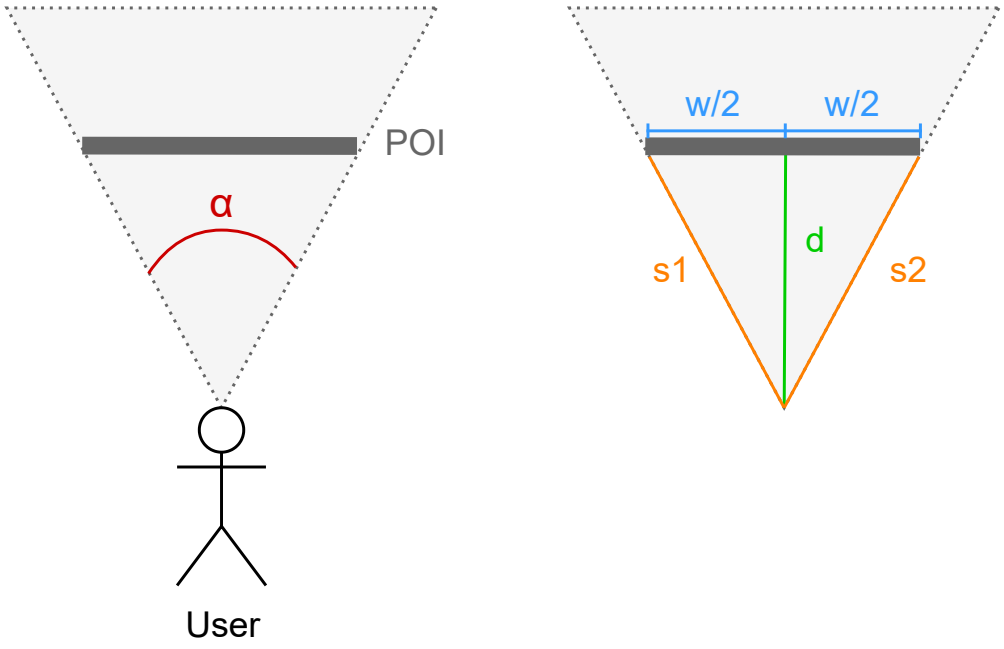


Abbildung 5.10: Die Fläche eines POIs, in der bei Überlappung ein Cluster gebildet wird

Folgende Berechnungen werden angestellt:

$$s = \sqrt{(w/2)^2 + d^2} \tag{5.1}$$

$$\alpha = \arccos\left(\frac{2 \cdot s^2 - w^2}{2 \cdot s^2}\right) \cdot \frac{360}{\Pi \cdot 2} \tag{5.2}$$

Der Winkel α wird für beide Objekte berechnet, die miteinander verglichen werden. Daraus kann der maximale Winkel γ bestimmt werden, der zwischen den Objekten liegen muss, bis zu welchem eine Überlappung stattfindet. Dieser bildet einen Grenzwert. Abbildung 5.11 verdeutlicht, dass sich der maximale Winkel aus der Summe von $\frac{\alpha 1}{2}$ und $\frac{\alpha 2}{2}$ ergibt. Es gilt $\gamma = \frac{\alpha 1}{2} + \frac{\alpha 2}{2}$.

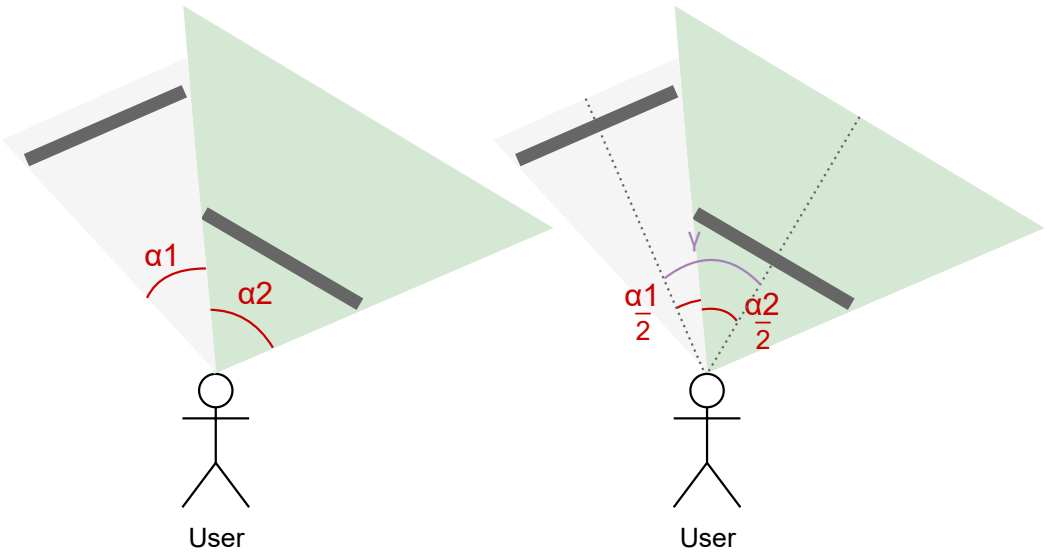


Abbildung 5.11: Darstellung zweier AR-Objekte, die den maximalen Abstand aufweisen, bei welchem eine Überlappung stattfindet

Um nun die Objekte auf eine Überschneidung zu überprüfen, wird der Winkel β berechnet, siehe Abbildung 5.12 links. Dieser liegt zwischen den Mittelpunkten beider Objekte. Der Winkel β wird anschließend mit dem Winkel γ verglichen, da dieser den Grenzwert einer Überlappung bildet. Auf der rechten Seite der Abbildung 5.12 wird die Gegenüberstellung beider Winkel verdeutlicht.
Tritt der Fall ein, dass der Winkel β kleiner ist als der Winkel γ liegt eine Überschneidung beider Objekte vor. Anschließend wird das POI dem Cluster hinzugefügt. Die neue Position des Clusters wird berechnet, indem der Mittelpunkt aller POI-Positionen bestimmt wird.

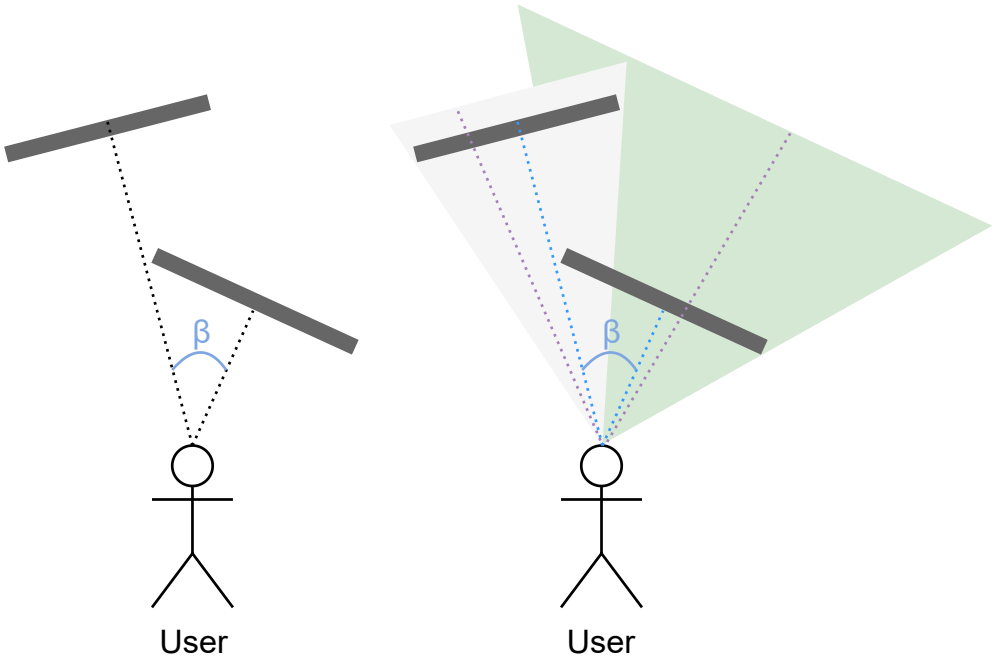


Abbildung 5.12: Darstellung zweier überlappender AR-Objekte und des Vergleichs der Winkel β und γ

Für eine genaue Berechnung und für die Vermeidung von Fehlberechnungen bei Randwerten mit Längen- und Breitengraden, werden diese in Vektoren umgerechnet. Die Längen- und Breitengrade sind in Radiant angegeben.

$$x = \cos(latitude) \cdot \cos(longitude) \tag{5.3}$$

$$y = \cos(latitude) \cdot \sin(longitude) \tag{5.4}$$

$$z = \sin(latitude) \tag{5.5}$$

Um den Mittelwert zu berechnen, werden die Koordinaten aller POIs, die sich in dem Cluster befinden aufsummiert und durch deren Anzahl dividiert. Das Ergebnis wird anschließend in Längen- und Breitengrade zurück gewandelt.

$$longitude = \arctan 2(y, x) \tag{5.6}$$

$$latitude = \arctan 2\left(z, \sqrt{x^2 + y^2}\right) \tag{5.7}$$

Wird das POI keinem Cluster hinzugefügt, so wird überprüft, ob Überschneidungen mit anderen POIs vorliegen. Die Überprüfung zwischen zwei POIs ist die gleiche wie die zwischen einem POI und einem Cluster. Wenn eine Überschneidung gefunden wird, wird ein neues Cluster-Objekt aktiviert. Die POIs werden diesem zugeteilt und das zugehörige Gameobject deaktiviert. Die Position des Clusters ist der Mittelpunkt beider POI-Positionen.
Werden für das POI keine Überschneidungen gefunden, so wird das POI einzeln dargestellt.
Die Reihenfolge der Überprüfungen hat zum Vorteil, dass neu erstellte Cluster nicht erneut mit anderen Objekten überprüft werden müssen, da eine Überschneidung der zugehörigen POIs mit anderen Clustern bereits geprüft wurde. Eine Überlappung zu den noch nicht überprüften einzelnen POIs wird anschließend ausgehend von diesen nachgeprüft.

```
Data: poi-gameobjects
Result: create clusters or activate single pois
foreach poi-gameobject do
    get distance between user and poi;
    if poi is too far away then
        deactivate poi;
    end
    else if poi is not clustered then
        foreach cluster-gameobject do
            check angle between cluster-gameobject and poi-gameobject;
            if angle is too small then
                add poi to cluster;
                calculate new position of cluster;
                deactivate single poi;
            end
        end
        if poi is not clustered then
            foreach following poi-gameobject do
                check angle between both pois;
                if angle is too small then
                    create cluster;
                    calculate position of cluster;
                    deactivate both single pois;
                end
            end
        end
        if poi is not clustered then
            poi is single poi;
        end
    end
end
```

Algorithm 1: Cluster-Algorithmus

5.3.3 Object Pooling

Für eine möglichst gute Perfomance und um lange Ladezeiten bei der Berechnung und Erstellung der POIs und Cluster zu vermeiden, wird eine dynamische Speicherverwaltung verwendet. Das Instantiieren und Zerstören von Objekten während der Laufzeit benötigt viel Speicher, wodurch die Anwendung verlangsamt wird. Um das zu verhindern, erstellt der ObjectPooler nach Erhalt der Daten die benötigten POI-Gameobjects und eine davon abhängige Anzahl an Cluster-Gameobjects. Falls die Anzahl an Gameobjects nicht ausreichend ist, werden bei Bedarf neue Objekte erstellt. Die Gameobjects werden initial deaktiviert. Wenn eines der Objekte für die Erstellung eines Clusters oder eines POIs benötigt wird, wird es aktiviert und die Werte werden gesetzt [21, 22].

5.3.4 Radar

Das Radar ist für den Benutzer in der oberen rechten Ecke sichtbar. Alle POIs, die sich in dem festgelegten Radius um den Benutzer befinden, werden in Form von schwarzen Punkten angezeigt. So wird deutlich, in welcher Richtung weitere POIs angezeigt werden und auch wie viele POIs sich im Sichtfeld des Benutzers befinden. Der Kompass passt sich der Richtung an, in welche der Benutzer schaut.
In Abbildung 5.13 blickt der Benutzer Richtung Norden. Ein POI befindet sich in seinem Sichtfeld und ein weiteres POI wird sichtbar, wenn der Benutzer sich jeweils Richtung Westen und Osten dreht. Um darzustellen, wo die POIs im Umkreis des Benutzers liegen, wird eine zweite Kamera, die Radar-Kamera eingesetzt. Die Abbildungen 5.14 und 5.15 zeigen den Aufbau der AR- und der Radar-Kamera. Die AR-Kamera nimmt das auf, was der Benutzer auf seinem Smartphone zu sehen bekommt, in diesem Fall POIs und Cluster, siehe Abbildung 5.14.
Die Radar-Kamera ist vertikal über der AR-Kamera mit Blick zu dieser positioniert, sodass der Benutzer immer den Mittelpunkt der Kameraansicht bildet. Anders als die AR-Kamera, sind für die Radar-Kamera nur die schwarzen Kugeln der POIs sichtbar, siehe Abbildung 5.15. Diese erscheinen in der Radar-Ansicht als schwarze Punkte. Die Radar-Kamera bewegt sich mit der AR-Kamera mit und übernimmt Rotationen entlang der y-Achse. Bei Neigungen der AR-Kamera entlang der x- oder z-Achse verhält sich

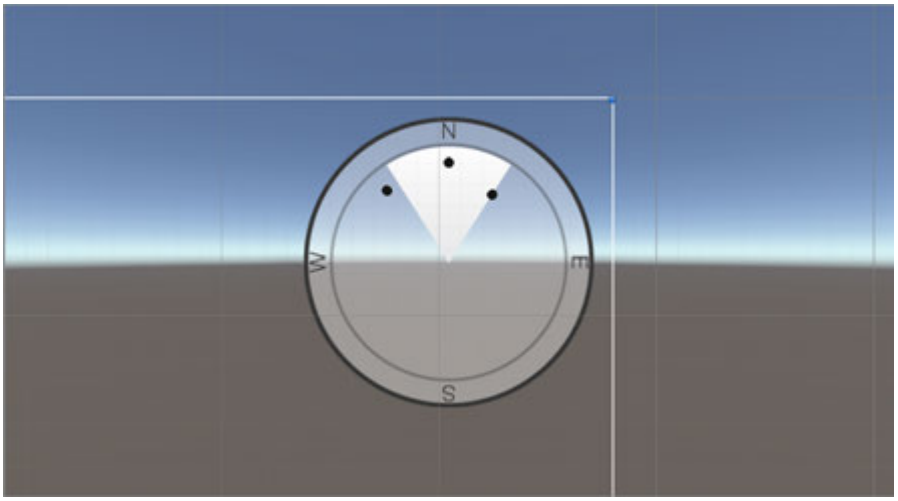


Abbildung 5.13: Radar mit drei sichtbaren POIs

die Radar-Kamera anders und behält ihre Position bei. Das verhindert, dass die Punkte im Radar aus einem anderen Winkel als gewollt angezeigt werden.
Um die Ansicht der Radar-Kamera an den angegebenen Radius anzupassen, wird die y-Position und die Sichtweite der Kamera angepasst. Beide Werte sind abhängig von der Distanz, in welcher POIs angezeigt werden sollen, und einem Offset-Wert. Des Weiteren wird die Größe der POI-Kugeln ebenfalls abhängig von der maximalen Distanz festgelegt, sodass jene in der Radar-Ansicht möglichst die gleiche Größe beibehalten.

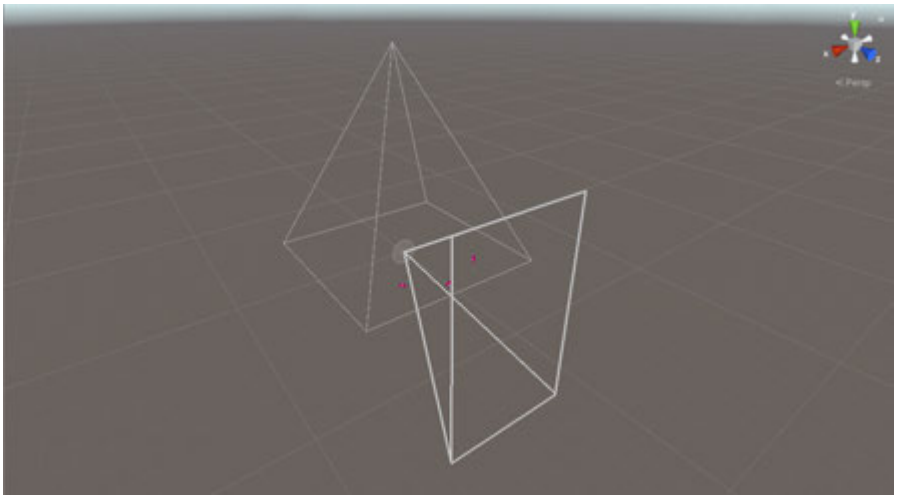


Abbildung 5.14: Sicht der AR-Kamera

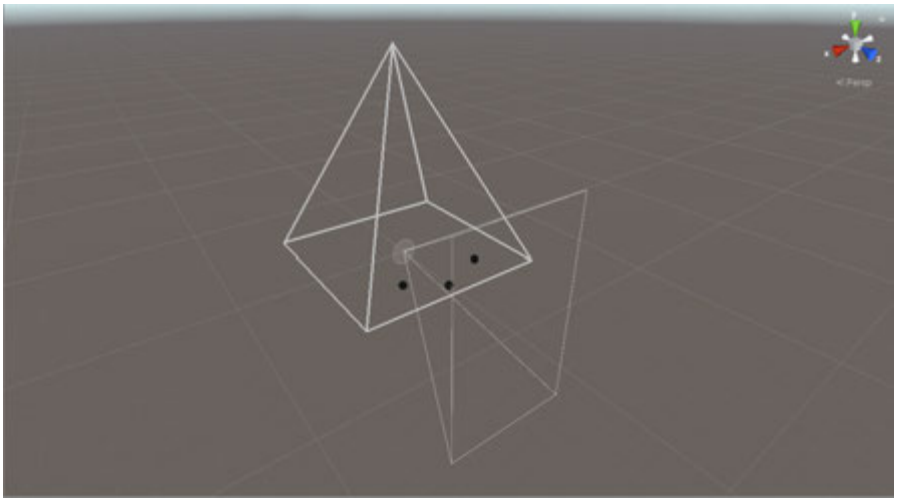


Abbildung 5.15: Sicht der Radar-Kamera

5.3.5 AR-Erweiterung

Die AR-Erweiterung ergänzt ein POI durch Verknüpfungen. Sie wird angezeigt, wenn ein POI, das mehr als eine Verknüpfung besitzt, ausgewählt wird. Dadurch hat der Benutzer die Möglichkeit auszuwählen aus welchem Bereich des POIs er nähere Informationen oder Materialien möchte. In Abbildung 5.16 ist die Ansicht eines POIs mit vier Verknüpfungen zu sehen.

Abbildung 5.16: POI-Erweiterung mit vier Verknüpfungen

Jedes POI speichert eine Liste, die angibt welche Verknüpfungstypen für jenes existieren. Abbildung 5.5 verdeutlicht diesen Aufbau. Die AR-Ansicht besteht aus acht Button-Gameobjects, welche jeweils eine Verknüpfung repräsentieren. Ein Button-Gameobject besitzt zu jeder Verknüpfung eine Komponente, welche deaktiviert ist. Dadurch kann jeder Button für eine beliebige Verknüpfung verwendet werden, indem die entsprechende Komponente aktiviert wird. Jede Verknüpfung implementiert eine individuelle Action-Methode, die in den meisten Fällen, von der nativen App verarbeitet wird, siehe dazu 5.8.

5.3.6 Unity-Applikation als Bibliothek

Am Ende soll das Unity-Modul nicht als eigenständige Anwendung verwendet werden. Stattdessen ist es das Ziel das Unity-Modul in eine native iOS- und Android-App einzubinden. Ab der Unity Version 2019.3 ist es möglich das Projekt als Bibliothek für beide Plattformen zu exportieren [23]. Im Folgenden gibt das Kapitel einen Überblick über die Einbindung in Android und in iOS.
Für den Android-Export generiert Unity ein Gradle-Projekt, das aus einem Modul namens unityLibrary besteht. Alle notwendigen Daten zur Laufzeit und zum Unity-Spieler werden darin zur Verfügung gestellt. Auf diese kann die native App bei einer Integration zugreifen. Um das Modul innerhalb der Anwendung aufzurufen, muss die UnityActivity gestartet werden [24]. Voraussetzung für die Benutzung des Unity-Moduls mit Android ist ARCore, das ab dem API Level 24 (Nougat, Version 7.0) mit den meisten Geräten kompatibel ist [25].
Für die iOS-Anwendung wird das Unity-Projekt zu einem Xcode-Projekt exportiert. Dabei wird ein Modul namens UnityFramework generiert, das die Daten für die native App zur Verfügung stellt. Für die Integration wird das native Xcode-Projekt mit dem von Unity generierten Xcode-Projekt über einen gemeinsamen Workspace vereint. Über das Unity-Framework kann das Unity-Modul zur Laufzeit gesteuert werden [26]. Voraussetzung für die Benutzung des Unity-Moduls mit iOS ist ARKit, das ab der iOS Version 11 unterstützt wird [27].
Für die Verwendung des Unity-Moduls als externe Bibliothek muss unter anderem be-

achtet werden, dass es nur möglich ist das Modul als Full-Screen-Anwendung zu starten. Außerdem kann in einer nativen Anwendung nur eine Instanz des Unity-Moduls geladen werden [24, 26].

6

Zusammenfassung und Ausblick

In diesem Kapitel wird eine Zusammenfassung über den Aufbau der Arbeit und deren Inhalt gegeben. Anschließend wird in einem Ausblick erläutert an welchen Stellen Verbesserungen vorgenommen und welche Funktionalitäten als Erweiterungen in den Liveguide eingebunden werden können.

6.1 Zusammenfassung

In dieser Arbeit wurde der Liveguide der Firma cm city media vorgestellt. Dieser findet Anwendung in der Stadt-App, welche für viele Gemeinden, Städte und Landkreise in den App Stores zur Verfügung steht. Der Liveguide bietet den Benutzern die Möglichkeit Orte virtuell zu erkunden. Dabei werden virtuelle Informationen zu interessanten Orten in die Gerätekamera eingebunden. Der Liveguide wurde von Grund auf selbst implementiert, wodurch das Framework AREA enstanden ist. Die Eigenimplementation des Kerns bringt allerdings einige Problematiken mit sich. Bei der Einbindung von Erweiterungen entsteht beispielsweise ein großer Aufwand, wenn die Funktionalitäten von Grund auf implementiert werden müssen. Der Fokus kann dadurch nicht auf der eigentlichen Funktion liegen. Außerdem muss der Kern regelmäßig an neue Technologien angepasst werden, um Fehlverhalten zu vermeiden. Um diese Probleme zu umgehen, wurde der Liveguide mit Hilfe des Frameworks AR-Foundation von Unity neu aufgesetzt. Das Framework hat den Vorteil, dass Grundfunktionalitäten bereits implementiert sind und durch regelmäßige Updates neue Funktionalitäten hinzukommen oder bereits bestehende Funktionalitäten verbessert werden.
Der Unity-Liveguide orientiert sich an dem AREA-Liveguide und übernimmt dessen

Design und Funktionen. Die folgende Abbildung 6.2 zeigt einen Vergleich beider Module. Auf der linken Seite ist jeweils ein Ausschnitt des AREA-Liveguides zu sehen und auf der rechten Seite ein Ausschnitt des Unity-Liveguides. Wie die Abbildung deutlich macht, bildet der Unity-Liveguide den AREA-Liveguide ab und erweitert diesen in einigen Details. Während der AREA-Liveguide alle POIs und Cluster in gleicher Entfernung anzeigt, unterscheidet der Unity-Liveguide die Entfernung der Orte. Das hat zur Folge, dass Objekte, die in der realen Welt weiter entfernt sind, kleiner dargestellt werden. Hinzu kommt die Implementierung der AR-Erweiterung, welche es ermöglicht Verknüpfungen zu einzelnen POIs hinzuzufügen. Diese Verknüpfungen ermöglichen es dem Benutzer auszuwählen aus welchem Bereich sie nähere Informationen zu dem POI angezeigt haben wollen.
Weiter stellt die Arbeit den Algorithmus vor, der bestimmt, wann POIs in einem Cluster zusammengefasst werden. Dabei kann ein POI entweder einem bereits existierenden Cluster hinzugefügt werden, ein neues Cluster mit einem weiteren POI bilden oder als einzelnes POI dargestellt werden. Das Radar bietet dem Benutzer eine Übersicht aller POIs in der Umgebung und zeigt an, welche POIs sich im Sichtfeld befinden.
Der Unity-Liveguide wird in die Stadt-App exportiert, welche nativ für Android und iOS existiert. Aufgrund der Nutzung von AR-Foundation, werden einige Anforderungen an die Geräte gestellt. Daher wird der Unity-Liveguide bei allen Geräten aktiv, welche die Anforderungen erfüllen. Bei der Benutzung eines Geräts, das die Anforderungen nicht erfüllt, wird der AREA-Liveguide aufgerufen.
Zusammengefasst ersetzt der Unity-Liveguide den AREA-Liveguide in allen Funktionen. Zusätzlich ermöglicht er das einfache und flexible Entwickeln von Erweiterungen.

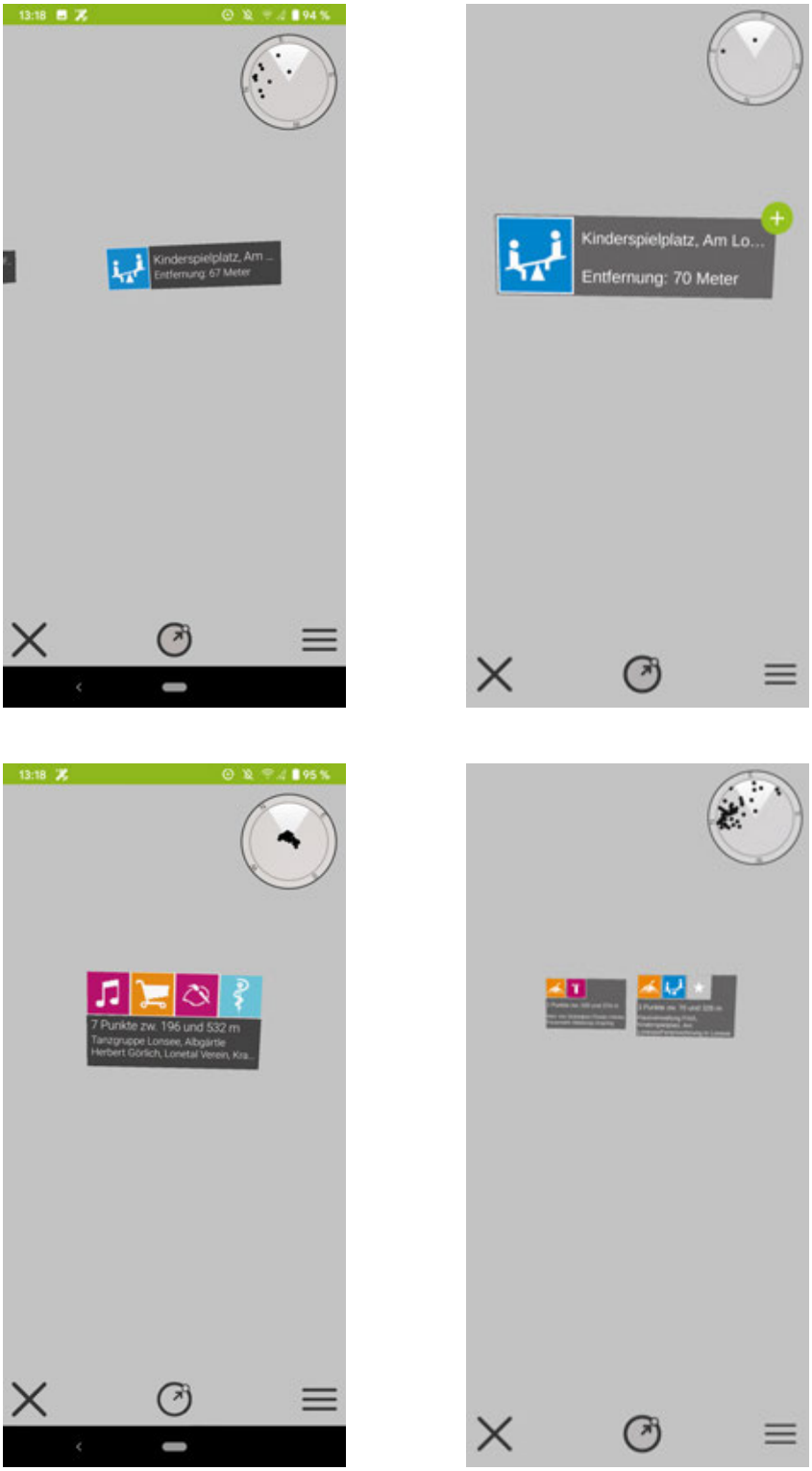


Abbildung 6.2: POI und Cluster im AREA-Liveguide (links) und Unity-Liveguide (rechts)

6.2 Ausblick

Technisch betrachtet, kann der Unity-Liveguide in einzelnen Details verbessert werden. So hat das Berechnen und Erstellen von POIs und Clustern beim Start des Moduls eine kurze Wartezeit für den Benutzer zur Folge. Weiter kann die Stabilität und Präzision bezüglich der Positionierung von AR-Objekten verbessert werden. Dies hängt allerdings von den Gerätesensoren und deren Positionsberechnung ab und kann daher auch zwischen verschiedenen Geräten schwanken.
Das Modul implementiert bisher nur die grundlegenden Funktionen einer standortbezogenen Anwendung. Wirft man einen Blick auf den AREA-Liveguide, gibt es noch zusätzliche Funktionalitäten, die übernommen werden können. Das betrifft das Einbinden von Strecken und Flächen. Strecken markieren Straßen und Wege, welchen der Benutzer folgen kann. So kann zum Beispiel eine Navigation zwischen mehreren POIs aktiviert werden, was einer virtuellen Stadtführung gleich kommen kann. Flächen hingegen markieren größere Gebiete, die eine Bedeutung haben und von der restlichen Umgebung abgehoben werden soll. Darunter zählen zum Beispiel Fußball- oder Sportplätze. Eine weitere Möglichkeit den Liveguide zu erweitern, ist die Interaktion mit 3D-Figuren. Diese können zum Beispiel positionsabhängig an Orte platziert werden. Wenn der Benutzer sich an diesem Ort befindet, kann er mit der Figur interagieren. Eine Interaktion ist auf verschiedenen Wegen möglich. Beispiele dafür sind das Sammeln von Punkten, wenn die Figur durch einen Touch ausgewählt wird oder das Unterhalten mittels eines Chatbots. Solche 3D-Figuren können außerdem als Navigation dienen und den Weg ausweisen, indem sie diesen ablaufen und der Benutzer folgen kann.

Literaturverzeichnis

[1] Schickler, M., Reichert, M., Geiger, P., Weilbach, M., Pryss, R.: The AREA Algorithm Framework Enabling Location-based Mobile Augmented Reality Applications. Procedia Computer Science **155** (2019) 193–200

[2] Statista Research Department: Number of mobile augmented reality (AR) users worldwide from 2015 to 2023. (https://www.statista.com/statistics/1098630/global-mobile-augmented-reality-ar-users/) [aufgerufen am 07.April 2020].

[3] Yovcheva, Z., Buhalis, D., Gatzidis, C.: Smartphone augmented reality applications for tourism. E-review of tourism research (ertr) **10** (2012) 63–66

[4] cm city media GmbH: Stadt sind wir. (https://stadtsindwir.de/) [aufgerufen am 01.April 2020].

[5] Stumpp, S., Knopf, T., Michelis, D.: User experience design with augmented reality (ar). In: European Conference on Innovation and Entrepreneurship, Academic Conferences International Limited (2019) 1032–XXVI

[6] Rauschnabel, P.A., Rossmann, A., tom Dieck, M.C.: An adoption framework for mobile augmented reality games: The case of pokémon go. Computers in Human Behavior **76** (2017) 276–286

[7] Zobel, B., Werning, S., Metzger, D., Thomas, O.: Augmented und virtual reality: Stand der technik, nutzenpotenziale und einsatzgebiete. In: Handbuch Mobile Learning. Springer (2018) 123–140

[8] Inter IKEA Systems B.V.: IKEA Place. (https://play.google.com/store/apps/details?id=com.inter_ikea.place&hl=de) [aufgerufen am 30.März 2020].

[9] Pryss, R., Schickler, M., Schobel, J., Weilbach, M., Geiger, P., Reichert, M.: Enabling Tracks in Location-Based Smart Mobile Augmented Reality Applications. Procedia Computer Science **110** (2017) 207–214

[10] 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** (2017) 13–21

[11] cm city media GmbH: cm city media. (`https://www.cmcitymedia.de/de/startseite`) [aufgerufen am 01.April 2020].

[12] cm city media GmbH: Bühlerzell. (`https://play.google.com/store/apps/details?id=de.cmcitymedia.buehlerzell&hl=de`) [aufgerufen am 01.April 2020].

[13] cm city media GmbH: Lonsee. (`https://play.google.com/store/apps/details?id=de.cmcitymedia.lonsee&hl=de`) [aufgerufen am 01.April 2020].

[14] Unity Technologies: Unity for all. (`https://unity.com/`) [aufgerufen am 26.März 2020].

[15] Šmíd, A.: Comparison of unity and unreal engine. Czech Technical University in Prague (2017)

[16] Google Developers: ARCore overview. (`https://developers.google.com/ar/discover`) [aufgerufen am 26.März 2020].

[17] Apple Inc.: ARKit. (`https://developer.apple.com/documentation/arkit`) [aufgerufen am 26.März 2020].

[18] Unity Technologies: AR Foundation. (`https://unity.com/de/unity/features/arfoundation`) [aufgerufen am 26.März 2020].

[19] Unity Technologies: About AR Foundation. (`https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@3.1/manual/index.html`) [aufgerufen am 26.März 2020].

[20] Daniel Fortes: Unity AR+GPS Location. (`https://docs.unity-ar-gps-location.com/`) [aufgerufen am 26.März 2020].

[21] Unity Technologies: Object Pooling. (`https://learn.unity.com/tutorial/object-pooling`) [aufgerufen am 16.März 2020].

[22] Microsoft: Performance recommodations for Unity. (https://docs.microsoft.com/de-de/windows/mixed-reality/performance-recommendations-for-unity) [aufgerufen am 16.März 2020].

[23] Unity Technologies: Using Unity as a Library in other applications. (https://docs.unity3d.com/2019.3/Documentation/Manual/UnityasaLibrary.html) [aufgerufen am 23.März 2020].

[24] Unity Technologies: Integrating Unity into Android applications. (https://docs.unity3d.com/2019.3/Documentation/Manual/UnityasaLibrary-Android.html) [aufgerufen am 23.März 2020].

[25] Google Developers: ARCore supported devices. (https://developers.google.com/ar/discover/supported-devices) [aufgerufen am 26.März 2020].

[26] Unity Technologies: Integrating Unity into native iOS applications. (https://docs.unity3d.com/2019.3/Documentation/Manual/UnityasaLibrary-iOS.html) [aufgerufen am 26.März 2020].

[27] Apple Inc.: ARKit. (https://developer.apple.com/documentation/arkit/) [aufgerufen am 26.März 2020].

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Leoni Holl Matrikelnummer: 866729

Erklärung

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

Ulm, den .....

Leoni Holl