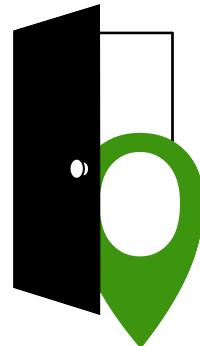


Technische Hochschule Georg-Simon Ohm Nürnberg Fakultät  
Elektrotechnik Feinwerktechnik Informationstechnik efi

**Projekt „Indoorino - Die Indoornavigation“**



**indoorino**

**Prüfungsstudienarbeit** von

Chantal Reng / Matr.-Nr.: 3037131

Kristina Pankova / Matr.-Nr.: 3155361

Inga Glotzbach / Matr.-Nr.: 2985250

Jeremy Turner / Matr.-Nr.: 2363363

Philipp Wolf / Matr.-Nr.: 3081830

Vorgelegt am 26.02.2019

B-ME 5

Wintersemester 2018/2019

## **i Ehrenwörtliche Erklärung**

Chantal Reng,  
Glotzbach Inga,  
Philipp Wolf,  
Jeremy Turner,  
Kristina Pankova

bestätigen, dass wir die Prüfungsstudienarbeit mit dem Titel:

### **„Indoorino - Die Indoornavigation“**

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet haben.

Wir bestätigen, dass wir der Technischen Hochschule Nürnberg für die in der Prüfungsstudienarbeit angefertigten Unterlagen, Dokumente, Fotografien und Filme sowie für die dabei entwickelte Soft- oder Hardware das uneingeschränkte nichtausschließliche Nutzungsrecht einräumen:

Datum: 26.02.2019

Reng, Chantal	
Glotzbach, Inga	
Wolf, Philipp	
Turner, Jeremy	
Pankova, Kristina	

# **Inhaltsverzeichnis**

i	<b>Ehrenwörtliche Erklärung</b>	
ii	<b>Inhaltsverzeichnis</b>	
<b>1</b>	<b>Abstract</b>	<b>5</b>
<b>2</b>	<b>Konzeption</b>	<b>6</b>
	2.1 Brainstorming	6
	2.2 Konzeption und erste Ausarbeitung	7
	2.3 Funktionalitäten der Anwendung	9
	2.4 Projektname	13
	2.5 Organisation	14
	2.5.1 Planungstool	14
	2.5.2 Versionskontrolle	16
	2.5.3 Gruppeninterne Organisation	16
	2.6 Recherche	17
	2.5.1 Android RTT	17
	2.5.1 GPS - Global Positioning System	27
<b>3</b>	<b>Umsetzung</b>	<b>28</b>
	3.1 Unity	28
	3.1.1 Grundfunktionen	28
	3.1.2 Objekt auf einer Route bewegen lassen	30
	3.1.3 Objektradius auf andere Objekte prüfen	32
	3.1.4 Unity GPS Abfrage	35
	3.2 Android Studio	37
	3.2.1 Android Library Plug-In für Unity	38
	3.2.2 Android GPS Abfrage	40
	3.2 Ortsabfrage und Berechnungen	43
	3.2.1 Geoid/Ellipsoid Berechnung	43

3.2.2 Koordinatensystem Global to Local	46
3.2.3 Umwandlung GPS zu ECEF Koordinaten	47
3.3 LibGDX	50
3.3.1 Einbindung LocationManager	52
3.3.2 Importieren von 3D Modellen	55
3.3.3 Drehung des Testgeländes Nordausrichtung	56
3.3.4 Positionierung des Spielerobjekts	57
3.3.5 Prototyp	61
3.4 Design	65
3.4.1 Erstellung 3D Modelle in 3D Max	65
3.4.2 Charakterdesign	71
3.4.3 Erstellung des App Menüs in Unity	75
<b>4 Marketing und Präsentation</b>	<b>88</b>
4.1 Logo	88
4.2 Plakatdesign	90
4.3 Video	91
<b>5 Zusammenfassung und Ausblick</b>	<b>94</b>
5.1 Resümee der Projektarbeit	94
5.2 Ausblick: Zoorino	95
5.2.1 Idee	95
5.2.3 Konzeptausblick	96
<b>6 Literatur- und Quellenverzeichnis</b>	<b>114</b>
<b>7 Abbildungsverzeichnis</b>	<b>118</b>
<b>8 Einzelnes Mitwirken am Projekt</b>	<b>121</b>

# **1 Abstract**

Das Projekt mit dem Namen Museums-Navigation mit Android RTT, unter der Betreuung von Prof. Dr. Stefan Röttger, wurde im Rahmen der Projektarbeiten des Studiengangs Media-Engineering durchgeführt. Diese werden vom Studienplan im fünften und sechsten Semester vorgesehen.

Die Bearbeitung des Gesamtprojekts erfolgte in einem Team, bestehend aus fünf Studierenden. Im Folgenden wird über die Tätigkeiten und Aufgaben berichtet, welche im Laufe des Semesters angefallen sind.

Die Vorgabe des Projektthemas lag darin, eine zuverlässige Indoor-Navigation durch Wifi-RTT, also durch die Messung des Abstandes zu einem WLAN Access-Point, zu realisieren. Diese Ortungsmethode ist in der neuen Version des Betriebssystems Android 9 Pie implementiert.

Als mögliche Anwendungsorte, mit deren Zusammenarbeit die Navigation umgesetzt werden könnte, wurden in der Projektbeschreibung das Deutsche Museum in Nürnberg sowie das Industriemuseum in Lauf vorgeschlagen.

Mithilfe einer positionserkennungsfähigen App soll die Navigation umgesetzt werden, um im Museum an bestimmten Stellen Informationen zu den Ausstellungsstücken zu liefern.

Durch die Bearbeitung des eben aufgeführten Themas wird sich mit Hilfe der Navigations-App eine bessere Zurechtfindung in dem Museum erhofft.

Das Team sieht dieser Herausforderung positiv entgegen, sich in die anfallenden Aufgabenbereiche sowie Programme einzuarbeiten und die Navigation mithilfe der neuen Technologie RTT umzusetzen.

## **2 Konzeption**

### **2.1 Brainstorming**

Die vorgegebenen Eckdaten für die Umsetzung des Indoor-Navigations-Projekts lauteten, dass dieses mithilfe von Android RTT zu realisieren ist, um an bestimmten Standorten eines Museums gezielt Informationen auf einer App angezeigt zu bekommen.

Zu Beginn des Semesters wurde sich überlegt, welche Anwendungsbereiche für dieses Projekt in Frage kommen. Eine mögliche Umsetzung dieser Vorgaben wäre auf eine virtuelle Schnitzeljagd zutreffend. Weiterhin kam kurz der Gedanke auf, die Indoor-Navigation für eine bessere Zurechtfindung mit einem Leitsystem an Airports zu entwickeln. Wenn nun die Zusammenarbeit mit dem Laufer Museum betrachtet wird, haben weitere Vorschläge der Ideenfindung ergeben, ein Navigations-Spiel in die bereits bestehenden Spielecken des Museums zu integrieren, in denen beispielsweise die verschiedenen großen Ventile aus Edelstahl den jeweiligen Motortypen zugeordnet werden. Letztendlich wurde sich dazu entschlossen, den Grundgedanken relativ simpel zu halten und eine Navigation zu entwickeln, bei welcher die Positionskoordinaten ausgewertet werden, um eine Navigation und Informationen zu den verschiedensten Museumsexponaten zu erhalten. Als Erweiterung des Konzepts ist die Informationsausgabe in Form eines Hörspiels und die Visualisierung durch Augmented Reality denkbar.

Aufgrund der Tatsache, dass bereits in den vergangenen Jahren ein Media-Engineering-Projektteam die gute Zusammenarbeit mit dem Industriemuseum in Lauf genießen durfte und sich ebenfalls die Thematisierung des Museums als positiv und ansprechend erwies, kristallisierte sich relativ schnell heraus, dass dieses als Location für die Navigation dienen würde.

Somit kam es zu dem Beschluss, den Schauplatz der Navigation genauer zu betrachten und das Team fuhr zu dem Industriemuseum in Lauf. Das dortige Gelände besteht aus mehreren Ausstellungshallen sowie aus einem großzügigen Außenbereich. Dort angekommen fand ein Austausch mit den Museumsinhabern über die technischen Geräte statt. Beispielsweise wurde geklärt, ob bereits Router auf dem Museumsgelände vorhanden sind und an welchen Stellen sich diese befinden, um die weitere Projektplanung im Hinblick auf die Umsetzung mit RTT besser einschränken zu können. Des Weiteren wurde das Museum von dem Team genau inspiziert und es wurde sich in die Lage eines Museumsbesuchers versetzt, immer im Hinterkopf, welche Bereiche und Details sich gut in die Indoor-Navigation einbauen lassen.

## **2.2 Konzeption und erste Ausarbeitung**

Nach diesem Ereignis entwarf Chantal Reng, eines der Teammitglieder, mit dem Hintergrundwissen der Museums-Exkursion, dessen Konstruktionsplan sowie mit dem im Museum ausgestelltem Inhalt, folgende Konzeptidee:

Die fiktive städtische Arbeiterfamilie aus dem 20. Jahrhundert führt die Besucher durch die verschiedenen Ausstellungsbereiche Industrie, Handwerk und Gewerbe. Ebenso kann die auf dem Museumsgelände ausgestellte Wohnung der Hauptfiguren im Jahre 1950 besichtigt werden.

Die Führung an sich erfolgt durch ein Hörspiel, das von akustischen Hintergrundgeräuschen begleitet wird. Diese lassen sich im Museum sehr gut einbauen, um den Charme der Industrie aus längst vergangenen Tagen einzufangen und zum Beispiel in der Nähe des Wasserrads das plätschernde Wasser und in den Hallen den Maschinenlärm sowie die Arbeiter akustisch festzuhalten.

Als Protagonisten des Hörspiels bieten sich die für die Zwecke der Navigation frei erfundenen Kinder der Laufer Familie an, mit welchen sich die jungen Museumsbesucher identifizieren können. Diese wohnen zusammen mit deren Eltern auf dem Fabrikgelände und wissen bestens über das Leben dort und die verschiedenen Abläufe Bescheid.

Das eben erwähnte Wissen geben die fiktiven Kinder anhand einer Geschichte an die Museumsbesucher weiter, welche sich an ausgewählten Stationen im Museum aufgrund der Positionierung durch RTT selbstständig abspielt. Dies soll durch eine App für Android Smartphones realisiert werden, auf die RTT-fähigen Geräte ab Android 9 ausgerichtet. Die Informationen erhält der Museumsbesucher über Kopfhörer, welche an das Smartphone angeschlossen werden.

In diesem Zusammenhang kam die Idee auf, Augmented Reality mit in das Projekt einzubauen, um an bestimmten Ortspunkten gezielt visuelle Informationen auf dem Handydisplay zu erhalten. Eine sehr bemerkenswerte, aber durchaus schwer realisierbare Vorstellung wäre es, wenn einige Exponate „zum Leben erwachen“, sobald der Besucher das Smartphone mit seiner Indoorino-App über das beispielsweise mittlerweile stillgelegte Wasserrad hält oder die Mehlmühle wie in früheren Jahren plötzlich zum Mahlen beginnt. Hierbei liegt leider die Schwierigkeit darin, nicht einzig die Gegenstände an sich, wie z.B. das Wasserrad oder die Mühle, sondern eine komplette Umgebung in 3D zu modellieren, um ebenso das Gewässer, das Mehl, die Halle, in der sich die Mühle befindet, den Hintergrund etc. darstellen zu können. Dies ist machbar, aber in der gegebenen Zeit für die Projektarbeit, welche zwei Semester beträgt, unmöglich umzusetzen.

Als Erweiterung werden die Besucher von einem virtuellen AR-Charakter durch die Ausstellung geleitet, welcher den Besuchern den Aspekt der Navigation erleichtert und über die App auf dem Smartphone verfolgt werden kann.

Da sich während der Museumsexkursion des Teams tatsächlich ein dort angetroffener Besucher in den verwinkelten Hallen verlaufen hat und den Ausgang nicht mehr entdeckte, findet eine für das Museum generierte Navigation, welche sowohl informativ, als auch wegleitend ist, im Museum Anklang.

Das Laufer Museum zählt eher rare Schilder vor den Ausstellungsstücken, somit wäre der Besucher sicherlich für einen weiteren Informationsinput offen. Ein Ansatz für technische Geräte ist im Museum bereits vorzufinden, da QR-Codes an einigen Stellen angebracht sind, welche theoretisch mit in das Projekt eingebunden werden können.

## **2.3 Funktionalitäten der Anwendung**

Die App repräsentiert die Schnittstelle zwischen der Navigation als Anwendung und dem Museumsbesucher. Der Besucher lädt sich zu Beginn die Indoorino-App aus dem App-Store auf sein eigenes Endgerät herunter. Von Vorteil wäre es, wenn der Benutzer Kopfhörer mit sich führt, um die Audiogeräusche während der Tour zu empfangen. Durch den Benutzer wird die Indoorino App gestartet. Zuerst tritt das Main Menu in Erscheinung. Die Auswahlmöglichkeiten der Menüpunkte bestehen aus „Navigation Starten“, „Audio“ und „Beschreibung“.

Die wichtigste Funktion der App stellt der „Navigation Starten“-Button dar. Beim Betätigen dieses Feldes wird der Nutzer zu einer weiteren Ebene geleitet, bei welcher die eingebettete Map, der Standort des Users sowie der Avatar auf einem neuen Screen erscheinen. An dem Punkt wird der vom Team geschriebene Code für die Navigation

angesprochen. In dieser Ansicht kann sich der Museumsbesucher frei auf dem Gelände bewegen und das Museum auf eigene Weise erkunden.

An großen Attraktionen, beispielsweise am Wasserrad, meldet sich nun der Avatar zu Wort und informiert den Besucher auditiv per Kopfhörer über einige Details. Zeitgleich wird eine Sprechblase unmittelbar neben dem Avatar erscheinen, indem die jeweiligen Informationen zu den Exponaten schriftlich vorzufinden sind. Die Existenz dieser Funktion ist von Bedeutung, wenn sich der User für eine tonlose Navigation entscheidet, falls dieser beispielsweise keine Kopfhörer mitführt. Folglich hat der Benutzer der App dennoch die Möglichkeit, mit der Indoorino-App das Museum zu erkunden, indem dieser die Informationen schriftlich in den Sprechblasen erhält.

In der Landkarte ist die Ansicht ähnlich des Pokemon-Go Spieles in einer 2.5D-Ansicht gestaltet. Der Verlauf der Navigation wird „durch die Veränderung der geographischen Position des Spielers beeinflusst“ [1]. Der Avatar ändert seine Position auf der Map synchron zu den Bewegungen des Benutzers und folgt dessen Routen. Hierbei wird „eine [...]umgebung auf dem Prinzip der erweiterten Realität [2] genutzt. Augmented Reality kommt zum Einsatz. Die Rede ist von dem virtuellen Charakter, welcher in die bestehende, den realen Maßstäben getreue Map eingebettet wird. Bei der ersten Konzeptionierung war der Plan, die Landkarte auf Basis der „Live Location Platform“ [3] MapBox zu realisieren. Visualisiert werden die Map und der Avatar, indem das Kamerabild überlagert wird.

Die Standortdaten werden über RTT ermittelt, um den Avatar in der Navigation zu positionieren, zu bewegen sowie um die Informationen an den verschiedenen Ausstellungspunkten zu koordinieren.

Auf der Map befinden sich einige grafische Elemente, z.B. in Form des Wasserrads oder der Mühle, auf welche der Benutzer klicken kann,

um direkt zu den jeweiligen Attraktionen geleitet zu werden. Wenn sich für diese Option entschieden wird, erscheint die gewünschte Tour rot hinterlegt auf der Map. Der Avatar setzt sich in Bewegung, damit der User dieser Route zu seinem Zielobjekt folgen kann.

Wenn der Appnutzer nun an den AR-Points, den besagten Attraktionen wie z.B. am Wasserrad angelangt ist, wird er durch Vibration des Geräts darauf hingewiesen. Es kommt zu einer Veränderung der Kameraperspektive und das Exponat sowie dessen Umgebung kann durch die Smartphonekamera betrachtet werden. In der Sprechblase sowie über die Kopfhörer erhält der User den an die Exponate angepassten Informationsinput. Dieser besteht beispielsweise aus dem geschichtlichen Hintergrund, dem Baujahr, der Funktionsweise und den Einsatzbereichen der Ausstellungsstücke.

Wenn sich der User in der Navigationsansicht befindet, kann dieser durch einen entsprechend gekennzeichneten Button am Rand des Screens wieder in das Main Menü navigiert werden.

Der nächste Menüpunkt repräsentiert die Audiofunktion. Hier sind zwei Regler für die Lautstärke vorhanden. Der eine Regler bezieht sich auf die Geräusche des Avatars, wenn dieser die User an den AR-Points über Ausstellungsinformationen in Kenntnis setzt. Durch Verschieben des zweiten Reglers wird die Lautstärke der Hintergrundgeräusche reguliert. Dazu zählen das Ertönen des Geplätschers in unmittelbarer Nähe des Wasserrads oder der allgegenwärtige Maschinenlärm auf dem Gelände.

Der dritte und letzte Menüpunkt ist die Beschreibung. In diesem spiegelt sich eine Art Anleitung wieder, wie die App zu bedienen ist. Der User wird über den begleitenden Avatar Rino informiert, welcher an gezielten Positionen im Museum aktiv wird und Wissenswertes über die Exponate an den User weitergibt.

Als Erweiterung der Navigation sollte die Möglichkeit bestehen, dem Avatar verschiedenste Attribute zuweisen zu können.

Der Nutzer kann zwischen einigen Eigenschaften wie Geschlecht, Bekleidung und Aussehen wählen, um das Erscheinungsbild des Avatars frei auszuwählen.

Bei der Umsetzung des Menüs ist die Hörspiel-Funktion erstmalig in den Hintergrund gerückt worden. Die Führung durch das Museum anhand der Geschichte wird als Erweiterung des Projektes gesehen, sobald die Grundfunktionen im Team umgesetzt wurden. Dabei stellt die Einbettung des Hörspiels in die App keine wesentliche Schwierigkeit dar. Hierfür wird die Oberfläche der App um einen weiteren Menüpunkt erweitert, da das Audiopanel ohnehin schon existiert, um die Lautstärke des Avatars sowie die Soundeffekte, also die Hintergrundgeräusche anzupassen. Zum Abspielen der Museumsgeschichte wird das Audiopanel nochmals um eine neue Tonspur erweitert.

## 2.4 Projektname

### Schlagwörter für die Namensfindung:

- *Interaktiv*
- *Tour*
- *Guide*
- *Scout*
- *Museum*
- *Tür*
- *Tor*
- *Innenraum*
- *Digital*
- *Navigator*
- *Helper*
- *Begleiter*
- *System*
- *Engine*
- *App*
- *Weg*
- *Interesse*
- *Track*
- *Orientierung*
- *Kompass*
- *Forschen*
- *Entdecken*
- *Informationen erhalten*
- *Sinne*
- *Markierung*
- *Industrie*
- *Technik/Mechanik*

Nachdem jeder von uns einige Namensvorschläge erarbeitet hat, haben wir unsere Ergebnisse zusammengeführt:

- *follow me*
- *Tour*
- *Activitour*
- *indoorino*
- *Dalang*

- *Kurator/Kustos/Kuratour/ Tourator*
- *sidekick*
- *Knowhere/Knowwhere/ Knoware*
- *...more?/...mehr?*
- *URNear ( you are near)*
- *Guide & Seek/ Guide'N'Seek ( „hide and seek“= verstecken/ Guide, Navigate, Seek – Führen, navigieren, entdecken)*
- *ming/MING (antike Vase, Museum Interaktiver Navigations Guide)*

Nach gruppeninterner Abstimmung haben wir uns für den Namen „indoorino“ entschieden. Wir waren der gemeinsamen Ansicht, dass der User von vornherein wissen soll, dass es sich um ein Applikation für Innenräume handelt. Den Hinweis auf die Navigation haben wir spielerisch sowohl im Schriftzug, als auch im eigentlichen Logo integriert.

## 2.5 Organisation

### 2.5.1 Planungstool

Zu Beginn hatten wir viele verschiedene Möglichkeiten ein geeignetes Planungstool zur Gruppenorganisation und Projektplanung auszuwählen. Hierbei war uns zunächst wichtig, dass es möglich war verschiedene Aufgaben in bestimmte Bereiche zu gliedern und diese den jeweiligen Mitgliedern zuteilen zu können. Aus diesem Grund haben wir uns vorerst für das Planungstool Asana entschieden [4].

Nach einiger Zeit jedoch sind viele weitere Aspekte dazugekommen, welche mit diesem Tool nicht umsetzbar waren. Zwar konnte man ein Fälligkeitsdatum für eine bestimmte Aufgabe stellen, allerdings war diese nicht in einer Zeitleiste sichtbar, weshalb es die Projektorganisation nicht sonderlich erleichterte. Außerdem ist der

Strukturaufbau recht unübersichtlich, da nicht auf einem Blick feststellbar war, welche Aufgaben in Bearbeitung oder erledigt waren, da es nur vereinzelte Seiten zu den jeweiligen Kategorien gab.

Aus diesem Grund haben wir uns dazu entschieden ein neues Planungstool zu suchen und haben viele verschiedene Programme ausprobiert. Hierbei haben wir drei Favoriten gefunden und hatten die Auswahl zwischen GitHub, Monday und Trello.

Da Monday [5] kostenpflichtig und GitHub [6] etwas unübersichtlich war haben wir uns schlussendlich für das Tool Trello entschieden, da es alle Funktionen beinhaltet welches wir für unsere Organisation brauchen. Trello hat die gleichen Eigenschaften wie das Tool Asana, allerdings ist es möglich alle Aufgaben auf einen Blick zu betrachten, welche auch in Form einer Zeitleiste sichtbar gemacht werden können. Zudem können die verschiedenen Tasks mit Farbe der jeweiligen Wichtigkeit bzw. Dringlichkeit versehen werden. Wenn beispielsweise eine wichtige und dringliche Aufgabe bearbeitet werden muss, kann diese mit Orange versehen werden.

Dies sind Eigenschaften, welche uns die Organisation innerhalb des Teams erleichtert hat, weshalb wir es auch nun im zweiten Projektabschnitt weiterverwenden möchten.

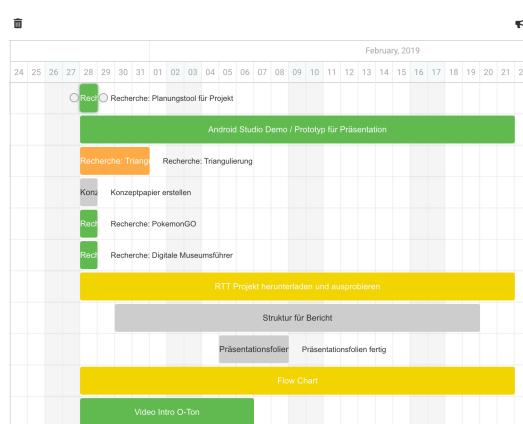


Abbildung 1: Zeitleiste Trello

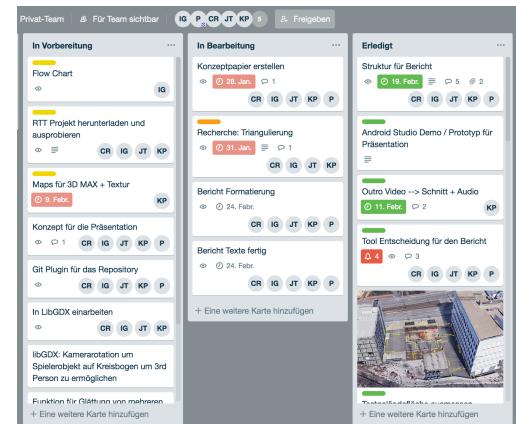


Abbildung 2: Board Trello

### **2.5.2 Versionskontrolle**

Da wir als Team gemeinsam an einem Projekt arbeiten, welches der Softwareentwicklung bedarf, war es wichtig ein gemeinsames Repository zu kreieren um zusammen an unserem Source Code zu schreiben.

Zu Beginn haben wir den Cloud Service der Dropbox verwendet um unseren Stand des jeweiligen Codes abzufragen. Allerdings hat sich dies als sehr umständlich ergeben, weshalb wir uns dazu entschlossen haben den Dienst von Github bzw. Git zu verwenden. Git verwendet dabei die Methode der Versionskontrolle.

Ein Versionskontrollsystem ist ein System welches die Handhabung der Quellcodeveränderung innerhalb des Team erleichtert. Hierbei gibt es ein lokales (Git) und serverseitiges Repository (GitHub) um seine Codeversionen zu verwalten. Falls beispielsweise eine Änderung am Quellcode gemacht wird, wird dieses zunächst in das lokale Repository mittels dem Kommando „Commit“ hochgeladen. Um den veränderten Code nun auf das cloudbasierte Repository zu laden, damit auch andere Teilnehmer auf den neusten Stand kommen, wird hierbei der Befehl „Push“ ausgeführt. Der große Vorteil hierbei ist, dass nach jeder Änderung die in Github hochgeladen wird, Versionen erstellt werden, welche es ermöglichen, falls jemanden ein Fehler unterlaufen ist, den Stand der vorherigen Version abzufragen und nach zu verfolgen um diesen zu korrigieren[8]. Android Studio unterstützt zudem, mittels eines Plugins im Programm selbst, das System von Git, weshalb die Programmierung auch einfacher und schneller zu handhaben war.

### **2.5.3 Gruppeninterne Organisation**

Zur Standbesprechung und für neue Projektschritte wurde ein wöchentliches Treffen angesetzt.

An diesen Treffen wurde besprochen wie man vorwärts kommt, wo die Schwierigkeiten liegen, was die neuen Arbeitsaufgaben sind und über Entscheidungen abgestimmt. Außerdem wurden alle dringenden Anliegen die unter der Woche anfallen in einer WhatsApp Gruppe behandelt oder durch Doodle Umfragen gelöst.

An den einzelnen Aufgaben wurde in kleineren Gruppen oder alleine Zuhause weiter gearbeitet.

Gegen Ende der Projektarbeit haben sich Teams neu gemischt da es Überschneidungen bei der Entwicklung gab und man für den Prototypen und die Präsentation gemeinsame Aufgaben hatte.

## 2.6 Recherche

### 2.6.1 Android RTT

Die einfache Orientierung auf den Straßen ist dank GPS heute nicht mehr wegzudenken. In Innenräumen dagegen, fehlt der Kontakt zu den Satelliten und eine Ortung im Innenbereich ist somit nicht möglich. Was viele Innenräume heutzutage gemeinsam haben, sind WLAN Router. Mit WLAN RTT kann das Endgerät eine ein bis zwei Meter genaue Positionsbestimmung errechnen, sofern mindestens drei WLAN Router gefunden werden. Das Beste dabei ist, das Endgerät muss nicht ein zusätzliches Gerät sein, sondern ist ab Android P mit dem eigenen Handy möglich.

#### WLAN (WiFi) Wireless Local Area Network

(Wireless LAN bzw. W-LAN, meist WLAN; deutsch drahtloses lokales Netzwerk) bezeichnet ein lokales Funknetz, wobei meist ein Standard der IEEE-802.11-Familie gemeint ist. Für diese engere Bedeutung ist in manchen Ländern (z. B. USA, Großbritannien, Kanada, Niederlande, Spanien, Frankreich, Italien) weitläufig beziehungsweise auch synonym der Begriff Wi-Fi gebräuchlich. Verwendet wird der Begriff häufig auch irreführend als Synonym für WLAN-Hotspots [9].

### API Level Application Programming Interface Level

Der API-Level gibt den Entwicklungsstand der eingebauten Funktionen (API, Application Programming Interface) an, die durch einen Entwickler, bspw. bei der Entwicklung Apps, verwendet werden können. Die Angabe ist besonders dann entscheidend, wenn es um die Kompatibilität einer App mit einer auf einem Gerät installierten Android-Version geht. [10]

### Beacon

Mit Beacon wird ein Sender oder Empfänger bezeichnet, der auf der Bluetooth Low Energy (BLE) oder auch Bluetooth Smart Technologie basiert. Im Grunde genommen ist dies eine Funktechnologie, die als Weiterentwicklung von Bluetooth verstanden werden kann.[11]

### VLC Visible Light Communications

VLC ist eine Datenübertragungstechnologie. Dabei werden Daten oder Informationen mit Hilfe des Übertragungsmediums Licht übertragen. Die Frequenz des zur Übertragung genutzten Lichtes befindet sich dabei im sichtbaren Bereich zwischen 400 THz (750 nm; 1 THz = 1000 GHz) und 800 THz (375 nm).[12]

### TCP Transmission Control Protocol

Internet Protocol (TCP/IP) ist eine Gruppe von Netzwerkprotokollen. Im Kern handelt es sich um das Internet Protocol (IP), das Transmission Control Protocol (TCP), das User Datagram Protocol (UDP) und das Internet Control Message Protocol (ICMP). Im weiteren Sinne wird auch die gesamte Internet-Protokollfamilie als TCP/IP bezeichnet [13].

## **RTT**

WiFi RTT = WiFi Round- Trip- Time (Rundumlaufzeit, Paketumlaufzeit) ist ein Teil von der IEEE 802. 11mc WLAN Spezifikation, die erlaubt den Standpunkt in Innenräumen mit einer Genauigkeit von 1-2 Meter zu bestimmen.

## **Das IEEE 802.11mc Protokoll**

Das IEEE 802.11mc Protokoll ist eine Weiterentwicklung des WLAN Standards.

### Funktionsweise RTT

Der Innenraum- Standort wird berechnet, indem mindestens drei WLAN Router vorhanden und deren Standorte bekannt sind. Mit Hilfe von Lateration (Trilateration oder Multilateration) lässt sich dieser Standpunkt bestimmen.

## **Android P**

Die Standortbestimmung benötigt ein Betriebssystem, welches das WLAN RTT Protokoll unterstützt. Aktuell unterstützt Android das Verfahren ab dem API Level 28 (Android P) Es stellt die entsprechenden Klassen zur Verfügung um Applikationen die Nutzung des WLAN RTT Protokoll zu ermöglichen.

Der größte Vorteil dabei ist, dass der Nutzer mit seinem persönlichen Smartphone diese Möglichkeit erhält. Damit ist die Innenraum- Positionsbestimmung für Jedermann zugänglich (bzw. alle Android P Nutzer).

## android.net.wifi.rtt

added in API level 28

Provides classes which allow applications to use Wi-Fi RTT (IEEE 802.11mc) to measure distance to supporting Access Points and peer devices.

The primary entry point to Wi-Fi RTT capabilities is the `WifiRttManager` class, which is acquired by calling `Context.getSystemService(Context.WIFI_RTT_RANGING_SERVICE)`

Some APIs may require the following user permissions:

- `ACCESS_WIFI_STATE`
- `CHANGE_WIFI_STATE`
- `ACCESS_FINE_LOCATION`

Usage of the API is also gated by the device's Location Mode: whether it permits Wi-Fi based location to be queried.

★ Note: Not all Android-powered devices support Wi-Fi RTT functionality. If your application only works with Wi-Fi RTT (i.e. it should only be installed on devices which support Wi-Fi RTT), declare so with a `<uses-feature>` element in the manifest file:

```
<manifest ...>
    <uses-feature android:name="android.hardware.wifi.rtt" />
    ...
</manifest>
```

Alternatively, if your application does not require Wi-Fi RTT but can take advantage of it if available, you can perform the check at run-time in your code using `hasSystemFeature(String)` with `FEATURE_WIFI_RTT`:

```
getPackageManager().hasSystemFeature(PackageManager.FEATURE_WIFI_RTT)
```

Abbildung 3: Android RTT Referenz [14]

Ein weiterer Vorteil ist die Möglichkeit den Standpunkt zu berechnen, ohne sich tatsächlich mit dem Router verbinden zu müssen. Um den Standpunkt zu erhalten reicht lediglich ein Ping aus, welches das Smartphone an die WLAN Router sendet. Die Entfernung wird dann mit der erhaltenen Round-Trip-Time vom Smartphone berechnet.

### Alternative Standortbestimmung

Ein Einblick auf die aktuellen Ortungsverfahren lässt feststellen, dass zur Zeit entweder die Position nur ungenau bestimmt werden kann oder das weitere Hardware dazu nötig ist die Genauigkeit zu optimieren.

Verfahren die eingesetzt werden um den Innenraum-Standort zu bestimmen:

Technology	Indoor/Outdoor	Accuracy	Range	Cross-Platform	Power Supply
GPS					
WiFi					
Bluetooth					
VLC					

Abbildung 4: Verfahren zu Ermittlung des Innenraumstandortes [15]

Wie man auf Abbildung 4. erkennen kann wird WLAN bereits verwendet, die Ortung geschieht jedoch durch die Signalstärke (Abb.3):

„Zur Positionsbestimmung wird das sogenannte Fingerprinting herangezogen. Aussagekräftig sind hierbei die Stärke der WiFi-Signale (Received Signal Strength Indication, kurz RSSI) und die MAC- Adresse (Media- Access - Control). Auf dem Smartphone muss eine entsprechende App installiert sein, die aufgrund dieser Daten die aktuelle Position berechnet. Voraussetzung ist eine Datenbank mit Informationen über die Standorte, mit der diese Daten abgeglichen werden.“ [16]

Während die weiteren Verfahren wie das Beacons (Bluethooth) und VLC (Visible Light Communication) Verfahren eine genauere Positionsbestimmung ermöglichen als das WLAN RTT, sind zusätzliche Geräte notwendig die installiert und gewartet werden müssen.

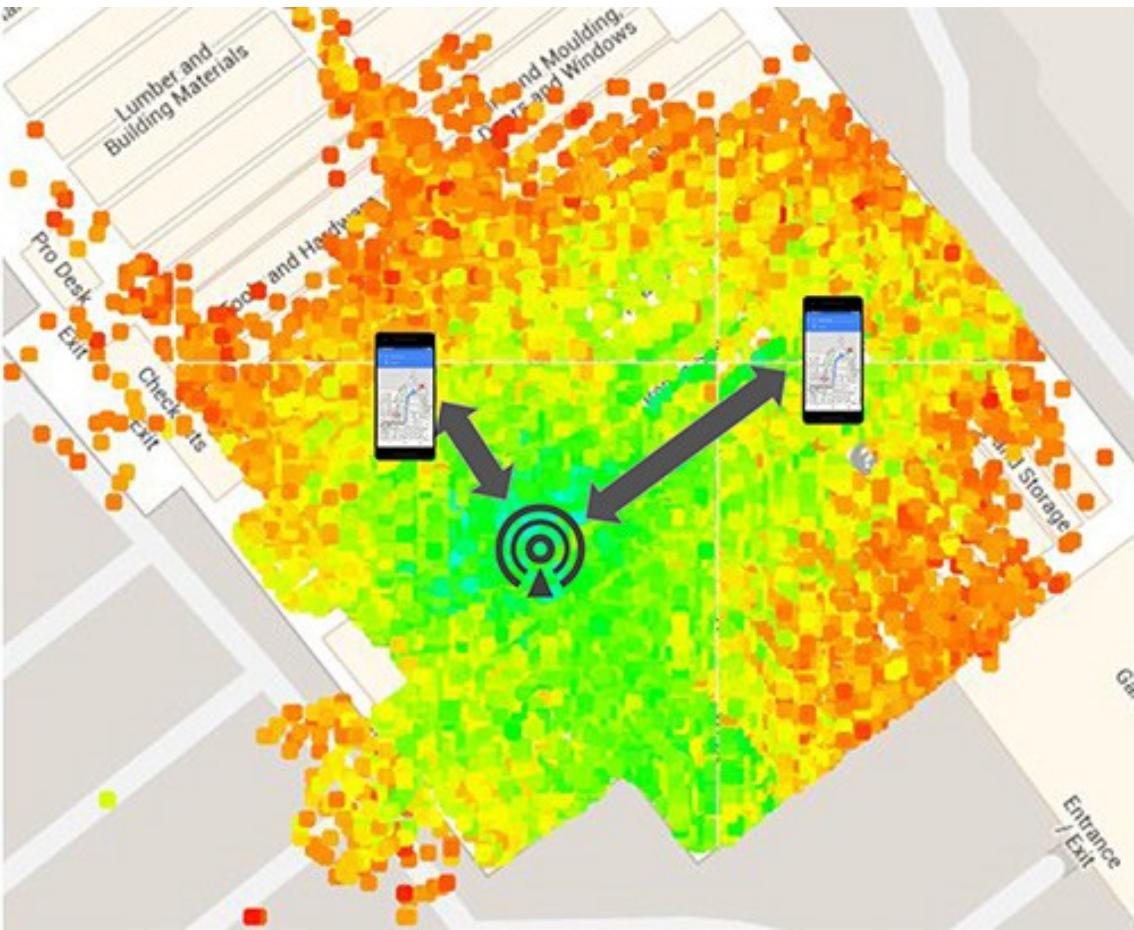


Abbildung 5: Positionsermittlung anhand von Signalstärke [17]

Deshalb ist die WLAN RTT Methode besonders geeignet, da die meisten Gebäude mit WLAN Routern ausgestattet sind.

Hinzu kommt der Kostenfaktor, die Entwicklung der RTT Kompatiblen Router wird sich auf alle WLAN Router ausbreiten und das Positionsbestimmungs- Gerät (das eigene Smartphone) hat heutzutage jeder Nutzer in der Tasche.

#### Aktuell verfügbare RTT Geräte: Compulab Wild

Zur Zeit (Stand Februar 2019) sind nur wenige Geräte für die RTT Methode vorhanden. Was die Router angeht bietet Compulab die ersten WLAN RTT kompatiblen Geräte an.



Abbildung 6: Compulab Wild Router [18]

### Google WiFi

Google hatte angekündigt die bereits auf dem Markt verfügbaren WLAN Router (Bild5) mit einem Softwareupdate auf WLAN RTT-Kompatibilität zu aktualisieren. Dieses Update lässt noch auf sich warten.



Abbildung 7: Google Wifi Router [19]

### Google Pixel 2/XL Android P.

Kompatible Smartphones gibt es aktuell (Stand Februar 2019) von Google (Google Pixel 2/XL , Bild 6). In naher Zukunft sind viele weitere Smartphones, die das Update auf Android P ermöglichen, ebenfalls kompatibel.



Abbildung 8: Google Pixel 2 [20]

### Das Wi-Fi RTT Verfahren /Funktion

Die Ortung des Smartphones beginnt mit einem WLAN- Scan. Das Smartphone erkennt verfügbare und RTT kompatible Router. Durch den vom Smartphone initiierten Ping startet der Router ein Ping- Pong Protokoll. Der Ping welcher vom Router zum Smartphone gesendet wird beinhaltet das FTM (Fine- Timing- Measurement) Paket. Der Pong, den das Smartphone an den Router schickt, ist lediglich die Bestätigung, dass das FTM Paket angekommen ist [21].

### FTM (Fine- Timing- Measurement)

Die Zeit, die ein WLAN Signal braucht um vom Smartphone zum Router zu gelangen, ist proportional zu der Distanz zwischen den Geräten. Da die Uhren der Geräte nicht synchron laufen könnten, ist eine unidirektionale Messung nicht ausreichend. Deshalb braucht man für die FTM ebenfalls die Zeit vom Router zum Smartphone. Als Resultat kann die RTT Zeit bestimmt werden ohne die genaue Uhrzeit der Geräte zu kennen. Mit einfacher Addition und Subtraktion aller vier Zeiten lässt sich diese berechnen. Für die Umwandlung in die Entfernung wird die RTT- Zeit mit der Lichtgeschwindigkeit multipliziert und halbiert.

$$\text{RTT} = (t_4 - t_1 + t_2 - t_3). [22]$$

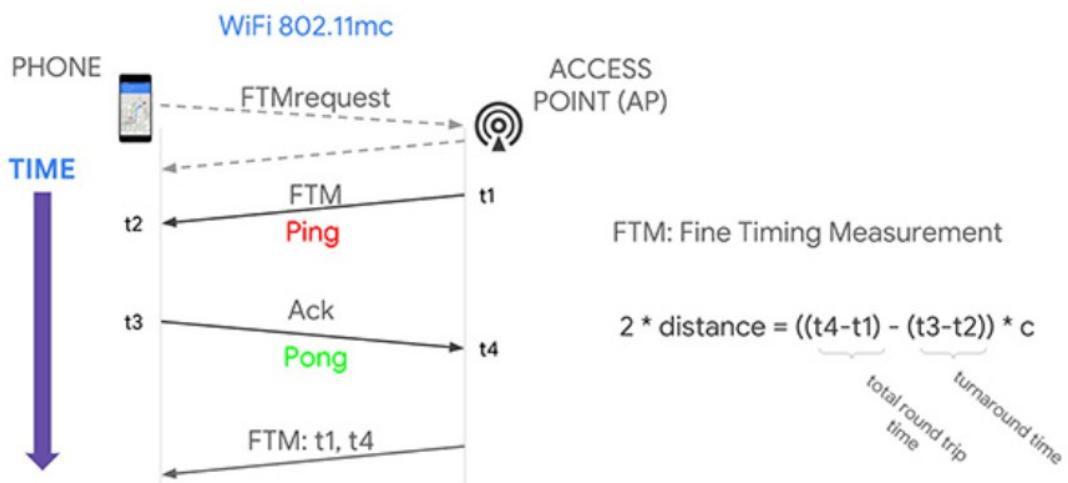


Abbildung 9: FTM Ablauf [23]

Auf Grund dessen, dass TCP architektonisch nicht sicherstellen kann das ein Paket auf dem Rückweg die gleiche Strecke nutzt wie auf dem Hinweg wird zur Erhöhung der Genauigkeit die RTT- Zeit mehrfach gemessen und anschließend gemittelt [24].

### Lateration

Lateration (lat. *lateral* = seitlich) oder Trilateration ist ein Messverfahren zur Positionsbestimmung eines Punktes. Während die Triangulation auf der Vermessung dreier Winkel basiert, beruht die Trilateration auf Entfernung- bzw. Abstandsmessungen zu drei Punkten [25].

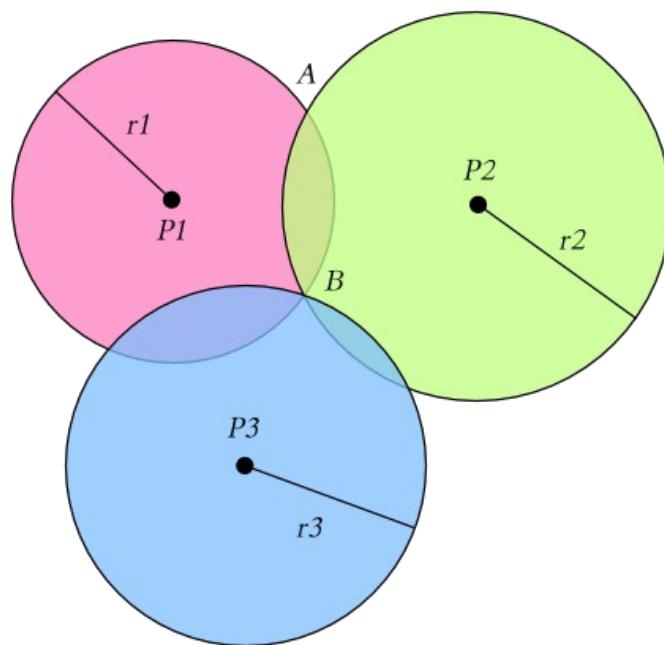


Abbildung 10: Trilateration [26]

Um letztendlich die Position des Smartphones zu bestimmen braucht man neben der berechneten RTT Zeit die Standorte der WLAN Router. Diese müssen bekannt sein und in einem Koordinatensystem hinterlegt werden.

In Bild 10 sind „P1-P3“ die Router mit bekannten Standorten. Die RTT- Zeit = Entfernung vom Router ist hier jeweils „r1 – r3“. Somit ermittelt sich der Standort des Smartphones „B“ durch den Schnittpunkt der Kreise.

## Schwierigkeiten und Herausforderungen

Neben den Problemen, dass aktuell nur Compulab RTT-fähige Router anbietet und Android P nur auf den neusten Google-Pixel Smartphones verfügbar ist, ist zu erwähnen, dass Android P eine Limitierung auf den WLAN Scan, der nach den verfügbaren Routern sucht, hat. Diese Limitierung legt fest, wie oft ein WLAN Scan vom Smartphone getätigt werden kann. Die Rate liegt bei 4 mal alle 2 Minuten. Mit einer durchschnittlichen Schrittgeschwindigkeit von 1,4m/sec, bedeutet das alle 42 Meter. Die Wahrscheinlichkeit da bereits aus dem Gebäude zu sein bevor man einen Router in der Umgebung orten kann, liegt relativ hoch [27].

### **2.6.3 GPS - Global Positioning System**

Für das Projekt stehen Google Pixel 2 Smartphone Geräte zur Verfügung welche zur Ortung einen Qualcomm Snapdragon 835 Chip verbaut haben [28]. Dieser bietet Möglichkeiten zur Verwendung von Satelliten Empfänger System wie folgt: GPS, GLONASS, BEIDOU, GALILEO, QZSS & SBAS.

Um mit den empfangenen Daten umgehen zu können sei hier kurz auf die Geschichte und Technik eben dieses eingegangen.

In den 70er Jahren vom US-Verteidigungsministerium entwickelt löste es ca. ab 1985 andere Satellitennavigationssysteme wie das NNSS der US-Marine ab. Ab 1990 war es voll funktionsfähig und seit der Abschaltung der künstlichen Signalverschlechterung im Mai 2000 auch für die zivile Bevölkerung frei empfänglich [29].

Die Position (GPS-Gerät) wird durch Entfernungsmessung zu mindestens drei Satelliten bestimmt. Es handelt sich hierbei um eine unidirektionale Verbindung, ein Signal das stetig von Satellit an Erde ausgestrahlt wird und vom Empfangsgerät aufgefangen wird.

Die Schnittpunkte der Distanzvermessungen werden innerhalb eines eigenen kartesischen Koordinatensystems lokal zum Standpunkt errechnet.

Da sich ein Satellit ca. mit 4 km/s auf seiner Erdumlaufbahn bewegt und somit nicht still steht ist es nötig zu wissen wann die Messung vorgenommen wurde. Aus diesem Grund werden neben Positions- auch Zeitdaten in Form der GPS-Systemzeit mitgeschickt. In der Satellitennachricht sind die Bahnparameter jedes Satelliten enthalten. Mit ihnen und der Sendezeit kann für jede Sendezeit der Satellitenort berechnet werden. Positionsbestimmungen sind bis auf 10m genau und durch den Einsatz von DGPS (Differential GPS), bei welchem Korrekturdaten von der Erde aus ausgestrahlt werden, sogar noch konkretisierbarer. Bei DGPS handelt es sich um festinstallierte Antennen welche als Referenzstation dienen.

## 3 Umsetzung

### 3.1 Unity

Für die technische Umsetzung unserer Projektarbeit haben wir uns aufgrund der geplanten Funktionen unserer App für die Verwendung eines Gaming Engine Tools entschieden. Hierzu haben wir verschiedene Tools ausprobiert und miteinander verglichen. Unity3D hat uns hierbei am meisten überzeugt.



Abbildung 11: Unity Logo [30]

#### 3.1.1 Grundfunktionen

Unity ist eine multiplattform Gaming Engine mit der sich sowohl 3D als auch 2D Spiele und Anwendungen für Web, Desktop und Konsolen entwickeln lassen.

Die Standard Version von Unity ist kostenfrei, kann aber auch als kostenpflichtige Pro Version erworben werden. Jedoch genügt die kostenlose Standartversion vollkommen aus, da diese den Entwickler mit allen essentiellen Tools ausstattet um vollwertige Spiele, Apps sowie Anwendung für Virtuell Reality und Augmented Reality zu entwickeln. Für den Programmieranteil greift Unity auf C# oder JavaScript zurück [31].

Das Erlernen der Grundkenntnisse und erweitern des Wissensstands im Umgang mit Unity, wird dem Entwickler durch die ausführlichen Text Dokumentationen und zahlreichen Online-Kurse und Tutorials auf der Firmeneigenen Website erleichtert [32].

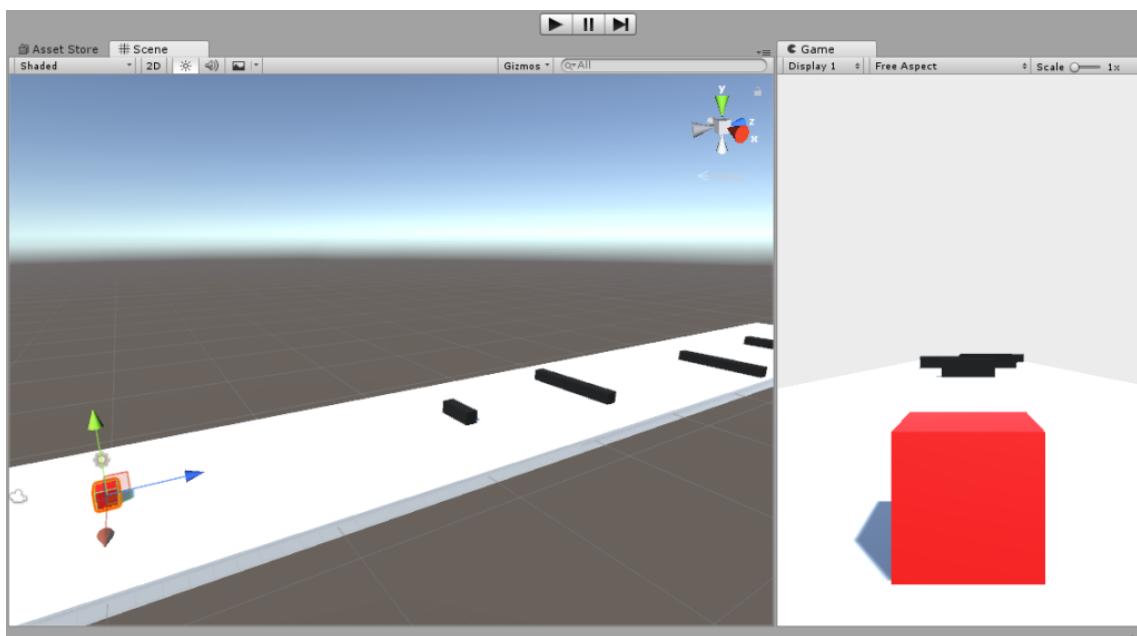


Abbildung 12: Unity Oberfläche

Das benutzerfreundliche User Interface ermöglicht Neulingen einen unkomplizierten Einstieg im Umgang mit dem Tool. Die übersichtliche Menüführung verschafft dem Entwickler einen guten Überblick über sein Projekt. Zudem erhält man über den Scene- und Game-Editor eine Voransicht seiner Arbeit und kann diese auch direkt testet.

Der im Programm integrierte Asset Store bietet eine große Anzahl an vorgefertigten Tools, Audio Files, 3D Modellen, Animationen, KIs und vieles mehr. Diese kann man entweder kostenlos oder gegen Bezahlung herunterladen und direkt in seinem eigenen Unity-Projekt integrieren [33].

Eine stetig wachsende Unity-Community erweist sich durch zahlreiche Foren und Blogs als große Hilfe, sollte man mit seinem Projekt ins Stocken geraten.

Nachdem sich unsere Projektgruppe mit den Grundfunktionen von Unity vertraut gemacht und eingearbeitet hat, haben wir mit der Entwicklung des Prototypen für unser Projekt begonnen.

### **3.1.2 Objekt auf einer Route bewegen lassen**

Um den späteren Avatar auf der Karte auf definierten Routen umher wandern lassen zu können müssen zunächst Punkte auf der Karte gewählt werden die als Start- und Zielpunkt fungieren. Beim Spielestart positioniert sich der Avatar bereits auf festgelegten Startpunkt und für jeweilige Zielpunkte können später z.B. Ausstellungsstücke, Toiletten oder Ein- und Ausgänge angepeilt werden. Für das bewegen des Objektes stellt Unity Funktionen zur Verfügung welche im Näheren erläutert werden.

Für das setzen des Startpunktes wird sich, im an Objekt angefügtem Script, auf das Objekt selbst mit *this* bezogen und seine momentane Position abgegriffen. Hierzu wird die, durch Unity bereitgestellte, Klasse Vector3 genutzt.

```
Vector3 startPunkt = this.transform.position;
```

Der Zielpunkt wird mit 3-dimensionalen Koordinatenwerten X, Y, Z willkürlich bestückt wobei Y die Höhe, X und Z die Fläche selbst darstellt. Zur Instanziierung werden Float-Werte benötigt.

```
Vector3 endPunkt = new Vector3(125f, 4f, 80f);
```

Für die Berechnung der Wegstrecke muss die Uhrzeit des Funktionsaufrufes entnommen werden und einmalig gesetzt werden.

```
float startZeit = Time.time;
```

Die Distanzberechnung zwischen den zwei Punkten wird wieder mit Unity-Funktionen gelöst.

```
float distanz = Vector3.Distance(startPunkt, endPunkt);
```

Die Berechnung für das Fortschreiten des Objektes von startPunkt nach endPunkt, welcher als Weg Teil zu verstehen ist, steckt in einem Loop welcher so lange ausgeführt bis der errechnete Wert in Form eines floats den Wert 1 und somit 100% der Wegstrecke erreicht hat. Ein Wert von 0 markiert den Anfang, ein Wert von 0.5 die Mitte des Weges.

Hierbei wird von der aktuellen Zeit die startZeit bei Funktionsaufruf subtrahiert und mit der willkürlich gewählten Geschwindigkeit multipliziert.

```
float geschwindigkeit = 80f;
float zurueckgelegteDistanz = (Time.time - startZeit) *
    geschwindigkeit;
```

Als nächstes die zurückgelegte Distanz noch ins Verhältnis zu gesamten Wegstrecke genommen. Daraus ergibt sich eine Prozentuale Berechnung der Gesamtwegstrecke.

```
float wegStueck = zurueckgelegteDistanz / distanz
```

Als letzter Schritt wird das Objekt durch zur Hilfenahme der Funktion *Lerp()* für jede durchlaufene Schleifeniteration um ein *wegStueck* näher vom startPunkt zum endPunkt positioniert wobei das zuvor ausgelesene Attribut *transform.position* neu gesetzt wird.

### 3.1.4 Objektradius auf andere Objekte prüfen

Für die Prüfung auf relevante Objekte im Sinne von Ausstellungsstücken auf der Karte muss ein Indikator für jedes Stück zur Verfügung stehen welches während der gesamten Dauer der Anwendung abgefragt werden muss. Hierfür wird jedem Objekt, welches ein Ausstellungsstück repräsentiert, über den Objektinspektor ein *Tag* gesetzt. Über das Anwählen der Tag-Option lassen sich bereits bestehende Kategorien wählen oder auch eigene Namen, über „*Add Tag...*“ vergeben.



Abbildung 13: Unity „Add Tag“

Für die eigentliche Abfrage nach Objekten in der Nähe des Spieleraufenthaltsort wird dem eben diesem ein Script angehängt in welchem alle Kalkulationen hierfür durchgeführt werden. Hierfür werden folgende Variablen vormals benötigt um auch in Kontext der Unity Funktionsweise logisch und sinnvoll operieren können.

In der Startfunktion des Scripts wird einmalig für jeden Start der Applikation alle Objekte mit dem gesetzten Tag: *Ausstellungsstück* in ein Array des Datentyps *GameObject* geladen.

```
private GameObject[] ausstellungsStuecke =  
GameObject.FindGameObjectsWithTag("ExhibitionPiece");
```

Es wird des weiteren ein Boolean Array benötigt in welchem der Status für „in der Nähe“ für jedes Instanz der *ausstellungstücke*-Objekte separat gespeichert werden muss. Diese werden zu Beginn alle auf den Wert false gesetzt. Ein weiterer Boolean welcher als

Schleifeneinstieg genutzt in der Updatefunktion wird den Start einer erneuten Abfrage einläutet, wird benötigt.

```
private bool[] ausstellungsStckInNaehe =  
    new bool[ausstellungsStuecke.Length];  
private bool pruefeNaheObjekte = true;  
  
private float updateIntervall = 1.0f;  
    //Updateintervall beliebig bestimmen  
  
private float radius = 50f;  
    //Radius für Objektnähe beliebig bestimmen
```

In der Updatefunktion des Scripts wird eine einfache if-Abfrage nach dessen Status durchgeführt welche im Falle von true in die weitere Schleife springt. Im Falle dessen wird die Funktion aufgerufen:

```
StartCoroutine(„findeNaheAusstellungsStueck“);
```

Eine *COROUTINE* kann nur aus einer Update-Funktion heraus aufgerufen werden, welcher zusätzlich noch als Übergabeparameter ein String mit dem Namen einer weiteren Funktion mitgegeben wird.

Hintergrund ist, dass die Update-Schleifeniteration mit einer *COROUTINE* nicht verlassen und zum späteren Zeitpunkt weiter aufgenommen wird, sondern eine synchron laufende weitere Schleife mit Namen des übergebenen Strings hervorruft. Wichtig hierbei ist, dass die die Funktion welche aufgerufen wird den Rückgabetyp *IEnumerator* aufweist.

```
private IEnumerator findeNaheAusstellungsStueck() { ... }
```

Sobald die Schleife für das Prüfen der naheliegenden Ausstellungsstücke erstellt wurde, wird zuerst der Boolean *pruefeNaheObjekte* auf false gesetzt und verhindert somit ein unendliches Erzeugen von *COROUTINEN* aus der Updatefunktion heraus da dies wie eben erwähnt der Einstiegsparameter aus der sich unendlich wiederholenden Updatefunktion ist.

Nun kommt es zum eigentlichen Prüfen der Objekte welche auf der Karte verteilt sind und bei der Startfunktion für die Abfrage verfügbar gemacht worden sind. Eine *FOR*-Schleife über die Anzahl des *ausstellungsStuecke* Array wird begonnen in welcher folgende Berechnungen passieren. Spielerposition wird von *this*, bezogen auf Objekt welchem Script angehängt wurde, abgefragt.

```
Vector3 spielerPosition = this.transform.position;
```

Differenz Vektor zwischen Ausstellungsstückposition und Spielerposition wird durch einfache Subtraktion der Vektoren ermittelt.

```
Vector3 distanzVektor =
ausstellungsStuecke[i].transform.position -
spielerPosition;
```

Für die Berechnung des benötigten Vektorbetrages wird Pythagoras bemüht.

$$\text{Distanz}^2 = x^2 + y^2 + z^2$$

```
float distanz * distanz = distanzVektor.sqrMagnitude
```

$$\text{Distanz} = \sqrt{x^2 + y^2 + z^2}$$

```
float distanz = Mathf.Sqrt(distanzVektor.sqrMagnitude)
```

Danach erfolgt die Abfrage ob sich der errechnete Distanzbetrag von Spieler zu Ausstellungsstück im gewünschten Radius liegt sowie den Boolean *ausstellungsStckInNaehe* für i-tes Objekt prüft.

```
if ((distanz < radius) && (ausstellungsStckInNaehe[i] ==
false) ) { ... }
```

Befindet sich das Objekt im Spielerradius, *distanz* kleiner als *radius*, so wird für entsprechendes Objekt im BooleanArray *ausstellungsStckInNaehe* der Wert auf *true* gesetzt sowie ein möglicher Auslöser hier betätigt welcher verschiedene Aktionen in der Applikation hervorrufen kann.

Würde der Boolean-Wert nicht auf `true` gesetzt werden würde die oben aufgeführte Schleife für jede Iteration der Distanzprüfung den Auslöser für „in Reichweite“ auslösen. Mit dieser einfachen Schranke wird eben dies verhindert und nur das erstmalige Nähern zum gewünschten Verhalten führen.

Verlässt der Spieler, des sich im Radius befindlichen Objektes Nähe wird in einer weiteren Abfrage der erneut Distanz zum Objekt prüft der Boolean-Wert auf `false` gesetzt.

```
        else if ((distanz > radius) &&
    (ausstellungsStckInNaehe [i] == true) )
    { ausstellungsStckInNaehe [i] = false; }
```

Zum Erreichen des Endes der `IEnumerator` Funktion wird mit dem Aufruf der `WaitForSeconds(updateIntervall)` Funktion das Updatezeitverhalten nach Wunsch geändert.

Ein weiteres Boolean-Array könnte dafür genutzt werden um alle Ausstellungsstücke welche besucht wurden auf `true` zu setzen und somit zwischenzeitlich Erinnerungen anzeigen zu können falls der Spieler sich in Richtung Ausgang bewegt, so dass er , im Falle nicht Besuchens eines Ausstellungsstückes eine Erinnerung erhält welche in darauf hinweist und ob eine Navigation dorthin gestartet werden soll.

### 3.1.4 Unity GPS Abfrage

Um GPS Daten für die Positionierung des Spielerobjektes auf der Karte beziehen zu können, musste vorher eruiert werden inwiefern diese aus Unity heraus verfügbar gemacht werden können. Drei Möglichkeiten ergeben sich aus Recherche auf der Unity Dokumentation Website [34].

## 1.0 LocationService API [35]

Eine einfache Einbindung wird hier direkt unter der Einbindung der LocationService Library ermöglicht. Nach Implementierung der vorgegebenen Anleitung wird zunächst geprüft ob der User den Ortungsdienst im Allgemeinen aktiviert hat. Ist dies nicht der Fall wird direkt mit einer Fehlermeldung abgebrochen. Fortfolgend wird im Falle der bereits vorherigen Aktivierung mit einem einfachen Methodenaufruf das Modul über `Input.location.Start();` aufgerufen.

Danach wird dem Modul ein Zeitfenster im eigenen Ermessen zur Initialisierung bereitgestellt in welchem er durch Iteration einer Schleife den Zustand des Moduls und den Zeitzähler prüft.

So lange sich der Zustand in `LocationServiceStatus.Initializing` und Rest Zeit vorhanden ist, wird gewartet. Das Warten wird mit der Funktion `WaitForSeconds(1);` gelöst.

Ist nach Ablauf des Timers keine Initialisierung möglich, wird der Service Status auf `LocationServiceStatus.Initializing` geprüft. Auch in diesem Fall wird eine Fehlermeldung ausgegeben welche verdeutlicht das der Vorgang fehlgeschlagen ist.

Durchläuft die Initialisierungsfunktion mit Erfolg, lassen sich die benötigten Längen- und Breitengrad Parameter, in Form eines Doubles, einfach rückgeben lassen:

`Input.location.lastData.latitude`      `Input.location.lastData.longitude`

Im Falle dessen, dass der Zugriff auf das Ortungsmodul nicht mehr von Nöten ist, kann dieses mit `Input.location.Stop();` terminiert werden.

## Eigene Versuche

Nach Übernahme des Codes in das Projekt wurde schnell klar das die Schnittstelle nicht sauber und zuverlässig funktioniert was zu guter Letzt diverse Foreneinträge bestätigten. Unter Verwendung von verschiedenen Endgeräten (Motorola XT1633, Samsung Galaxy S8, Samsung Galaxy S7 und Google Pixel 2) konnte dies erneut beobachten werden.

Falsche Positionsdaten in unregelmäßigem Abstand zu unregelmäßigen Bewegungsdistanzen waren zu verzeichnen.

Es wurde des Weiteren versucht die Schnittstelle für jede Standortabfrage erneut zu starten und nach erfolgreicher Positionsabfrage diesen wiederum zu beenden um herausfinden zu können wo der Fehler genau liegen könnte. Leider hat auch diese Herangehensweise keine Läuterung erbracht. Für ein planmäßiges voranschreiten in eingehaltenem Zeitplan wurde dann schnell nach einer anderen Lösung gesucht.

## **3.2 Android Studio**

Android Studio ist eine kostenlose Entwicklungsumgebung (IDE) welche von Google speziell für das Betriebssystem Android implementiert wurde. Ein Grund weshalb wir diese IDE verwendet haben, war die Kompatibilität mit dem Betriebssystem des Android 9.0 Pie und der App Entwicklung für unser RTT Projekt.

Da wir schon einige Erfahrung mit der Sprache Java machen konnten, war es ein weiterer Vorteil, dass die Sprache der Entwicklungsumgebung in Java möglich war.

Außerdem lies sich eine gute Dokumentation zu verschiedenen Themengebiete im Bereich der Android Entwicklung finden.

### 3.2.1 Android Library Plug-In für Unity

Nachdem der Versuch über die direkte Abfrage der in Unity verfügbaren Standortabfrage API zu keinem Ergebnis geführt hat wurde ein anderes Konzept in Form einer, in Android Studio erstellten Bibliothek, welche die Funktionen auf Betriebssystemebene bereitstellt, ausprobiert.

Grundidee dieses Verfahrens ist aus Android Studio eine Java-Klasse, welche alle benötigten Funktionen bereitstellt, in eine Bibliothek einzubauen, diese als .JAR (Java Archive) oder AAR (Android Archive) zu kompilieren und in Unity als externe Bibliothek einzubinden.

Da es sich bei der dieser Methode um eine komplexere Angelegenheit handelt wurde zunächst versucht aus Unity lediglich auf eine Funktion der Bibliothek zuzugreifen welche einen einfachen String mit dem Inhalt „Zugriff erfolgt“ zurückgibt.

In Android Studio kreiert man dazu über /Datei/Neu/Modul ein neues Modul bei welchem im Erstellungsfenster „AndroidLibrary“ gewählt werden muss. Nachdem die IDE einige Zeit braucht um benötigte Dateien hinzuzufügen, erweitert sich das die Projekt Ordnerhierarchie.

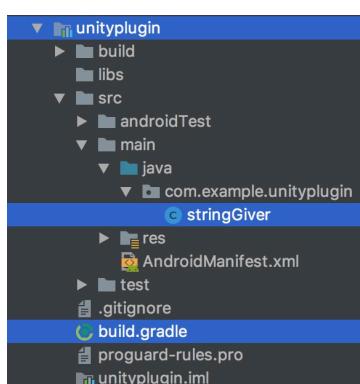


Abbildung 14:  
Ordnerstruktur „src“

Das Plug-In trägt wie gewählt den Namen: „unityplugin“. Im src-Ordner befindet sich wiederum in Subordnern die Java-Klasse „StringGiver“ in welcher der Library zugehörige Code geschrieben werden muss. In der build.gradle Datei muss zusätzlich noch eine Änderung erfolgen um dem Compiler mitzuteilen, dass es sich hierbei lediglich um eine Library und nicht um eine Activity handelt.

```
apply plugin: `com.android.app`  
—> Änderung in: apply plugin: `com.android.library`
```

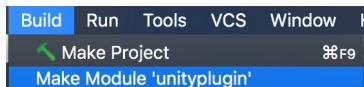


Abbildung 15: Kompilieren des Moduls

Nach Erstellung einer trivialen Funktion welche nur einen String zurückgibt kann über Menü statt dem ganzen Projekt, das Modul zum kompilieren angestoßen werden.

In der Projekteigenen Ordnerstruktur unter „outputs“ wird im Anschluss die benötigte .aar Datei generiert. Diese wird später in Unity per Drag'n'Drop in den Assets/Plugins/Android abgelegt.

Es sei hier kurz auf den Unterschied zwischen .JAR und .AAR eingegangen. Beide Files beinhalten Java Dateien und sind lediglich komprimiert. Wird der Suffix .aar oder .jar in .zip umbenannt kann diese entpackt werden und ein Ordner mit mehreren Dateien erscheint. In einem aar. File befindet sich ein JAR Datei sowie weitere Metadaten, darunter auch das AndroidManifest.xml welches unter anderem Anweisungen für Zustimmungen bei Nutzung der Bibliothek inne hält.

Mit diesem Wissen wird zukünftig für die Komprimierung von Java Klassen Dateien keine IDE mehr benötigt, sondern lediglich ein zip-er Programm dessen Ausgangssuffix nur noch von .zip in .jar umbenannt wird.



Abbildung 16:  
Inhalt entpackter .aar

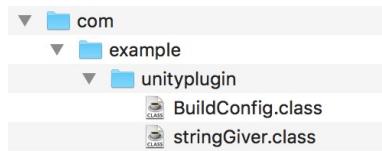


Abbildung 17:  
Inhalt entpackter .jar

In einem .jar File befinden sich diverse Subordner welche die gleiche Struktur aufweisen wie der Package-name in der Klasse selbst. In unserem Fall somit: „com.example.unityplugin“.

Unity empfiehlt in ihrer Dokumentation die Verwendung der AAR-Files und nicht JAR-Files.

### 3.2.2 Android GPS Abfrage

#### Zugriff aus Unity

Nachdem das Android Archive in den Unity Assets abgelegt wurde, muss nun aus einem Skript heraus, darauf zugegriffen werden. Unity stellt hierfür zwei Klassen zur Verfügung in welche bei Instanziierung ein String, bestehend aus KlassenPackagename.Klassenname, übergeben wird.

Da es sich bei der erzeugten Funktion um eine statische Funktion handelt reicht es aus, diese als „AndroidJavaClass“ zu Initialisieren. Wäre, wie später benötigt, ein Objekt mit zugehörigen Methoden von Nöten müsste die „AndroidJavaObject“ Klasse genutzt werden.

```
var plugin =  
new AndroidJavaClass("com.example.unityplugin.StringGiver");
```

Für das Aufrufen der Funktion wird das instanzierte Objekt unter Angabe des zu erwarteten <Rückgabewertes> benannt. Die Funktion wird hierbei lediglich als String-Paramater mit übergeben.

```
plugin.CallStatic<string>("returnString");
```

Das Programm, auf dem Rechner selbst ausgeführt, gibt den String nicht zurück. Erst beim Build der Applikation auf dem Smartphone wird dieser ausgegeben.

#### Einbindung GPS in Java

Nachdem der erste Versuch mit primitiven Datentypen geglückt war, wurde versucht GPS in Java verfügbar zu machen. Für der Einbindung der GPS Funktionalität in die Android Library ist es aufgrund der Architektur seitens Android notwendig den LocationManager,

welcher als Handle für den GPS Chip fungiert, entweder aus einem „Kontext“ oder aus einer „Activity“ heraus zu starten.

Somit gilt die Voraussetzung GPS aus der aktiven Applikation heraus aufzurufen.

Für die benötigte Java Klasse war dieser jedoch nicht verfügbar. Weiter Recherchen nannten die Möglichkeit eine Klasse des Java-Plugins als Einstieg zu nutzen unter Vererbung der UnityPlayerActivity [36].

### 1. Unity-Java-Library erstellen.

Hierzu wird ein bestehendes Unity Projekt, egal in welchem Bearbeitungsstand, zunächst in ein extern weiterentwickelbares Projekt exportiert. Die zuvor in Unity eingestellte Zielplatform Android wird nun als Android Studio Projekt exportiert.

#### *Unity : File/Build Settings*

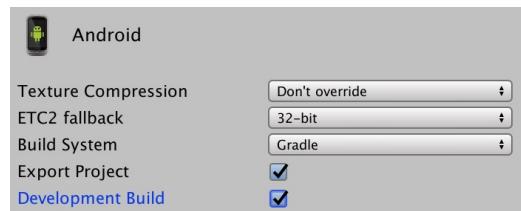


Abbildung 18: Unity Zielplatform wählen

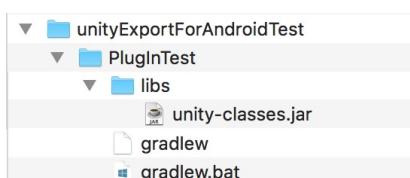


Abbildung 19: Library einbetten

2. Library in Android Projekt einbetten. Die Library „unity-classes.jar“ welche unter rechts stehendem Pfad zu finden ist muss in das Android Projekt eingebettet und als Library „hinzugefügt“ werden.

### 3. Die aktuelle Java-Klasse kann nun von UnityPlayerActivity erben und das GPS Modul implementiert werden.

4. Im build.gradle muss nun noch die Zeile „compileOnly: Pfad zu Bibliothek“ unter *Dependencies* angegeben werden was bewirkt das nur benötigte Bibliothek Bausteine in das Plug-In Einzug erhalten.
5. Als Eintrittspunkt der App wird im AndroidManifest.xml Datei des Plug-Ins noch die Klasse als Start deklariert werden.

```
<activity android:name=".OverrideExample" android:label="@string/app_name">
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

Nach erfolgreichem erstellen des Plug-Ins, Import und Aufruf in Unity war das Ergebnis leider ernüchternd. Nach weiteren Recherchen um den GPS Abruf zu ermöglichen bzw. was dessen Fehler verursacht stößt man immer wiederkehrend auf JNI, das Java Native Interface welches die Schnittstelle, den Kommunikationskanal, für Unity mit dem Java-Code darstellt. Leider waren weitere Versuche erfolglos, und für somit den weiteren Fortschritt eine Sackgasse darstellt.

### Projektexport aus Unity in Android Studio

Als letzte Option bietet Unity über die bereits oben erwähnte Android Studio Exportfunktion die Möglichkeit ein bereits fertiges Unity Projekt für nachträgliche Erweiterungen aufzurüsten. Großer Nachteil an dieser Variante ist jedoch dass nachdem Unity verlassen wird nachträgliche Änderungen über einen „Reimport“ darin nicht oder nur schwer möglich sind. Auch der direkte Zugriff auf 3D Objekte welche bereits Teil des Unity Projekts waren werden nicht einfach zugänglich gemacht sondern müssen auch hier über eine Plug-In bereits in Unity miteingebunden werden.

## Fazit

Leider hat sich Unity als Entwicklungsplattform in dieser Situation als Sackgasse erwiesen. Trotz des notwendigen Wechsels auf eine andere Plattform waren die Einblicke in dieses umfassende 3D Grafiktool nicht nutzlos.

## 3.2 Ortsabfrage und Berechnungen

### 3.2.1 Geoid/Ellipsoid Berechnung

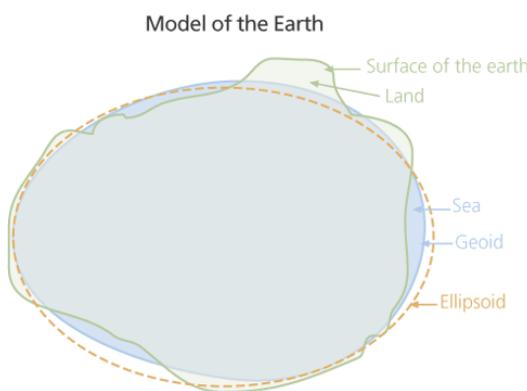


Abbildung 20: Erdreferenzellipsoid [37]

Um eine möglichst genaue Positionierung über GPS auf der Karte zu ermöglichen reicht die Konvertierung des Längen- und Breitengrades zum ECEF Koordinatensystem welche im Anschluss aus dem globalen in ein lokales Koordinatensystem überführt wird nicht aus. Die errechneten Koordinaten für den

3-dimensionalen Vektor (X,Y,Z) liegen auf dem Erdreferenzellipsoid, in Realität jedoch kann dies um einige Meter höher oder tiefer dessen sein.

Aus nebenstehender Grafik wird das Problem ersichtlich, dass sich das Geoid auch am MSL (Main Sea Level) orientiert, bzw. diesen berücksichtig. Nehme man als aktuellen Standort eine Position auf der Erhöhung oder auf einem Berg so wäre das Ergebnis um mehrere 100 Meter fehlerhaft.

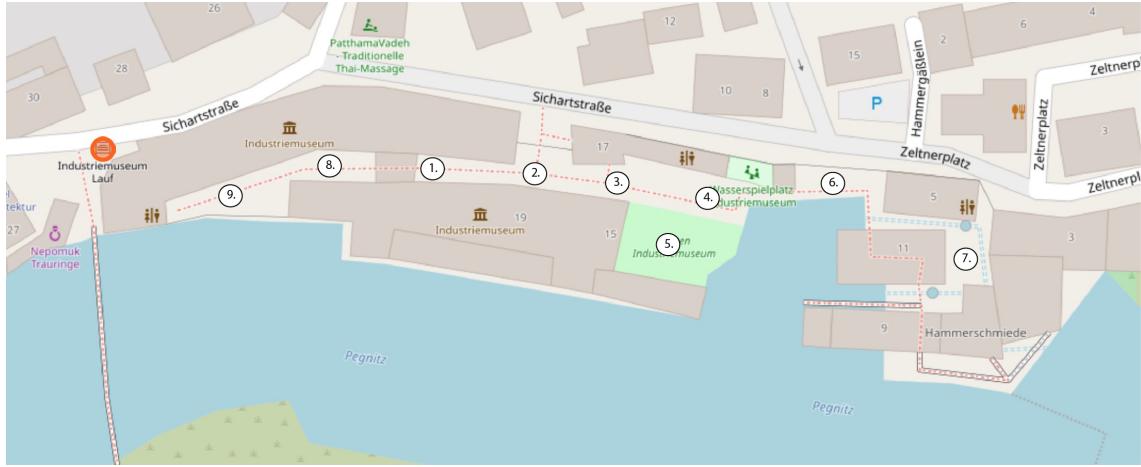


Abbildung 21: Geländeumriss des Industriemuseum Lauf

An den in Kreisen mit Nummern markierten Positionen wurden jeweils die geographischen Koordinaten in Form von Längen- und Breitengrad entnommen.

Gemessene Werte wurden danach mit der Ellipsoid/Geoid Höhendifferenzberechnung des EGM2008 welche von der NATIONAL GEO-Spatial-INTELLIGENCE AGENCY (NGA) im Internet frei zugänglich gemacht wurde, berechnet.

Auch eine Berechnungsmöglichkeit über das EGM84 ist frei verfügbar. Dieser unterscheidet sich jedoch zum EGM96 stark, da dieser auf dem Ellipsoid Referenzmodell basiert und im Vergleich zum späteren (genauerem) EGM96 nur eine vereinfachte Annäherung an die Erde hat. Das komplexe Modell welchem die Erdgravitationskraftmessung zu Grunde liegt war somit die genaueste Geoid-Annäherung bis 2008, bevor das EGM2008 publiziert wurde. Auch dieses war seinem Vorgänger in Präzision wieder überlegen.

In unserem Fall war der EGM84 dem EGM96 um ca. 70cm, und wiederum der EGM2008 dem EGM96 um ca. 8cm überlegen.

<b>Mar. k.</b>	<b>Längengrad</b>	<b>Breitengrad</b>	<b>EGM84</b>	<b>EGM96</b>	<b>EGM2008</b>
1.	49.509929	11.276383	47.4412m	46.7566m	46.6747m
2.	49.509927	11.276677	47.4412m	46.7564m	46.6747m
3.	49.509915	11.276914	47.4411m	46.7563m	46.6746m
4.	49.509894	11.277122	47.4412m	46.7561m	46.6745m
5.	49.509820	11.277045	47.4410m	46.7561m	46.6745m
6.	49.509897	11.277369	47.4410m	46.7559m	46.6745m
7.	49.509771	11.277757	47.4408m	46.7556m	46.6743m
8.	49.509936	11.276039	47.4412m	46.7568m	46.6748m
9.	49.509809	11.275712	47.4412m	46.7569m	46.6747m

Abbildung 22: Tabelle Ergebnisse Ellipsoid/Geoid Höhendifferenzberechnung

Die Entscheidung hierbei auf eine Neuberechnung der Differenzen dürfte aus obenstehenden Ergebnissen klar ersichtlich sein. Die Werte ändern sich marginal was somit als Irrelevanz betrachtet werden kann. In unserem Falle der bereits aufwändigen Grafik- und Standortabfragen Berechnung führt eine weitere Abfrage einer Höhendifferenztabelle nur zu Prozessorbelastung die später anderweitig sicherlich noch von Bedeutung sein wird. Trotz alle dem sollte nicht unerwähnt bleiben, dass Versuche unternommen worden sind, eine solche globale Tabelle zu erstellen, diese in ein passendes Raster (Franken) zu überführen und anschließend über ein entsprechendes Script nach jeder Standortberechnung abzufragen.

Ein Abgriff der GPS Daten mit dem Android Gerät welcher erst im späteren Verlauf des Projektes erfolge wird hier jedoch der Verdeutlichung halber aufgeführt.

```

2019-02-08 13:02:52.009 1241-1262/? I/
GnssLocationProvider: WakeLock released by
handleMessage(REPORT_LOCATION, 1, Location[gps
49,448100,11,095885 hAcc=14 et=+8m34s830ms
alt=361.0426025390625 vel=2.02 bear=263.5 vAcc=8 sAcc=2
bAcc=44 {Bundle[{satellites=10, maxCn0=34,
meanCn0=26}]})
// Android GPS Ausgabe auf dem Testgelände der TH Nürnberg

```

Besagt: 361m – WGS96 (46,8m) Differenz ist aktuelle  
Höhenmeterangabe: ~ 311m

Als Alternative könnte vor Ort über ein Barometer der Luftdruck  
gemessen werden und daraus die Höhe errechnet werden.

### 3.2.2 Koordinatensystem Global to Local

Damit wir für das Lokale Koordinatensystem unserer Applikation vernünftige Werte erhalten sind zwei Schritte notwendig [38]. Zuerst müssen wir die geodätischen Koordinaten (Längen- und Breitengrad) in ECEF Koordinaten umwandeln. Dadurch erhalten wir einen Punkt auf der Erdoberfläche. Die hierdurch erhaltenen ECEF Werte sind jedoch zu groß um diese direkt für unser Lokales Koordinatensystem zu verwenden zumal ihr Masse-schwerpunkt im Zentrum der Erde liegt [39].

Daher müssen wir die ECEF Koordinaten in ENU Koordinaten umrechnen. Zuvor möchten wir Kurz erläutern was es mit ECEF Koordinaten auf sich hat.

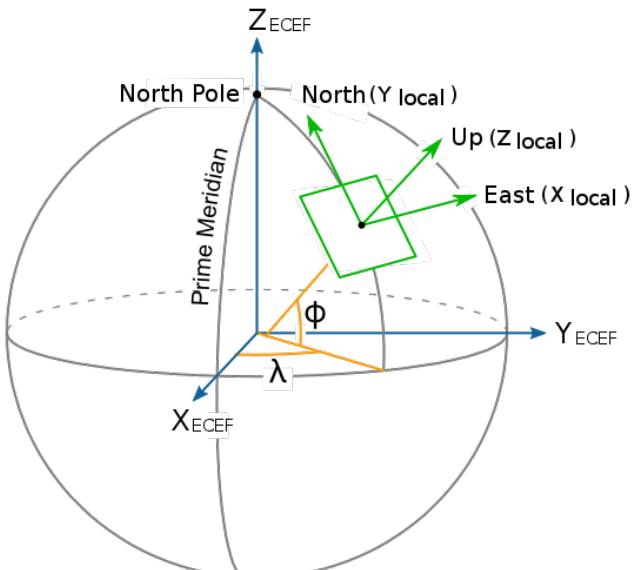


Abbildung 23: Koordinatensysteme [40]

### 3.2.3 Umwandlung GPS zu ECEF Koordinaten

In der obigen Abbildung sieht man ein Geozentrische Koordinatensystem ECEF Anhand der Erdkugel. Es besitzt eine X, Y sowie eine Z Koordinate und hat seinen Ursprung im Erdmittelpunkt. Die Z-Achse ist gen Norden ausgerichtet. Der Schnittpunkt der Äquatorebene und des Nullmeridian ergeben die Ausrichtung der X-Achse. Die Y-Achse liegt in einem Winkel von  $90^\circ$  senkrecht in Richtung der Äquatorebene [41].

#### Konvertierung zu ECEF

Damit wir einen Punkt auf der Erdoberfläche darstellen können benötigen wir neben den Latitude, Longitude und Altitude Werten noch weiter.

**Coordinate Conversion  
Geodetic Latitude, Longitude, and Height to ECEF, X, Y, Z**

$$X = (N + h) \cos \phi \cos \lambda$$

$$Y = (N + h) \cos \phi \sin \lambda$$

$$Z = [N(1 - e^2) + h] \sin \phi$$

where:

$\phi, \lambda, h$  = geodetic latitude, longitude, and height above ellipsoid

$X, Y, Z$  = Earth Centered Earth Fixed Cartesian Coordinates and:

$$N(\phi) = a / \sqrt{1 - e^2 \sin^2 \phi} = \text{radius of curvature in prime vertical}$$

$a$  = semi-major earth axis (ellipsoid equatorial radius)

$b$  = semi-minor earth axis (ellipsoid polar radius)

$$f = \frac{a - b}{a} = \text{flattening}$$

$$e^2 = 2f - f^2 = \text{eccentricity squared}$$

Peter H. Dana 8/3/96

Abbildung 24: Konvertierung zu ECEF [42]

Die Höhenunterschiede auf der Erde erschweren eine klare Definition der Erdform und somit auch die Ortung des aktuellen Standpunkts.

```
public class CoordinateUtilities
{
    static double a_ = 6378137.0;           // WGS-84 Earth semimajor axis (m)
    static double b_ = 6356752.314245;      // Derived Earth semiminor axis (m)
    static double f_ = (a - b) / a;          // Ellipsoid Flatness
    static double e_sq = f * (2 - f);        // Square of Eccentricity

    // Center Koordinates of calculated Area
    private double centerLat;
    private double centerLon;
    private static double centerAlt; // 311 // WGS84 46.87;           // Meters

    //private static double centerLat = 49.448256; // Degrees
    //private static double centerLon = 11.095962; // Degrees

    CoordinateUtilities(double lat, double lon, double alt)
    {
        centerLat = lat;
        centerLon = lon;
        centerAlt = alt;
    }
}
```

Abbildung 25: Berechnung ECEF in Android Studio

Aufgrund dieser Tatsache verwendet man für eine bessere Darstellung das WGS84 Ellipsoid Referenzmodell [43]. Mithilfe der vordefinierten Radien des WGS84 Referenzmodell lässt sich die Ebenheit f („flatness“) des Ellipsoiden berechnen. Die zwei Radien liegen auf der Äquatorebene und werden als semi-major-axis a und semi-minor-axis b bezeichnet. Hierbei entspricht a der großen Halbachse und liegt auf der X-Achse. Der Wert b entspricht der kleinen Halbachse und liegt auf der Y-Achse [44]. Mit dem Wert der Ebenheit f lässt sich nun der Wert zu Exzentrizität  $e^2$ , welche den Abstand vom Brennpunkt zur Ellipsenmitte definiert, ermittelt. Hat man die Exzentrizität  $e^2$ , kann nun auch der Krümmungsradius N bestimmt werden [45]. Die resultierenden Ergebnisse können nun für die Berechnung der ECEF Koordinaten in die dafür entsprechende Formel eingesetzt werden [46].

```
//Convert Lat, Lon, Altitude to Earth-Centered-Earth-Fixed (ECEF)
//Input is a three element: lat, lon and alt (m) // used to be lat, lon in (rads)
//Returned array contains x, y, z in meters
//http://danceswithcode.net/engineeringnotes/geodetic\_to\_ecef/geodetic\_to\_ecef.html
public static double[] geo_to_ecef(double lat, double lon, double alt)
{
    double[] ecef = new double[3]; //Results go here (x, y, z)
    lat = Math.toRadians(lat); // changed here
    lon = Math.toRadians(lon); // changed here
    double n = a / Math.sqrt(1 - e_sq * Math.sin(lat) * Math.sin(lat));
    ecef[0] = (n + alt) * Math.cos(lat) * Math.cos(lon); //ECEF x
    ecef[1] = (n + alt) * Math.cos(lat) * Math.sin(lon); //ECEF y
    ecef[2] = (n * (1 - e_sq) + alt) * Math.sin(lat); //ECEF z
    return (ecef); //Return x, y, z in ECEF
}
```

Abbildung 26: 2.Berechnung ECEF in Android Studio

### East-North-Up (ENU)

Anhand der Abbildung 23 (**3.2.3 Global zu Lokal**) wird das ENU Koordinaten System als grüne Fläche dargestellt. Diese repräsentiert unser späteres Lokales Koordinatensystem.

Hier verläuft die X-Achse Richtung Osten, die Y-Achse verläuft gen Norden und die Z-Achse in Aufwärtsrichtung.

Da uns die meisten Werte aus der Berechnung der ECEF Koordinaten bereits zur Verfügung stehen müssen wir diese nun in die Konvertierungsformel an der entsprechenden Stelle anwenden. Eine Matrixtransformation sorgt dafür dass wir Koordinaten aus einem beliebigen Koordinaten System für unser eigenes verwenden können. Hierfür benötigen wir zunächst den Ursprung unseres Koordinatensystems. Dieser errechnet sich indem wir die aktuelle Position des Spielerobjekts vom dazugehörigen ECEF Wert in Abhängigkeit des WGS84 Ursprungs, subtrahieren. Mit den hieraus resultierenden Ergebnissen kann im Anschluss die Matrixmultiplikation zur Errechnung der ENU Koordinaten, durchgeführt werden.

```
// Converts the Earth-Centered Earth-Fixed (ECEF) coordinates (x, y, z) to
// East-North-Up coordinates in a Local Tangent Plane that is centered at the
// (WGS-84) Geodetic point (lat0, lon0, h0).
// https://gist.github.com/govert/1b373696c9a27ff4c72a
public double[] ecef_to_enu(double x, double y, double z, double lat0, double lon0, double h0)
{
    double lambda = Math.toRadians(lat0);
    double phi = Math.toRadians(lon0);
    double s = Math.sin(lambda);
    double N = a / Math.sqrt(1 - e_sq * s * s);

    double sin_lambda = Math.sin(lambda);
    double cos_lambda = Math.cos(lambda);
    double cos_phi = Math.cos(phi);
    double sin_phi = Math.sin(phi);

    double x0 = (h0 + N) * cos_lambda * cos_phi;
    double y0 = (h0 + N) * cos_lambda * sin_phi;
    double z0 = (h0 + (1 - e_sq) * N) * sin_lambda;

    double xd = x - x0;
    double yd = y - y0;
    double zd = z - z0;

    // This is the matrix multiplication
    double xEast = -sin_phi * xd + cos_phi * yd;
    double yNorth = -cos_phi * sin_lambda * xd - sin_lambda * sin_phi * yd + cos_lambda * zd;
    double zUp = cos_lambda * cos_phi * xd + cos_lambda * sin_phi * yd + sin_lambda * zd;

    double[] enu = { xEast, yNorth, zUp };
    return enu;
}
```

Abbildung 27: ENU Berechnung in Android Studio

Abbildung 28: ENU Formel

$$U = -(X - X_0) \sin \lambda_0 + (Y - Y_0) \cos \lambda_0$$
$$V = -(X - X_0) \sin \varphi_0 \cos \lambda_0 - (Y - Y_0) \sin \varphi_0 \sin \lambda_0 + (Z - Z_0) \cos \varphi_0$$
$$W = (X - X_0) \cos \varphi_0 \cos \lambda_0 + (Y - Y_0) \cos \varphi_0 \sin \lambda_0 + (Z - Z_0) \sin \varphi_0$$

Die ermittelten ENU Koordinaten können nun für die Positionierung des Spielerobjekts zur weiteren Bearbeitung genutzt werden.

```
// Wrapper for both geo2ecef And ecef2enu
// Input Arguments: current position latitude, current position longitude, current position altitude
public double[] geo_to_enu(double lat, double lon, double alt)
{
    double[] ecef = geo_to_ecef(lat, lon, alt);
    double[] enu = ecef_to_enu(ecef[0], ecef[1], ecef[2], centerLat, centerLon, centerAlt);
    //Log.i("ENU in geo2enu func","X = " + enu[0] + ", Y = " + enu[1] + ", Z = " + enu[2]);
    return enu;
}
```

Abbildung 29: Konvertierungsablauf ENU

Damit wir für den gesamten Konvertierungsablauf nur eine Funktion aufrufen müssen, werden die beiden Funktionen - Umwandlung von Latitude, Longitude und Altitude zu ECEF sowie von ECEF zu ENU – als Wrapper zusammengefasst.

### 3.3 LibGDX

Nachdem durch die fehlende Möglichkeit Unitys, die gewünschte GPS Funktionalität nicht integriert werden konnte musste eine Alternative für ein 3D Grafikframework gefunden werden.

Durch bis dato gesammelten Erkenntnisse, was genau für die Verwendung von GPS in einem Java basierten Android Studio Projekt notwendig war, musste ein, IDE integrierbares, 3D Framework gefunden werden welches in nativen Code mit eingebettet werden konnte.

Nach Recherche ergaben sich wenige Treffer, nach deren Tests ein Entscheidung zu Gunsten des Frameworks **libGDX** gefällt werden konnte.

Auch wenn das Framework nach erster Recherche zu 95% für 2D Grafik eingesetzt wird, verfügt es über, für uns ausreichende, Möglichkeiten eine 3D Szene erstellen und Einstellungen darin vornehmen zu können. Auch die Möglichkeit extern erstellte Modelle zu importieren ist nach ersten Recherchen gegeben.

## Setup



Abbildung 30: LibGDX Setup Menü

Über die Download Seite [48] von libGDX wird eine Java Datei heruntergeladen mit welcher ein Projekt für beliebige IDE unter Angabe diverser Pfade erstellt werden kann. Wichtigster Punkt ist die Angabe des Software Developer Kit (SDK) Pfades.

Dieser lässt sich entweder mit Hilfe Android Studios unter: AndroidStudio/Preferences und sich im Pop-Up gewählten Reiter „Android SDK“ einfach herauskopieren.

Alternativ gibt eine kurze Google oder auch Betriebssystem Suche nach dem Standardpfad für die Ablage dieser Auskunft. Nachdem die Einstellungen vorgenommen wurden erstellt die Anwendung am Zielpfad ein neues Android Studio Projekt welches über diese leicht importiert werden kann.

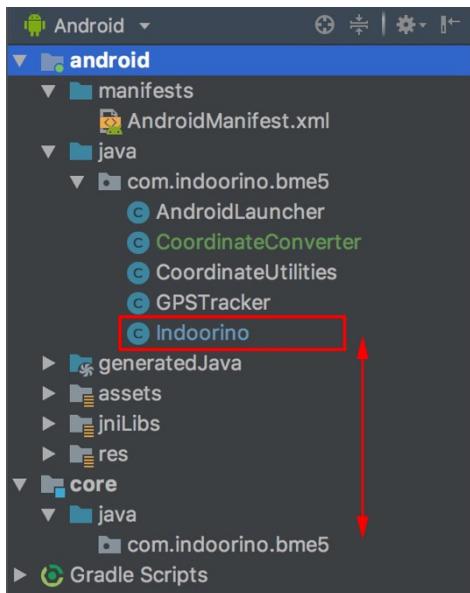


Abbildung 31: LibGDX Struktur  
Android Studio

Nach dem Import vollzieht *Gradle* seine Arbeit und lädt aus dem Internet alle verknüpften libGDX Bibliotheken herunter welche in den *Gradle Scripts* jeweils eingebettet wurden.

In der Android Ansicht werden zwei Hauptordner ersichtlich welche zum einen für die Laufzeit des Applikation (*Android*) als auch für die libGDX Implementierung (*core*) zuständig sind.

Zu Beginn befindet sich die Klasse welche die Grafikszene besitzt noch unter *core*. Hier ist jedoch keinerlei Bezug auf das Betriebssystem möglich. Deswegen wurde die Java-Klasse einfach in den Java Ordner des *android*-Ordners transferiert.

Eine Instanz der Grafikklassse (*Indoorino*) wird zur Initialisierung der Applikation durch die Klasse des *AndroidLauncher* vorgenommen welcher als Einstiegspunkt fungiert.

### 3.3.1 Einbindung LocationManager

Wie in der Dokumentation Androids zur Standortabfrage genauer beschrieben ist es zu heutigem API Level (28) relativ einfach diese durchzuführen in dem zwei Hilfsinstanzen in Form des *LocationManagers* und *LocationListeners* initiiert werden.

Wie für sehr viele Betriebssystemschnittstellen, welche Zugriff auf Sensoren nutzen, ist es notwendig diesen, bei Aufruf, einen Kontext der Applikation als Parameter zu überreichen.

Da die Klasse *Indoorino*, welche von der libGDX Frame Klasse *ApplicationAdapter* erbt, keine eigentliche *Activity* ist, gibt es direkt darin keine Möglichkeit den benötigten Kontext abzugreifen.

Umgangen wird dieses Problem indem bei Instanziierung des Frames *Indoorino*, ein Übergabeparameter der *AndroidLauncher* Klasse mitgegeben wird, welcher als Handle für sämtliche betriebssystembedingte Zugriffe dient. Die Klasse *Indoorino* muss lediglich um einen Konstruktor erweitert werden welcher wiederum einen Parameter erwartet. Der Parameter ist diesem Fall die komplette Applikation selbst um maximalen Zugriff gewährleisten zu können.

```
public Indoorino(AndroidApplication myapp)
    {app = myapp;}
```

---

Konstruktor der Klasse

```
AndroidApplicationConfiguration config =
    new AndroidApplicationConfiguration();
initialize(new Indoorino( myapp: this), config);
```

---

Initialisierung der Klasse mit Übergabe der Applikation

Doch selbst jetzt ist der Abruf der GPS Daten noch nicht möglich da der übergebene Applikationshandle nicht selbst über einen Kontext verfügt. Diverse Forenbeiträge bestätigen die Nichtverfügbarkeit des Kontextes in diesem Thread [48]. Um sich nun etwas tiefer in das Betriebssystem einzudringen wird über den *Handle* auf den User Interface verarbeitenden Thread zu gegriffen. In dieser können alle notwendigen Einstellungen und Aufrufe vorgenommen werden.

```
appl.runOnUiThread() {...}
```

Der *LocationListener* dient durch sein Observer Pattern als Beobachter der GPS Empfangsdaten.

Bei seiner Instanziierung kann entsprechendes Verhalten bei Änderung des Status oder Standortes definiert werden.

Der *LocationManager* dient als Quelle und direkteste Verbindung zum GPS Sensor. An ihm wird der Anfrage nach Standortabfragen angemeldet.

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 1, locationListener);
```

Übergeben wird zum einen die Information, was genau überwacht werden soll, nämlich GPS. Parameter zwei entspricht dem Updateintervall in Millisekunden, gefolgt von der minimalen Standortänderung in Metern. Zuletzt der Beobachter selbst übergeben, da an diesen Informationen im Falle einer Änderung, entsprechend des Rasters, übermittelt werden.

```
locationManager = (LocationManager)
appl.getContext().getSystemService(LOCATION_SERVICE);
locationListener = new LocationListener() {
    @Override public void onLocationChanged(Location location) {
        locationLon = location.getLongitude();
        locationLat = location.getLatitude();
    }
    @Override public void onStatusChanged(String provider, int status, Bundle extras) {};
    @Override public void onProviderEnabled(String provider) {};
    @Override public void onProviderDisabled(String provider) {};
};
```

Wie in oben gezeigtem Code Fragment zu sehen wird innerhalb der Update Funktion bei Standortänderung der Funktionscode überschrieben (`@Override`) und für eigene Zwecke in Form einer Änderung der globalen Variablen für Längen- und Breitengrad genutzt. Nachfolgendes Bild illustriert das Verhalten der API Seitens Android:

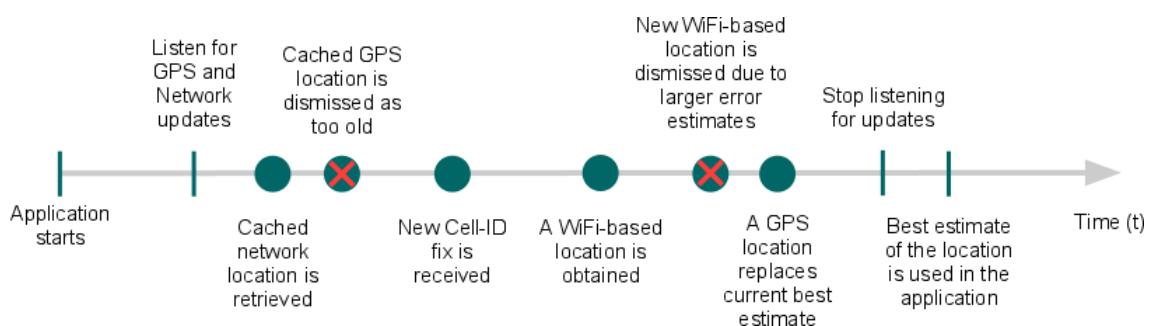


Abbildung 33: Verhalten API über Android [49]

### **3.3.2 Importieren von 3D Modellen**

Da später der eigentliche Nutzer in Form eines Avatars als auch das Spielfeld an sich in der Szene platziert werden soll, müssen die Modelldaten welche in anderen 3D Modellierungsprogrammen erzeugt und exportiert wurden, in die Applikation integriert werden.

Nach Recherchen in den Dokumentationen [50] des libGDX Frameworks wurde an mehreren Punkten auf die Kompatibilität von .obj Dateien verwiesen welche funktionieren aber nicht die erste Wahl dafür darstellen. Empfohlen laut Entwicklerseite sind .g3db Dateien.

Der gängige Export aus diversen 3D Modellierungsprogrammen ist eine Datei des Formates .3DS. Das für das Projekt gewählte Modellierungsprogramm 3D Max verfügt über die Möglichkeit in dieses jedoch nicht in .obj oder .g3db zu exportieren bzw. konvertieren.

Nach Erstellung des Modells im .3DS Format konnte dieses mithilfe von, im Internet frei zugänglichen Tools, umgewandelt werden. Für eine direkte Umwandlung von .3DS zu g3db ist keine Softwarelösung zu finden. Lediglich durch einen Schritt über ein anderes Format ist dies möglich. Für die Konvertierung von 3DS zu .obj wurde der Onlinedienst greentoken [51] verwendet welcher von Alexander Gessler kostenlos zur Verfügung gestellt ist. Für die Umwandlung von .obj in .g3db ist direkt von libGDX die Software *FBX-Converter* [52] erhältlich welche lediglich aus einer Binary Datei besteht.

Da das Binary nicht über ein User-Interface verfügt muss dieses über den Terminal ausgeführt werden. Mit folgendem Befehl lässt sich die .obj Datei umwandeln:

```
./fbx-conv-mac -o g3db ..../CityBlock.obj
```

./ führt das Binary aus, -o ist Befehl um Zielfileformat zu bestimmen mit g3db.  
..../CityBlock.obj = relative Pfadangabe zu Dat

### 3.3.3 Drehung des Testgeländes in Nordrichtung

Für den Import der Testfläche (Areal zwischen Parkhaus und BB Gebäude) musste sich vorher auf einen Referenzpunkt zwischen GPS Lokalisation und Model geeinigt werden. Hierfür wurde das obere Eck des BB-Gebäudes, welches an die Bahnhofstraße angrenzt gewählt. Für spätere Koordinatensystemtransformationen wird auf diesen GPS Referenzpunkt Bezug genommen. Für das Model der Testfläche ist dieses Eck der Nullpunkt, der Ursprung, gesetzt.

Ein Import der Testfläche lässt dies ausgerichtet nach Norden im Koordinatensystem erscheinen. Da die Fläche bzw. die Gebäude jedoch nicht nach Norden ausgerichtet sind muss die Fläche im lokalen Koordinatensystem um den Differenzgrad gen Norden rotiert werden um somit eine genaue grafische Positionierung des Spielerobjektes auf der Karte zu ermöglichen.

Hierzu wurde am vereinbarten Referenzpunkt entlang der Arealseite, Streifen welcher sich Fläche und Frontwand des BB-Gebäudes teilen.

Mit einem Kompass wurde der Differenzgrad an mehreren Stellen der Seite gemessen. Voraussetzung, damit dieses Unterfangen von Nutzen ist, dass sich alle Messpositionen in einer Linie befinden und den gleichen Winkel aufweisen. Da an manchen Stellen keinerlei Parallelität zum Areal befindlich war, wurde der an der obersten Kante (Referenzpunkt) befindliche Winkelgrad genommen. Für die Messung ergab sich die Ausrichtung von ca.  $335^\circ$  NW was einer Rotation von  $25^\circ$  gegen den Uhrzeigersinn entspricht ( $360^\circ - 335^\circ = 25^\circ$ ).

Die Skizze verbildlicht das oben geschilderte Problem.

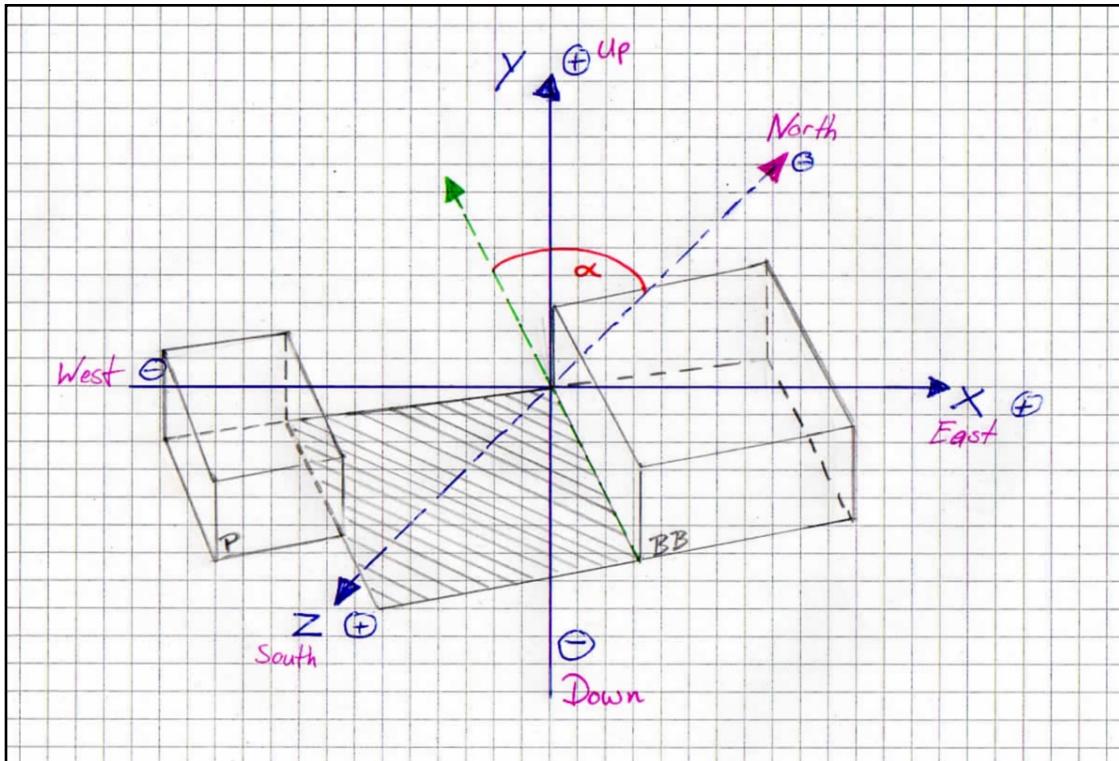


Abbildung 34: libGDX Koordinatensystem mit Achsenbezeichnung

Der Winkel  $\alpha$  (in rot) gibt den notwendigen Rotationswinkel für das Areal, an der Nordachse ausgerichtet, an.

Als alternative Methode könnte in die Rotation an der Koordinate selbst vorgenommen werden.

Auf diese Weise müsste die Fläche nicht gedreht und die Kamera nicht aus Linearitätsgründen versetzt werden.

### 3.3.4 Positionierung des Spielerobjekts

Nachdem die Transformation der GPS Daten in das lokale Koordinatensystem als auch der Import des modellierten Testgeländes erfolgt war musste das Spielerobjekt im Spielfeld richtig positioniert werden. Hierfür kann für die Instanz der Klasse *ModelInstance* folgende Methodik genutzt werden:

Die Positionsänderung wird innerhalb der GPS-Funktion *onLocationChanged()* implementiert, da sie dort sobald sich die GPS

Koordinaten ändern angestoßen wird. Hierfür wird die Methode `set` verwendet welche neben einem dreidimensionalen Vektor noch eine Drehungsinformation in Form von Quaternionen als Übergabeparameter erhält. Das Update der Positionsdaten erfolgt schnellsten falls sekündlich was dem *Live* Spielverhalten bei Drehung nicht zugutekommt.

Die Methode lässt ein setzen ohne Drehungsinformation nicht zu. Um diese dennoch verwenden zu können wird ein temporärer Datenplatzhalter vom Typ der benötigten Quaternionen erzeugt welche die aktuelle Ausrichtung (Drehung) des Spielers durch die Methode `getRotation()` enthält. Diese Variable wird im Anschluss wieder in seinen eigentlichen Extraktionspunkt, die Spielerinstanz, eingespeist. Somit bleibt die Ausrichtung gleich, die Position jedoch ändert sich.

```
Quaternion tmpQuat = new Quaternion();
playerInstance.transform.getRotation(tmpQuat);
playerInstance.transform.set(new Vector3((float)enu[0],1,(float)enu[1]*(-1)), tmpQuat);
```

Die Resultate der Transformation von GPS zu unseren lokalen Koordinaten erhalten wir in Form eines Double-Arrays was der Nachkommastellengenauigkeit geschuldet ist. Hieraus erstellen innerhalb des `set` einen neuen dreidimensionalen Vektor für welchen jeder Wert in einen Float gewandelt werden muss. Der Konstruktor des `Vector3` wird mit den Argumenten für X-, Y-, und Z-Achse in eben dieser Reihenfolge erstellt.

Die Überführung der Koordinaten des Spielers in das libGDX Koordinatensystem erweist sich durch nicht übereinstimmende Achsenwertung als etwas schwieriger.

Wie in Abbildung 34 zu sehen entspricht die X-Achse einwandfrei den Anforderungen. Die Y Achse welche nach Errechnung der Tiefe der Fläche entspricht ist in libGDX jedoch die Höhe/Tiefe. Aufgrund der

Tatsache das sich im Spielfeld keinerlei Erhöhungen oder Täler befinden reicht es die Spielerposition konstant mit einem Wert von 1 zu füttern. Die Y-Achse des Vektors ist im Koordinatensystem durch die Z Achse dargestellt welche jedoch wertinvertiert ist. Hierfür wird der Wert einfacherweise mit -1 multipliziert.

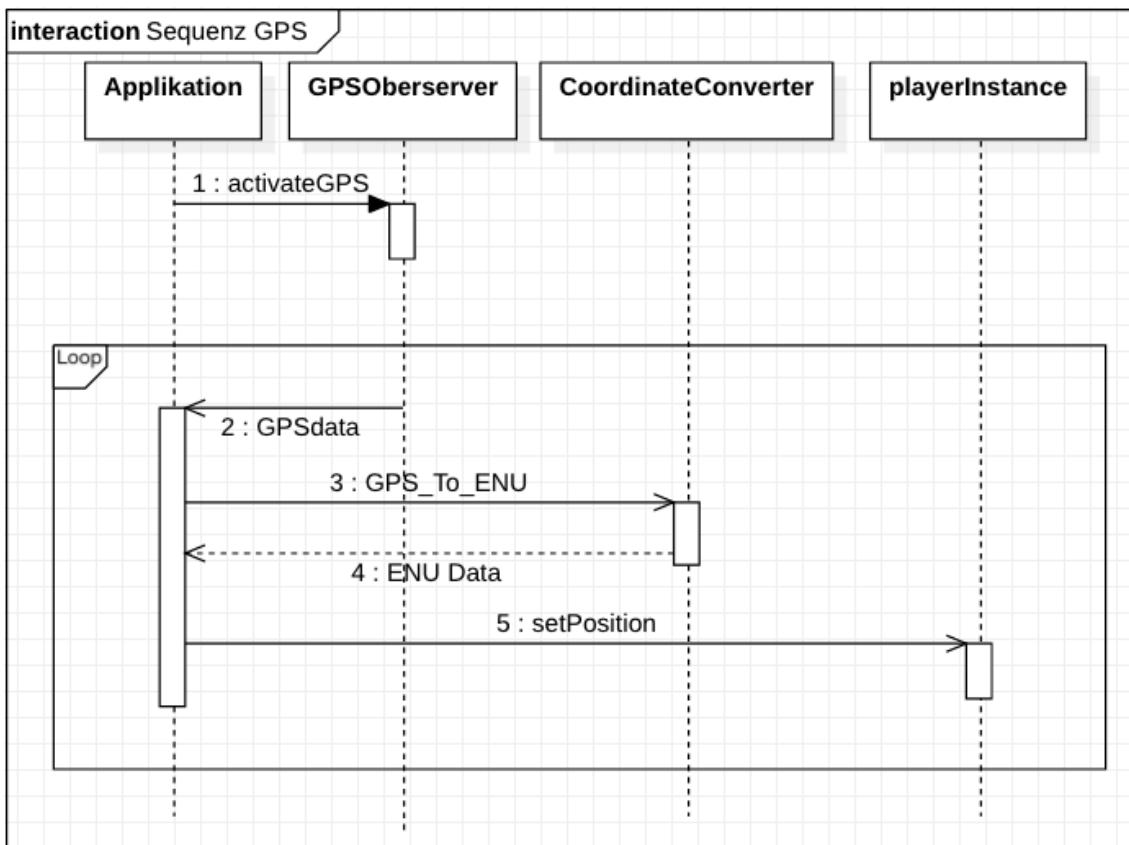


Abbildung 35: Sequenzdiagramm des Prototypen

Für alle UML Sequenzdiagramme wurde das Tool StarUML [53] verwendet, welches als kostenlose Evaluationsversion für Mac zur Verfügung steht.

Das Sequenzdiagramm welches den Kern für den Prototypen darstellt, setzt sich aus dem aktivieren des GPS-Observers, dem stetigen erhalten von GPS Informationen, die Konvertierung dieser in das lokale Koordinatensystem und Platzierung des Spielerobjektes anhand dessen. Es beinhaltet eine Schleife in welchem applikationslaufzeitabhängig GPS Daten empfangen, verarbeitet und für

die Anwendung gezielt, im Sinne der Spielerpositionierung, eingesetzt werden. Die Instanziierung des Spielerobjekts ist bereits erfolgt.

### Spielerobjektrotation auf Karte

Für die Rotation des Spielers wird auf der smartphone-interne Kompasssensor zugegriffen welcher im eigentlichen Sinne ein Magnetometer darstellt. Die Nutzbarkeit der Sensorik und deren Schnittstelle ist nur unter Einbeziehung der zugehörigen Android Bibliotheken möglich.

```
import android.hardware.Sensor;           // import  
import android.hardware.SensorEventListener; // import  
import android.hardware.SensorManager;      // import android.hardware.SensorEvent;
```

Ähnlich der GPS Abfrage wird über den Applikationshandle ein SensorManager des Systems abgerufen welcher für die Applikationslaufzeit über Informationen bereit stellt.

```
sensorManager = (SensorManager) appl.getSystemService(SENSOR_SERVICE);  
compass = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);  
if (compass != null) {  
    sensorManager.registerListener(this, compass,  
SensorManager.SENSOR_DELAY_GAME);  
}
```

Die Rotation wird von jeder Position aus neu errechnet somit ein einfaches setzen der Rotation zu Folge hätte dass sich das Spielerobjekt unaufhörlich um die eigene Achse dreht. Um dies zu vermeiden wurde folgende Funktion implementiert welche vom aktuellen Wert den vorherigen abzieht und somit die Differenz ergibt.

```
public float rotateDiff(float input){  
    float tmp = input - lastRotation;  
    lastRotation = input;  
    return tmp;  
}
```

Für die Implementierung wurde sich an vorher erstelltem Sequenzdiagramm orientiert für welches verschiedene Informationen aus der Android API zusammengetragen werden mussten.

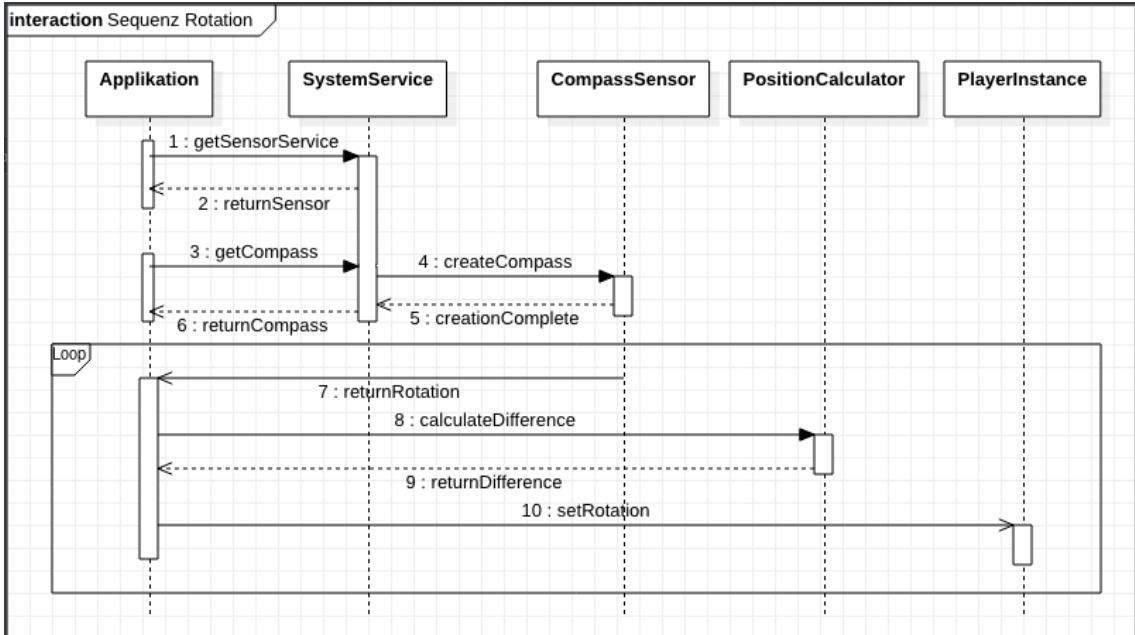


Abbildung 36: Sequenzdiagramm des Prototypen nach Implementierung

Nach einigen Tests der bezogenen Sensordaten stellte sich heraus dass diese relativ unzuverlässig in Genauigkeit waren. Im direkten Vergleich zur bereitgestellten API von iOS (Apple) konnte erhebliche Unterschiede festgestellt werden.

### 3.3.5 Prototyp

#### Testbereich Vermessung

Um den Prototypen auf Funktionalität zu testen wurde der Platz zwischen BB Gebäude und Parkhaus als Testfeld gewählt. Zugleich entspricht dieser Platz als schwierigste zu handhabende Situation aus, da der notwendige Satellitensignalempfang für GPS durch die hohen Gebäude abgeschirmt werden kann. Um die Fläche im gleichen Maß/Verhältnis später in der Applikation verwenden zu können war ein genaues ausmessen dieser notwendig.



Abb.37

Für die Abstandsmessungen zwischen den Gebäuden wurde durch die Hochschule das „*Bosch PLR-50*“ Laser Entfernungsmessgerät bereit gestellt.

Dessen Handhabe war durch platzieren der Unterkante (Oberkante auf Wunsch möglich) an ein Ende und anvisieren der Laserleuchte auf das Ende der Messdistanz besonders einfach. Die gemessene Entfernung ist in der Einheit Meter erfolgt.

Um alle Werte notieren zu können wurde vorab per Hand der große Grundriss abgezeichnet und gemessene Entfernungen mit Pfeilen und zugehörigen Distanzen in Meterangaben erweitert. Jede Messung wurde mehrfach durchgeführt und ein Durchschnittlicher Wert gebildet.

Für die nachträgliche Aufbereitung dieser wurde *GoogleMaps* zu Hilfe genommen, in welchem eine 3D schräg Aufsicht auf das gewünschte Gebiet ermöglicht wurde. Vorher erstellte Fotos der Fläche aus dem obersten Stockwerk des BB-Gebäudes waren aufgrund des fehlenden Weitwinkels und der mangelnden Perspektive nicht dafür geeignet um als Grundlage zu dienen.

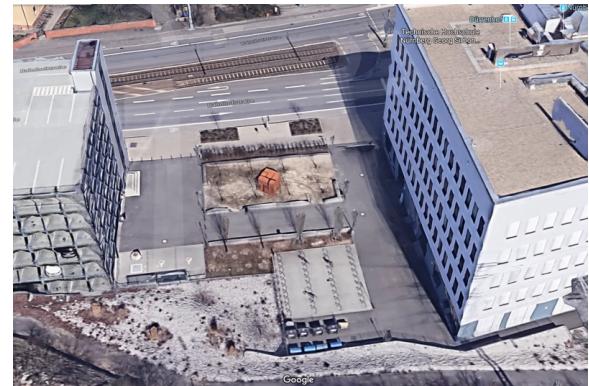


Abbildung 38 & 39: Testgelände BB Gebäude

Für die Übersichtlichkeit war es essenziell den gesamten Bereich möglichst simpel darstellen zu können um die nachfolgenden Arbeiten der Flächenmodellierung entgegen kommen zu können. Insgesamt wurden 16 Distanzen für die Reproduzierbarkeit der Karte gemessen.

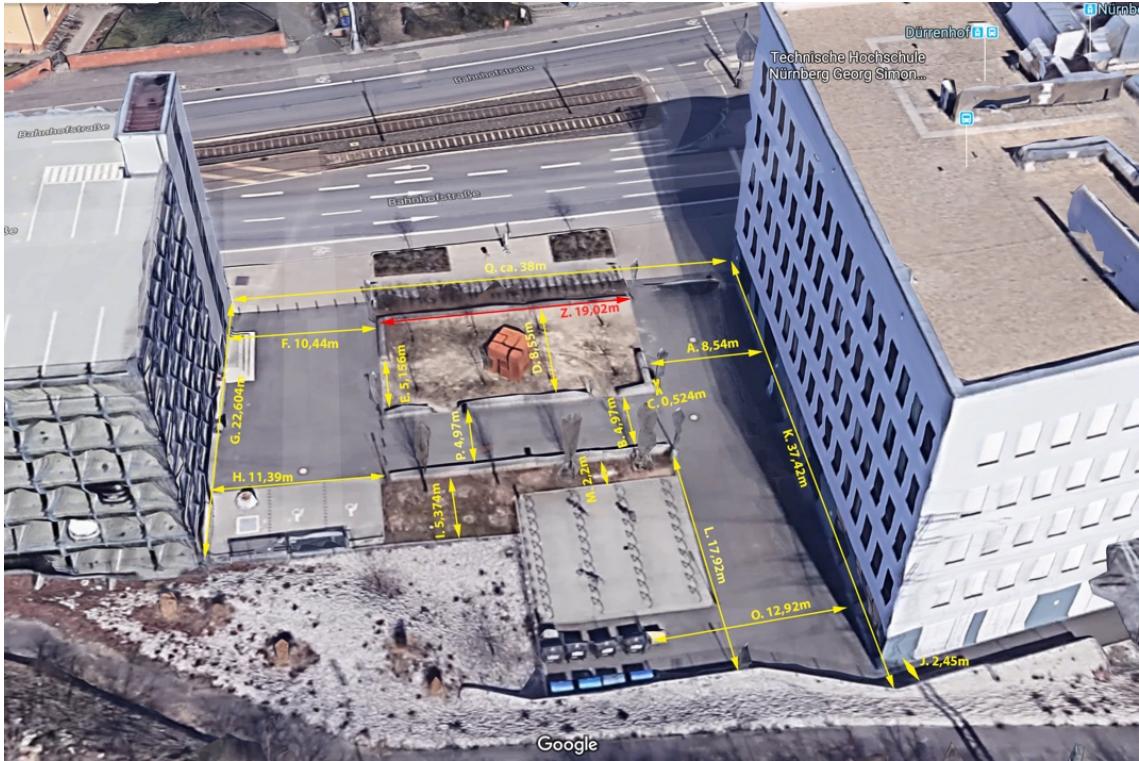


Abbildung 40: Ausgemessen Testgelände BB Gebäude

Zum aktuellen Stand ist der in Android Studio, unter Verwendung des libGDX Frameworks, erstellte Prototyp im Stande, die GPS Koordinaten zu empfangen, diese in ein lokales Koordinatensystem zu überführen und das Spielerobjekt dementsprechend auf dem 3D-modellierten, importierten Testgelände zu platzieren. Auch die Spielerobjektausrichtung durch auslesen des internen Magnetometers funktioniert bereits.

Für die Zusammenarbeit zwischen den Komponenten und deren eigentlichen Interaktionsablauf wurde sich bei der Implementierung an vorher erstellten Sequenzdiagrammen orientiert.

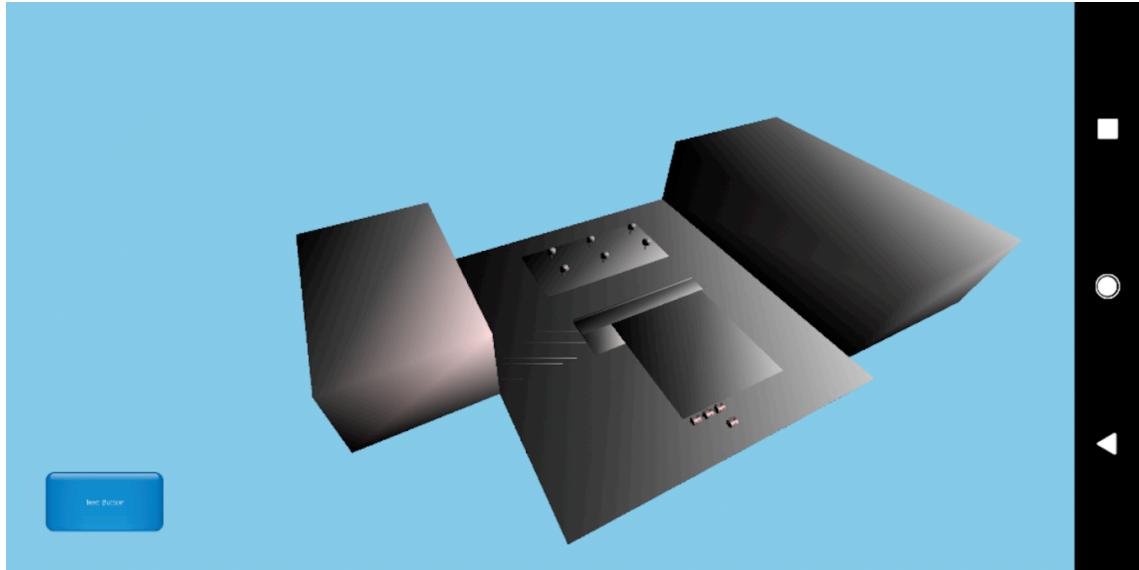


Abbildung 41: Prototyp Testgelände BB Gebäude

Im Vergleich dazu die Oberfläche des in Unity erstellten Prototypen

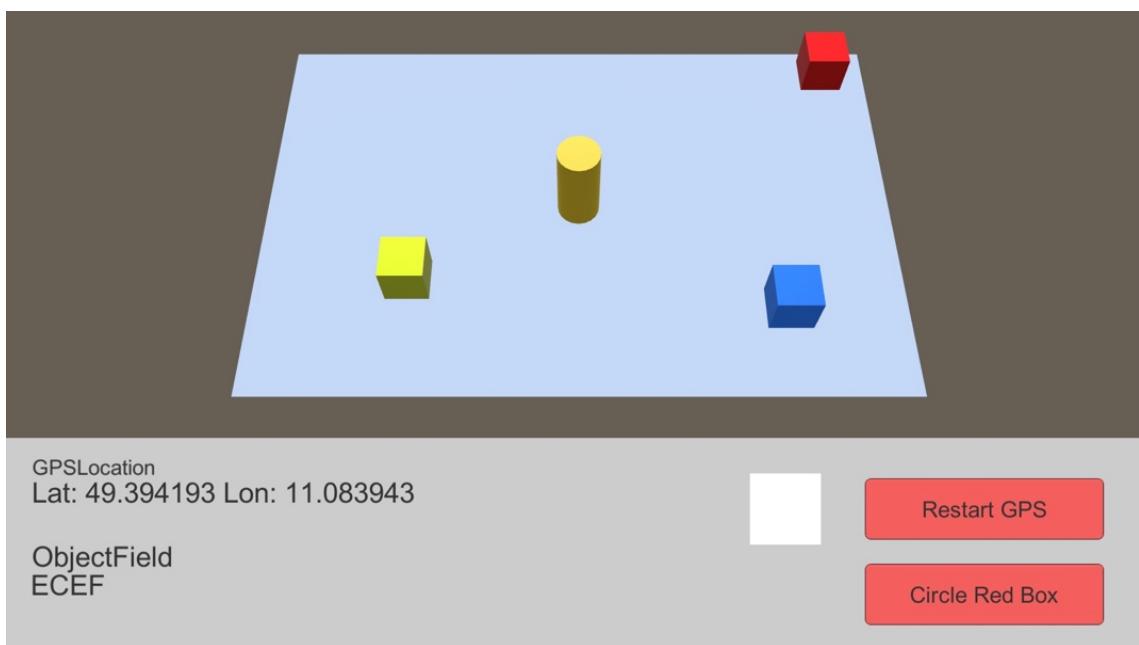


Abbildung 42: Unity Prototyp

## 3.4 Design

### 3.4.1 Erstellung 3D Modelle in 3DsMax

Um die Funktionalität der App testen zu können muss das als Testfläche festgelegtes Gelände in 3D nachmodelliert werden.

#### 3DsMax vs. Blender

An dieser Stelle ist zu entscheiden, welches Programm für die Umsetzung des Modells besonders geeignet ist. Anbei sind die Einsatzbereiche der zwei gängigen Programme in diesem Feld aufgelistet:

Overview of Autodesk 3ds Max Features	
<ul style="list-style-type: none"><li>✓ 3D Animation and Dynamics</li><li>✓ General Animation Tools</li><li>✓ Character Animation and Rigging Tools</li><li>✓ Motion Paths</li><li>✓ Particle Flow Effects</li><li>✓ 3ds Max Fluids</li><li>✓ Max Creation Graph Controllers</li><li>✓ Simple Simulation Data Import</li><li>✓ Geodesic Voxel and Heatmap Skinning</li><li>✓ 3D Rendering</li><li>✓ Improved ActiveShade Rendering</li><li>✓ Physical Camera</li><li>✓ Arnold for 3ds Max</li><li>✓ Rendering in A360</li><li>✓ Autodesk Raytracer Renderer (ART)</li><li>✓ UI, Workflow, and Pipeline</li></ul>	<ul style="list-style-type: none"><li>✓ Scene Converter</li><li>✓ Asset Library</li><li>✓ Smart Asset Packaging</li><li>✓ Customizable Workspaces</li><li>✓ Improved pipeline Tools Integration</li><li>✓ Live Link with Stingray Game Engine</li><li>✓ High DPI Display Support</li><li>✓ MAX to LMV</li><li>✓ 3D Modeling, Texturing, and Effects</li><li>✓ Spline Workflows</li><li>✓ Mesh and Surface Modeling</li><li>✓ Texture Assignment and Editing</li><li>✓ Data Channel Modifier</li><li>✓ Hair and Fur Modifier</li><li>✓ Blended Box Map</li></ul>

Abbildung 43: 3Ds Max Overview Features [54]

#### *3DsMax*

Anwendung überwiegend in:

- Architektur
- Objektdesign
- Produktdesign

- Engineering (z.B detailliertes Auto Design)
- statische Objekte
- einfache Animationen
  - + Oberfläche: intuitiv
  - nur als Studentenversion kostenlos verfügbar

## *Blender*

The screenshot shows a dark-themed web page titled "Overview of Blender Features". On the left is a gear icon. The main content area lists various features with checkmarks:

<ul style="list-style-type: none"> <li>✓ Rendering</li> <li>✓ High-End Production Path Tracer</li> <li>✓ GPU Rendering</li> <li>✓ Game Creation</li> <li>✓ Animation Toolset</li> <li>✓ Fast Rigging</li> <li>✓ Visual Effects</li> <li>✓ Camera and Object Motion Tracking</li> </ul>	<ul style="list-style-type: none"> <li>✓ Masking</li> <li>✓ Compositing</li> <li>✓ Phyton Scripts</li> <li>✓ Video Editing</li> <li>✓ Simulation</li> <li>✓ Modeling</li> <li>✓ Customizable UI</li> <li>✓ Integrate with Pipeline Tools</li> </ul>
--	---

Abbildung 44: Blender Overview Features [55]

Anwendung überwiegend in:

- Objektdesign • Produktdesign • Animationen
- Architektur
- VFX

- Oberfläche: etwas unstrukturiert  
 + kostenlos  
 -/+ Blender ist ein „Allrounder“ was je nach Einsatzbereich Vor- und Nachteile haben kann.

3DsMax und Blender eignen sich ebenfalls für Visual Effects und Animationen. Es gibt jedoch bessere Programme die darauf spezialisiert sind. Das sind z.B Maya oder Houdini. Diese Programme sind darauf spezialisiert vor allem organische Animationen realistisch umzusetzen. Für unser Projekt ist es nicht relevant.

An dieser Stelle ist zu erwähnen, dass beide Programme, Blender und 3DsMax sich für die Umsetzung von der Testoberfläche eignen. Die Vor- und Nachteile spielen erst bei größeren Projekten die Wert auf Animationen, Rigging oder Visual Effects legen, eine Rolle.

#### *Einarbeitung in Blender und 3DsMax / Festlegung*

Als Freeware bietet hiermit Blender den größten Vorteil. Mich persönlich hat dabei die Einarbeitungszeit sehr lange gekostet. Ich empfinde die Oberfläche etwas unstrukturiert und mir haben auch mehrere Ansichten auf ein mal gefehlt. Es gibt dafür reichlich Tutorials.

Nach einiger Zeit habe ich zu 3DsMax gewechselt. Dieses Programm ist für Studenten kostenlos. Ich empfand die Oberfläche sofort als intuitiv. Die vier Ansichten sind jederzeit gleichzeitig sichtbar was den Überblick enorm erweitert und sehr viel Zeit beim hin und her switchen spart. Es gibt deutlich mehr qualitativ hochwertiger Tutorials für 3DsMax, die sehr kurz aber sachlich auf ein spezielles Tool eingehen.

Es ist jedoch jedem Entwickler selbst überlassen das passende Programm zu wählen. Blender ist eins der beliebtesten, kostenlos verfügbaren Programme.

#### *Wichtige Parameter in 3DsMax*

Bevor man mit der Modellierung der Testfläche beginnt, müssen die entsprechender Parameter im Programm eingestellt werden. Da es sich um eine Fläche aus dem realen Leben handelt und das Objekt später auch mit GPS Daten arbeiten wird, müssen die Entfernungsangaben dem selben Maß entsprechen. In diesem Fall sind das Meter. Diese Einstellung ist sehr wichtig und die lässt sich über das Menü mit wenigen Klicks festlegen.

## Units Set UP:

Customise → Units Setup → wähle Meter

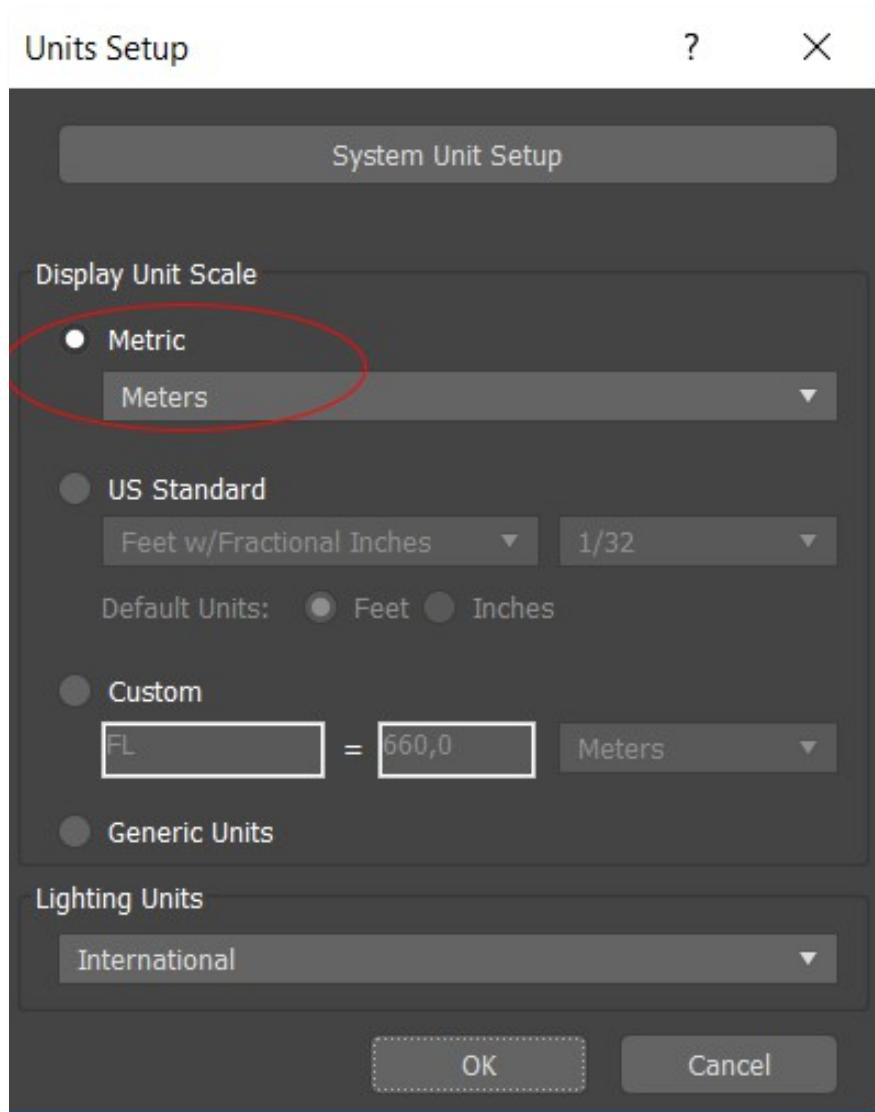


Abbildung 45: 3Ds Max Meter Setup

Will man aus 3DsMax das Objekt später in ein anderes Programm importieren, müssen auch die Units (die gedachten Kästchen die in allen 3D Programmen zur Orientierung dienen) ebenfalls auf Meter umgestellt werden. Andernfalls stimmt das Verhältnis bei der Übertragung nicht mehr überein.

Customize → Units Setup → System Unit Setup → System Unit Scale  
→ Meter

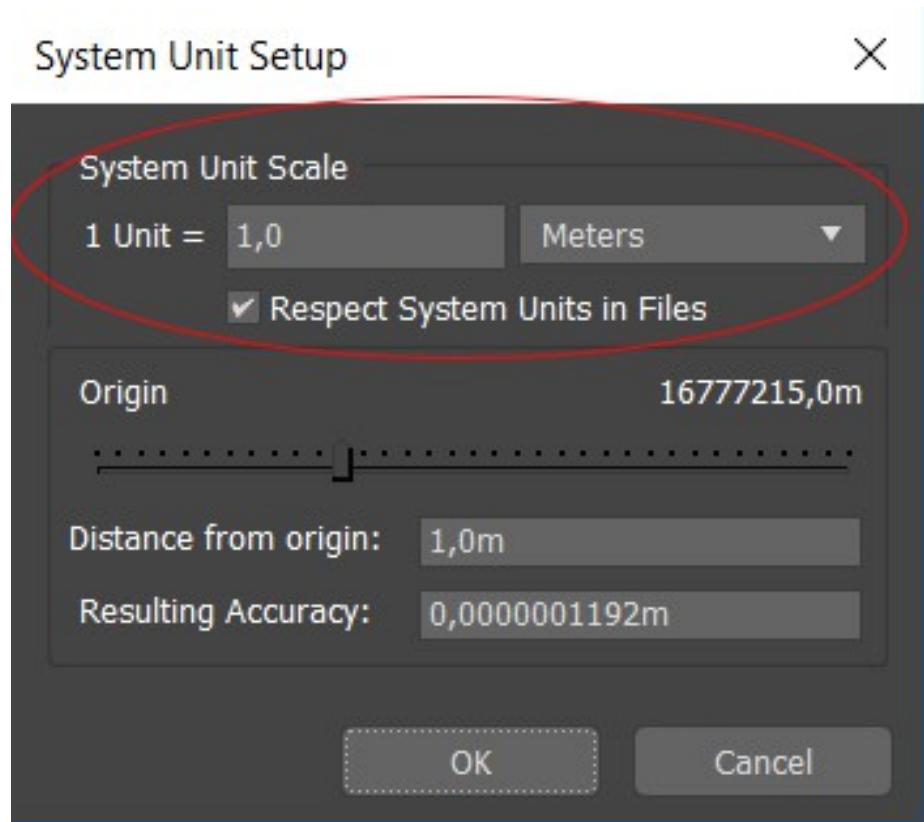


Abbildung 46: 3Ds Max System Unit Scale

Diese Einstellungen lassen sich ebenfalls nachträglich ändern.

#### Umsetzung

Anhand der Messdaten der Testfläche wurde ein einfaches, abstraktes Modell der Testfläche konstruiert. Hierbei lässt es sich mit „Primitives“ = einfache Formen wie Würfel, Kugeln und Rechtecke arbeiten. Komplexere Teile wie Bäume die zur Orientierung beitragen sollen, finden sich in einer der Freeeware 3D Objekt Bibliotheken.

(z.B auf <https://free3d.com/3d-models/3ds-max>)

Wie bereits erwähnt ist eins der Vorteile von 3DsMax gegenüber von Blender die vier Ansichten: rechts, links, oben und 3D die gleichzeitig angezeigt werden. Somit hat man jederzeit Überblick über alle drei Achsen auf denen das Objekt liegt.

Das so oft versehentliche „Abrutschen“ auf eins der Achsen wird somit verhindert.

Abbildung 47: 3Ds Max Übersicht

Zur Übersicht kann man den Objekten unterschiedliche Farben geben, nicht zu verwechseln mit den Maps. Maps = Strukturen, Bilder, Texturen die man auf die Objekte legen kann um diese realistisch aussehen zu lassen. Maps lassen sich in- und exportieren. Farben sind hierbei leider nur im Programm selbst sichtbar. An der Testoberfläche wurde erst mal ohne Maps gearbeitet.

### Kontrolle

Abbildung 48: 3Ds Max Übersicht Kontrolle

70

Hat man ein grobes, erstes 3D Modell erstellt, lassen sich die exakten Maße über das Menü kontrollieren und/oder neu eintragen.

### Export

Am Ende wird die Datei als .obj = Object exportiert und ist somit in andere 3DProgramme, in unserem Fall LibGDX importierbar.

#### **3.4.2 Charakterdesign**

Für die Innenraumnavigation bestand die Idee, dass ein virtueller Avatar die Person durch das Museums Gelände navigiert.

Die erste Idee war das Humpty -Dumpty -Männchen des bereits existierenden Projekts (Vorgängerprojekt) zu übernehmen und in 3D zu realisieren. Jedoch entschied sich die Gruppe nach einiger Zeit um, lieber etwas Neues und Eigenes zu entwickeln.

Da die Ursprungsidee des Museums von einer Geschichte der fiktiven Arbeiterfamilie handelt die auf dem Gelände arbeitet und wohnt, war als Protagonist entweder eins oder beide Kinder der Familie möglich oder deren mögliches Haustier, z.B. ein Hund.

Die Idee mit dem Hund kam in die nähere Auswahl. Damit begann die Recherche welche Hunderasse dafür in Fragen kommt. Da das Museum auf das 20 Jahrhundert fokussiert ist, das Wohnhaus der Familie speziell auf die 50 Jahre, kommen die zur entsprechender Zeit populären Hunderassen in Frage. Das war der Pudel, der Collie oder die Französische Bulldogge [56]. Nach einigen Entwürfen würde der Pudel in die nähere Auswahl genommen, da es sich verglichen zu den anderen beiden Hunderassen viel einfacher in einem 3D Programm umsetzen lässt, und man kein Fell oder sehr komplizierte Züge animieren muss, was die Umsetzung enorm vereinfacht.

## Erste Skizzen:

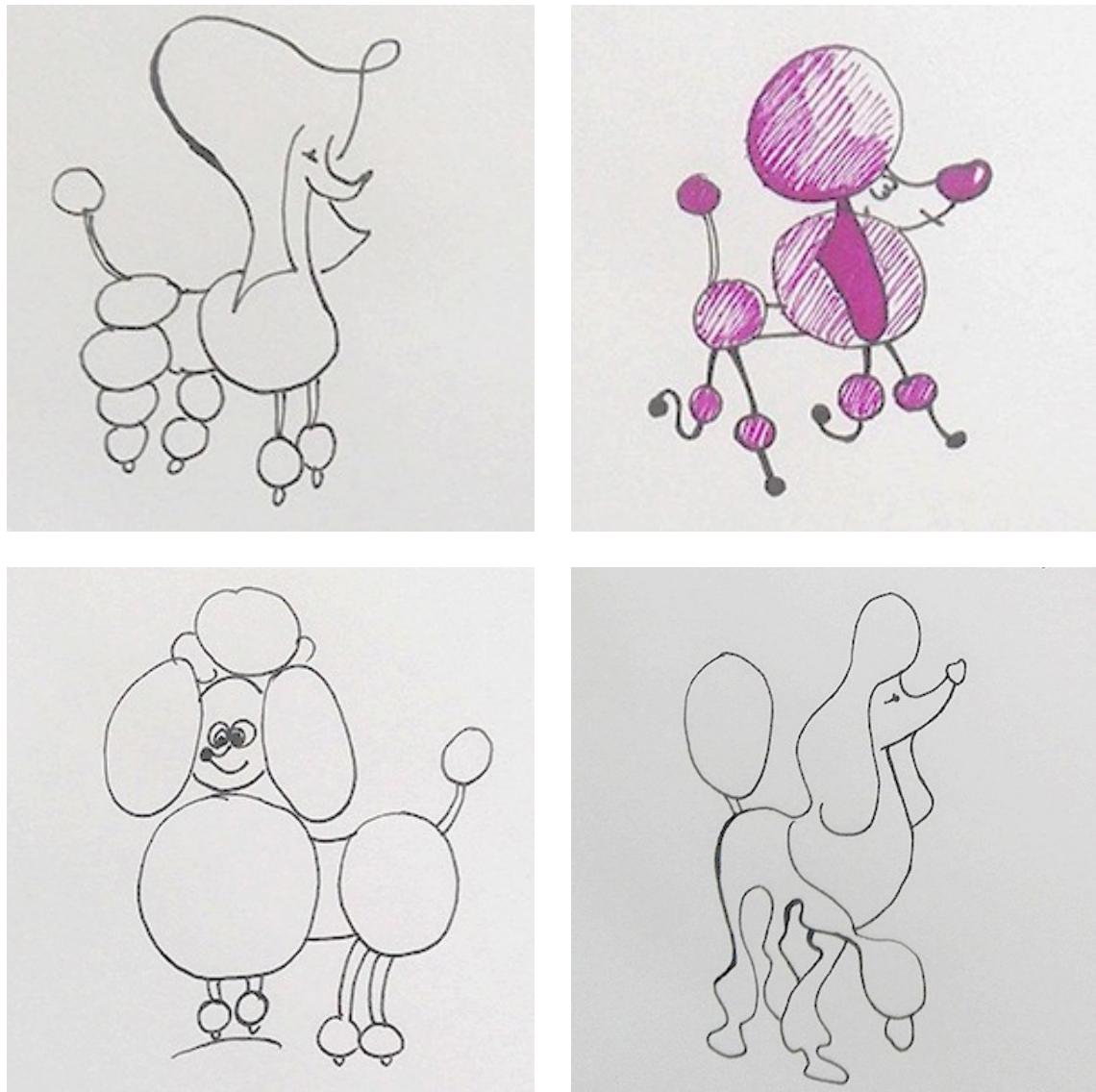


Abbildung 49: Erste Skizzen Charakterdesign

Dadurch das das Projekt einige Anlaufschwierigkeiten mit der Umsetzung hatte wurde die Charakter Design Entwicklung erst mal verschoben.

Somit ging der Fokus der 3D Umsetzung erst mal auf das Testgelände, für die Alternativmöglichkeit mit GPS, welches zunächst nachgebaut werden musste.

Im Laufe dieser Umsetzung und damit anfallender Schwierigkeiten, wurde letztendlich das Gesamtkonzept verworfen. Was dazu führt, dass für das neue Projekt ein neuer Avatar im kommenden Semester entwickelt und umgesetzt werden muss.

Das neue Produkt ist auf eine GPS Navigation im Zoo ausgelegt. Dafür kommen sämtliche Tiere in Frage. Tiere mit besonders guter Orientierung oder sehr ansprechende Tiere = beliebte Tiere, die als Maskottchen dienen können.

Die Recherche zu besonders gut orientierenden Tieren hat ergeben, dass es einige Tiere gibt die sich auf unterschiedliche weise orientieren können.

„Tauben, Fische, Eulen oder Ameisen sind dank ungewöhnlicher Sinnesleistungen wahre Orientierungsexperten“[57]

Dazu zählen Kiefernhäher – eine Vogelart die eine Art fotografisches Gedächtnis besitzt, Tauben – die sich am Magnetfeld der Erde orientieren, Wüstenameisen – die sich anhand der Polarisationsmuster der Sonne orientieren und Schleiereulen die sich am Schall zurechtfinden.

Diese Hintergründe sind sehr spannend, und geben dem möglichen Maskottchen/Avatar einen tieferen Sinn. Denn oft werden einfache Logos und Avatare hinterfragt, es ist eine gute Option für eine Geschichte dahinter und um noch mehr über die Tiere zu lernen.

## Erste Skizzen:

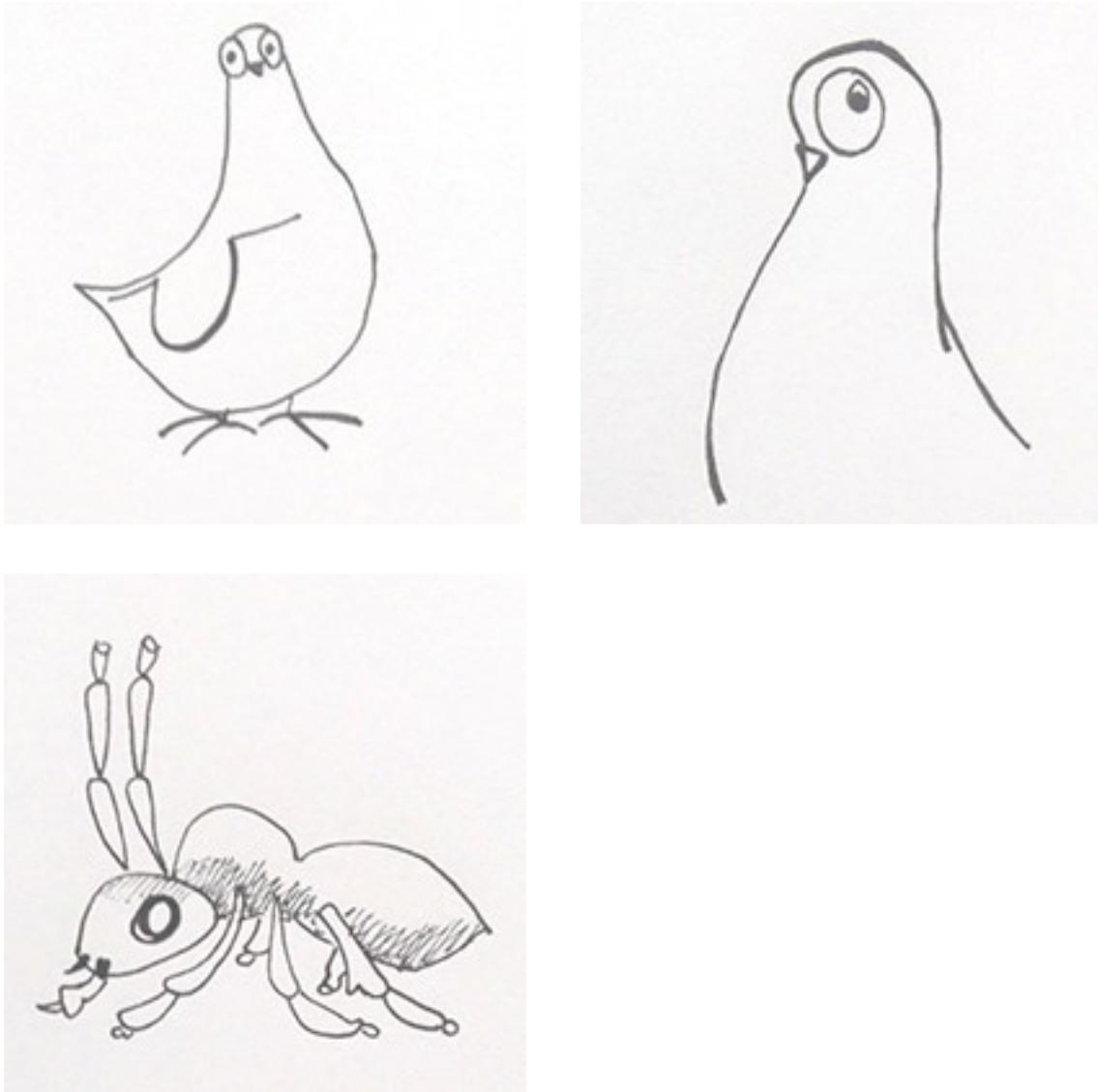


Abbildung 50: Erste Skizzen spezifisches Charakterdesign

Eine weitere Idee für das Avatar ist einen Nashorn, - Rhinoceros zu nehmen. Dadurch das der Endklang des neuen Produkts „Zoorino“ - Nach einem Rhinozeros klingt. Diese Idee ist schlüssig und bietet Bezug auf den Namen der App. Als sehr passend für das Gesamtkonzept empfunden, um Wiederholungen im Namen, Logo und Avatar zu haben, hat diese Idee es auch in die nähere Vorauswahl geschafft.

Noch gibt es aber jedoch keine konkrete Festlegung auf den Charakter des Avatars. Im kommenden Semester wird an der Umsetzung weiter gearbeitet.

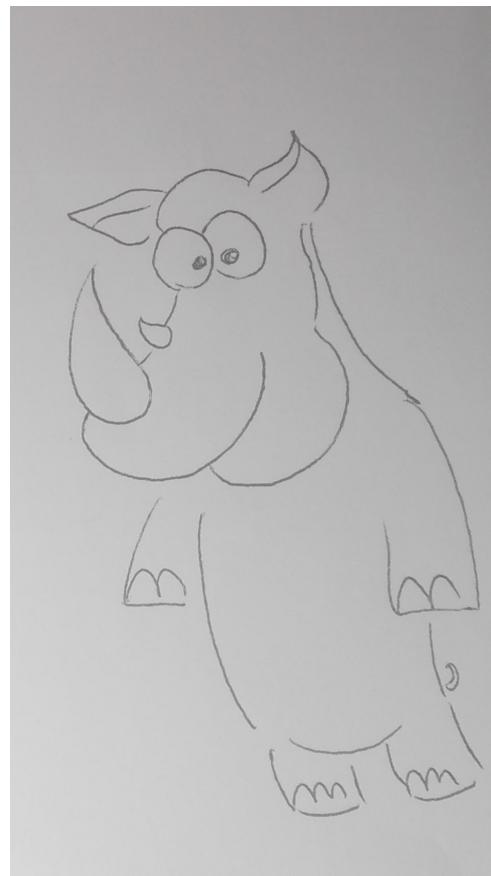


Abbildung 51: Finale Skizze spezifischer Charakter

### **3.4.3 Erstellung des App Menüs in Unity**

Für die Erstellung des Menüs der Indoorino-App wurde sich an den Informationen auf der Unity Webseite orientiert [58].

Mithilfe des Unity UI-Systems ist es möglich, eine „UI Canvas“ in Unity zu erstellen. Die Canvas ist die Grundlage des Menüs, auf ihr werden alle weiteren UI-Elemente wie die Buttons, Images, Texte, Events und Ähnliches platziert.

Außerdem war das „Unity Samples UI“ Package aus dem Asset Store essenziell.

„Der Unity Asset Store ist eine wachsende Bibliothek von Assets“. Sowohl „Unity Technologies“ [59] als auch die „Mitglieder der Community“<sup>15</sup> sind dazu befähigt, die besagten Assets zu generieren und online im Store den Nutzern zur Verfügung zu stellen. Assets reichen von „Texturen, Animationen und Modellen [...] bis hin zu vollständigen Projektbeispielen, Tutorials und Editor-Erweiterungen“. Dabei gibt es die Möglichkeit, zwischen freien und kostenpflichtigen Assets zu wählen. Diese können anschließend unmittelbar in das eigene Unity-Projekt heruntergeladen werden. Außerdem kann der Asset Store direkt über die Unity Game Engine aufgerufen werden oder auch klassisch über den Browser aufrufbar.

Das Unity-Asset an sich ist ein Objekt, das für ein in Unity erstelltes Spiel oder Projekt Anwendung findet.

Die einzelnen Menü-Buttons können gehighlightet werden, das bedeutet, sobald man den entsprechenden Button antippt bzw. mit der Maus darüberfährt, kommt es zu einem Farbwechsel.

Durch den Start-Button wird eine neue Unity-Szene geladen, in die der fertige Navigations-Code eingebettet wird.

In dem Punkt „Funktionalitäten der App“ wurde bereits beschrieben, wie das Menü in den einzelnen Punkten aufgebaut ist. In diesem Schritt wird die Vorgehensweise der Erstellung näher erläutert.

Zuerst wird eine neue Unity-Instanz kreiert. In diesem Fall ist das die Datei mit dem Namen „NewMainMenu“. An dieser Stelle hat der Entwickler die Möglichkeit, zwischen den beiden Modi 2D und 3D zu wählen.

Über die Game Engine wurde im Asset Store das „Unity Samples UI“-Package gedownloadet und direkt in das neu erstellte Unity-Projekt importiert.

An dieser Stelle ist es sinnvoll, zum Zwecke der Übersichtlichkeit einen neuen Ordner im Projekt zu erstellen, der speziell die Objekte aus dem Asset-Package enthält.

Nun liegt dem Entwickler die leere Szene vor. Es ist empfehlenswert, das „2 by 3“ -Layout einzustellen, damit die Szene und die Game-Ansicht zeitgleich in der Game Engine sichtbar sind. Auf der rechten Seite des Bildschirms befindet sich das Default-Layout, bei welchem man die Werte für die erstellten Objekte definieren kann.

Um den Hintergrund für das Menü aufzusetzen, wurde nach einem für das Industriemuseum passenden Foto gesucht. Auf einer Webseite, welche lizenfreie Bilder anbietet, wurde ein entsprechendes Foto ausgewählt und lokal in dem Unity-Ordner abgespeichert. Über das Programm selbst kann auf den eben genannten Ordner zugegriffen werden, um das Industrie-Foto zu importieren. Nachträglich wird die Formatierung der Koordinatenwerte entsprechend angepasst und auch farbliche Änderungen werden vorgenommen. Die Game-Ansicht wird auf 16:9 eingestellt.

Auch das kleine Detail, die Standard-Hauptkamera zu deaktivieren, darf nicht übersehen werden. Ansonsten würde im Playmode, also beim Abspielen der Szene, die Fehlermeldung erscheinen, dass zwei Kameras existieren, da im Ordner der „SF Scene Elements“ eine weitere Kamera vorhanden ist. Dies ist der übergeordnete Ordner, der alle Objekte der weiteren Schritte enthalten wird.

Der Basic-Hintergrund liegt nun vor, anschließend wird das User Interface erstellt. Unter dem Pfad GameObject -> UI können die verschiedenen UI-Komponenten ausgewählt werden.

Zuerst wird die Canvas hinzugefügt. Diese repräsentiert die Grundlage für alle weiteren UI-Elemente, die als Children angesehen werden. Hierzu gehören beispielsweise Buttons, Panels und Sliders. Wenn die Canvas nicht erstellt wurde und dennoch ein Children-Element hinzugefügt wird, generiert sich die Canvas automatisch.

Auch ein EventSystem-Gameobjekt wird von Unity erstellt. Dieses enthält die Komponenten „Event System“ und „Standalone Input Module“, welche dazu benötigt werden, das User Interface interaktiv zu gestalten. Damit wird unter anderem der Klick auf die Buttons oder das Verschieben der Sliders, also der Schieberegler ermöglicht. Mit den Sliders kann im fertigen Produkt beispielsweise die Lautstärke angepasst werden. Auch die Funktion, per Tastatur zu navigieren, benötigt das Vorhandensein des EventSystems in der Szene.

Im nächsten Schritt wird ein Panel hinzugefügt, dass den kompletten Rahmen ausfüllt. Die Fixpunkte des Panels müssen so eingestellt werden, dass lediglich die Mitte der Canvas mit dem Panel bedeckt wird. Dafür werden im Inspector-Feld die Parameter der Fixpunkte entsprechend eingestellt. Das Panel hat eine Bildkomponente beigefügt, dies soll als Grundlage für die Buttons des Menüs dienen. Nun wird das Sprite, also der Hintergrund gesetzt und über das Panel gelegt. Hier kommt ein Element aus dem zu Beginn importierten „Unity Sample UI Package“ zum Einsatz. In diesem Packet stehen einige Rahmen für das Menü zur Auswahl, auf den später die UI-Children positioniert werden.

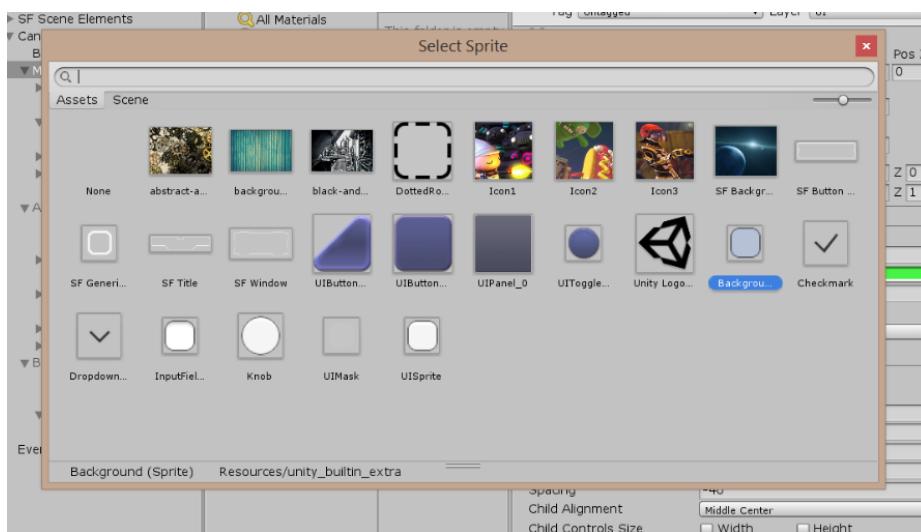


Abbildung 52: Die im Asset-Package enthaltenen Rahmen

Es besteht auch die Möglichkeit, selbst ein Sprite zu erstellen, falls die vorgefertigten Rahmen aus dem Package nicht den Erwartungen des Entwicklers entsprechen.

Jetzt existiert das „MainMenuPanel“ in Form des gewünschten Rahmens. Es ist die unterste Schicht, auf die alle weiteren Buttons gelegt werden.

Das Panel kann farbig gesetzt werden.

Das Objekt MainMenuPanel muss ausgewählt sein, um über den wie oben beschriebenen Pfad einen UI Button hinzuzufügen. Dieser muss entsprechend als Child des MainMenuPanels untergeordnet werden.

Was die Buttons betrifft, kann ebenfalls ein vorgefertigtes Image-Design aus dem Asset-Package ausgewählt sowie die Farbe des Buttons geändert werden. Hier gibt es die Möglichkeit, den jeweiligen Status des Buttons farblich darzustellen. Dadurch kann entsprechend gekennzeichnet werden, ob der User der App den Button im momentanen Zustand berührt oder daraufklickt.



Abbildung 53: Der Button im Normalzustand

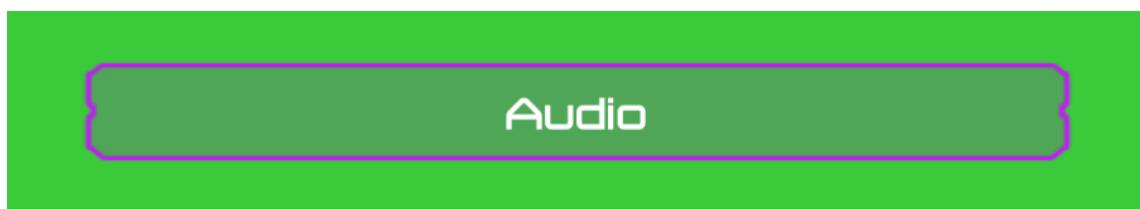


Abbildung 54: Der Button hervorgehoben bei Berührung



Abbildung 55: Der Button im veränderten Farbzustand beim Klicken

Im Inspector-Feld kann der Text der Buttons geändert sowie die einzelnen Attribute wie Schriftgröße, Farbigkeit, Schriftart und vieles Weitere eingestellt werden.

Um Veränderungen vorzunehmen, muss in der Hierarchie des Projekts das entsprechende Element ausgewählt sein. Wichtig ist hierbei, den horizontalen sowie vertikalen Überlauf im Inspector auf „Overflow“ zu setzen, damit der Text über den Rand des Buttons hinausläuft, falls dieser zu klein eingestellt ist. Andernfalls könnte die Situation eintreten, wenn das Buttonfeld zu klein ist und die Werte nicht auf „Overflow“ gesetzt sind, dass die Schriftgröße folglich zu groß für den Button ist und der Text komplett verschwindet. Folglich wäre das Buttonfeld leer. In Unity ist die praktische „Best Fit“-Option in Bezug auf den Text bereits implementiert, damit sich die Schriftgröße automatisch an die jeweiligen Felder anpasst.

Um das Layout der Buttons einzustellen, bietet sich die „Vertical Layout Group“ an, welche als neue Komponente beim MainMenuItem eingestellt werden kann. Die „Vertical Layout Group“ ordnet automatisch die Werte für eine Gruppe von UI-Elementen an, damit der Entwickler nicht bei jedem Element einzeln die Parameter einzustellen hat. Als Nächstes kann lediglich der eine, bereits bestehende Button dupliziert werden, um die Anzahl der für das App-Menü benötigten Buttons zu erhalten. Nun kann ein Padding-Wert beim MainMenuItem eingestellt werden, damit sich die Buttons an die Canvas anpassen. Die generierten Buttons wurden an dieser Stelle mit den Bezeichnungen „Navigation Starten“, „Audio“ und „Beschreibung“ versehen. Hiermit ist die erste Ebene des MainMenus fertig erstellt.

Im weiteren Verlauf werden die nächsten beiden Panel kreiert, einmal das AudioPanel und ebenso das BeschreibungsPanel.

Da das MainMenuPanel bereits einige Funktionen enthält, die für die weiteren Panel ebenfalls benötigt werden, wird dieses noch einmal dupliziert.

Erstmals wird die Erstellung des AudioPanels beschrieben. Hier wird lediglich ein Button für die „Zurück“-Funktion benötigt. Zwei weitere Label werden eingefügt, welche die Texte „Audio Avatar“ und „Audio Soundeffekte“ enthalten. Jedem Label wird ein UI Slider zugewiesen. Um die Position der Slider, also der Schieberegler, zu ändern, wird der Slider in der Hierarchie nach oben beziehungsweise nach unten verschoben. Auch dieser spezielle Slider entspricht nicht dem Default Slider von Unity, sondern wurde dem Asset-Package entnommen.

Um das Beschreibungs-Panel zu erstellen, wird das Audio Panel dupliziert und alle Elemente, bis auf das Label und ein Textobjekt, gelöscht.

In das Label wird die Beschreibung für die App eingefügt. In der untenstehenden Grafik kann der besagte Text betrachtet werden.

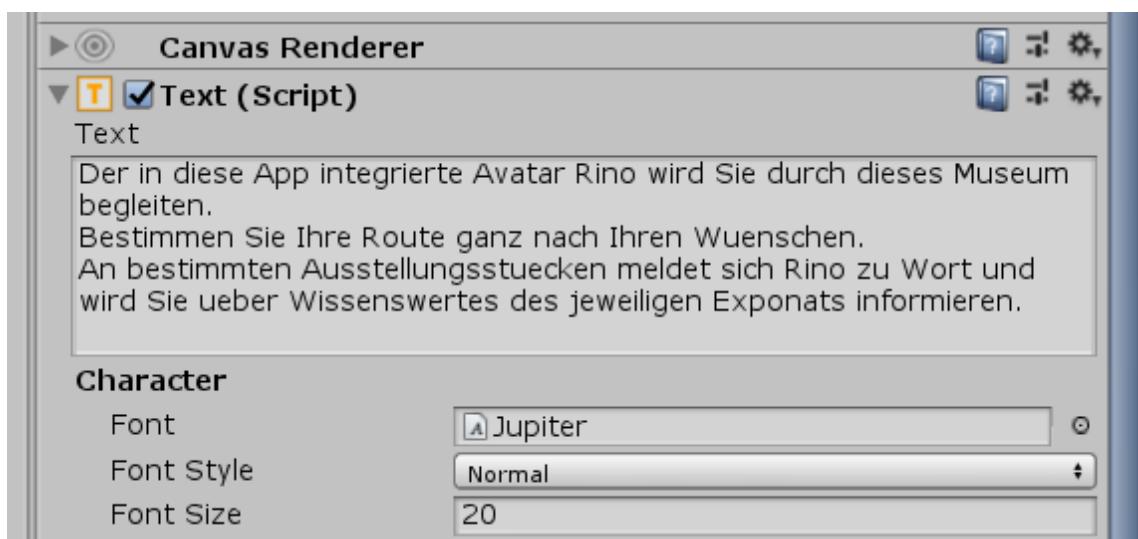


Abbildung 56: Beschreibungs-Text für das Label



Abbildung 57: Beschreibungs-Text im Play-Mode beim Abspielen der Szene

Hier ist es sinnvoller, die horizontalen und vertikalen Overflow-Werte, anders als bei den Buttons, auf „Wrap“ und „Truncate“ zu setzen, damit automatisch Textumbrüche entstehen und der eingegebene Text sich in das Panel einfügt.

In der Hierarchie sollten die einzelnen Elemente entsprechend benannt werden.

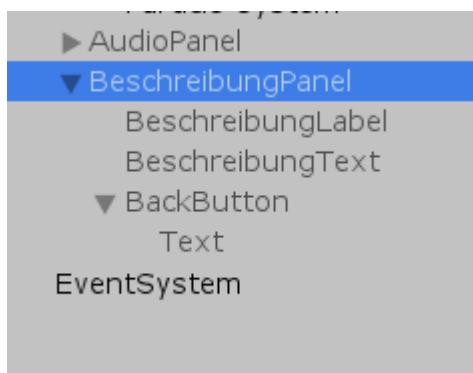


Abbildung 58: Beispiel für die Hierarchie des Beschreibungs-Panels

Im weiteren Verlauf werden den Buttons einige Basic-Aktivitäten hinzugefügt.

Der „Navigation Starten“ Button erhält ein kleines Skript, dessen Funktion es ist, eine Szene zu laden, die auf der Index-ID der Szene basiert. Dafür wird unter dem Reiter „Build Settings“ die geöffnete Szene „MainMenu“ hinzugefügt, die im Verlauf dieses Kapitels erstellt wurde. Diese besitzt die Index-ID „0“. Das bedeutet, dass dies die erste Szene ist, die beim Starten der App geladen wird. Es wird eine bestehende „Control Scene“ aus dem Asset-Package in die „Build Settings“-Methode hinzugefügt. Der „Control Scene“ wird der Index „1“ zugewiesen. Wenn nun zu der „Control Scene“ gewechselt werden soll, muss die ID 1 angesprochen werden.

Als Nächstes wird das Skript geschrieben, um die „Control Scene“ zu laden, sobald der „Navigation Starten“-Button gedrückt wurde.

Der Button „Navigation Starten“ muss in der Hierarchie markiert sein, um eine neue Komponente hinzuzufügen. Hier wird der selbst erdachte Name des Skripts „LoadSceneOnClick“ eingegeben. Da dieses noch nicht existiert, muss an dieser Stelle ein neues C# Skript erstellt und hinzugefügt werden.

Diese Szene wird in Visual Studio geöffnet.

Eine public Funktion „LoadByIndex“ wird erstellt, die einen Integer-Parameter namens „SceneIndex“ enthält. Beim Aufrufen dieser Funktion wird eine Szene mit dem SceneManager geladen. Um diesen nutzen zu können, muss er zu Beginn des Editors durch „using UnityEngine.SceneManagement;“ importiert werden.

```
public class LoadSceneOnClick : MonoBehaviour
{
    public void LoadByIndex(int sceneIndex)
    {
        SceneManager.LoadScene(sceneIndex);
    }
}
```

Die geladene Szene wird diese sein, die beim Aufrufen der Funktion gerade übergeben wird.

Der Integer-Wert wird vom Button übergeben, also wird der Wert im Button selbst gesetzt. Durch den Button können die verschiedenen Szenen geladen werden.

Um die Aktionen durchzuführen, beziehungsweise damit der Button die Funktionen aufruft, wird die „OnClick()“-EventList in Unity genutzt. In die EventList wird das gerade erstellte C# Skript importiert. Hiermit kann dem Button „Navigation Starten“ die LoadSceneOnClick-Methode zugewiesen werden. Außerdem kann ausgewählt werden, dass das Skript beim Index geladen werden soll. Dem Integer muss an dieser Stelle der gewünschte Wert zugewiesen werden. Da die „ControlScene“ den Integer-Wert „1“ besitzt, wird in das Parameterfeld eine „1“ eingetragen, weil die „ControlScene“

geladen werden soll, wenn der User den Button „Navigation Starten“ betätigt.

Die „Control Scene“ dient hier nur als Beispiel. In der fertigen App wird an dieser Stelle der Code für die Navigation eingesetzt.

Die Verknüpfung der weiteren App-Funktionen erfolgt auf eine andere Weise.

Beim Aufrufen des Beschreibungs Panels wird kein Skript benötigt. Es wird ein neues OnClickEvent() erstellt.

Anstelle der LoadSceneOnClick-Methode wird an dieser Stelle lediglich das MainMenu Panel in das entsprechende Feld gezogen. Es soll erreicht werden, dass das MainMenu Panel in dem Moment ausgeblendet wird, sobald ein Klick auf das Beschreibungs Panel erfolgt. Nun wird unter GameObject SetActive(bool) gesetzt. Durch die Checkbox kann die boolesche Variable entweder auf ‚true‘ oder auf ‚false‘ gesetzt werden. Die ausgewählte Checkbox steht für ‚true‘.

Dadurch wird das MainMenu Panel auf inaktiv gesetzt.

Es wird ein weiteres Objekt benötigt, durch das die neue Ebene gestartet wird. Aus diesem Grund wird das Beschreibungs Panel hinzugefügt. Alle Komponenten in der Liste werden beim Klicken des Buttons abgerufen. Hier wird die Funktion wieder auf SetActive und auf ‚true‘ gesetzt.

Daraus resultiert, dass bei einem Klick auf den Button „Beschreibung“ im MainMenu Panel die Weiterleitung zu dem Beschreibungs Panel mit dessen festgelegten Objekten und Formatierungen, also dem Label sowie dem Textobjekt erscheint.

Für das Audio Panel wird die gleiche Vorgehensweise angewendet. Somit wird beim Betätigen des „Audio“-Buttons das MainMenu Panel durch die nicht ausgewählte, also auf ‚false‘ gesetzte Checkbox, ausgeblendet und der User wird zum Audio Panel mit dessen

Lautstärke-Reglern geleitet. Dabei ist der Wert des AudioPanels durch die Checkbox auf ‚true‘ gesetzt.

Die Buttons „Zurueck“, welche sich auf den Beschreibungs und Audio Panels befinden, werden auf die gleiche Weise mit dem MainMenu Panel verknüpft, damit der Benutzer der App durch Drücken des „Zurueck“-Buttons wieder ins MainMenu gelangt.

In diesem Zustand kann das Menü soweit genutzt werden, dass sich die Navigation startet und die Lautstärke des Avatars sowie der Hintergrundgeräusche anhand der Regler verstellt werden kann. Außerdem findet der User die Beschreibung für die App, wenn der entsprechende Button ausgewählt wird.

Die nachfolgende Abbildung zeigt den fertigen Stand des Menüs in der Entwicklungsumgebung Unity. <sup>15</sup>

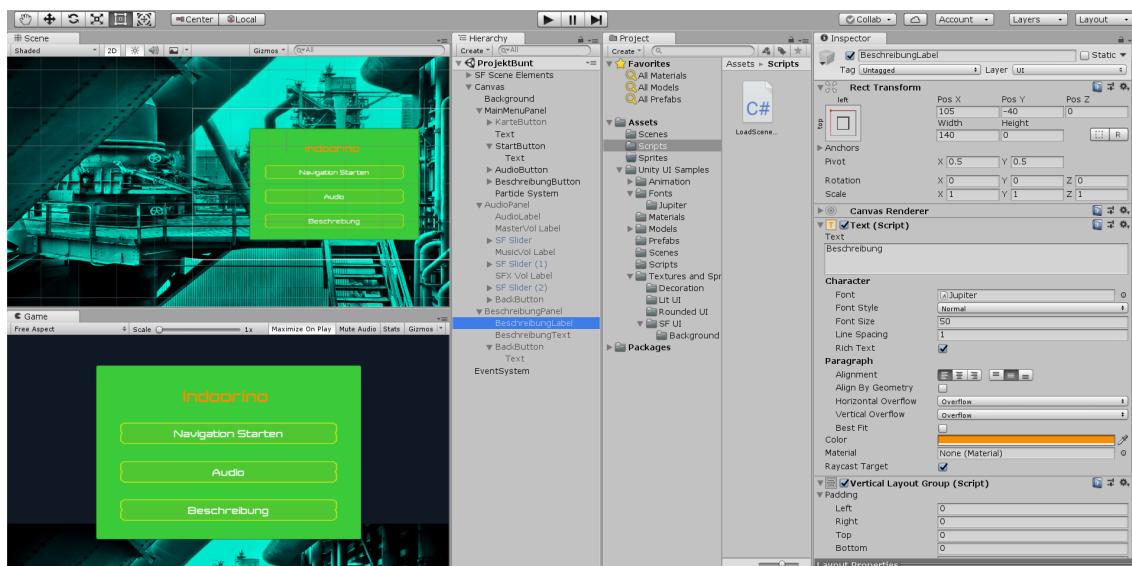


Abbildung 59: Unity Game Engine mit der Ansicht der fertigen Menü-Version.

### Design des Menüs

In den weiteren Schritten wurde sich über das Design des Menüs Gedanken gemacht. Der technische, industrielle Aspekt soll genauso zum Vorschein kommen, wie die auf die Zielgruppe angepasste Oberfläche. Da viele Kinder und Schulklassen zu den Museumsbesuchern gezählt werden, sollte das Menü eine kindlich verspielte

Seite besitzen. Hierfür wurde sich bezüglich der Farbigkeit für kräftige und fröhliche Farben entschieden.

Der Anwendungsbereich der App für das Industriemuseum wird durch den Hintergrund deutlich, auf welchem in die Jahre gekommene Stahlkonstruktionen im industriellen Design erkennbar sind. Dieses Hintergrundbild wurde in ein kaltes Türkisblau getaucht.

Es wurde auch bei den Formen der Panels experimentiert. Außerdem wurden verschiedene Texturen sowie einige Transparentwerte ausprobiert.

Das transparente Panel wirkt um einiges innovativer und moderner, leider hat dies aber den Nachteil, dass die Texte der App durch den Industriehintergrund für den Leser schwer erkennbar sind. Somit wurde sich für ein Einfärben des Panels im vollen Grün entschieden.

Um etwas mehr Abwechslung zu erhalten, färben sich die Buttons in violetten Farbtönen, sobald der User darüberstreicht. Wenn der Button betätigt wird, erscheint eine orange Umrandung, dieser Farbton spiegelt sich bereits in den Überschriften wider.

Bilderreihe zum fertigen Menü der Indoorino-App:



Abbildung 60: MainMenu Panel mit den verschiedenen Buttons

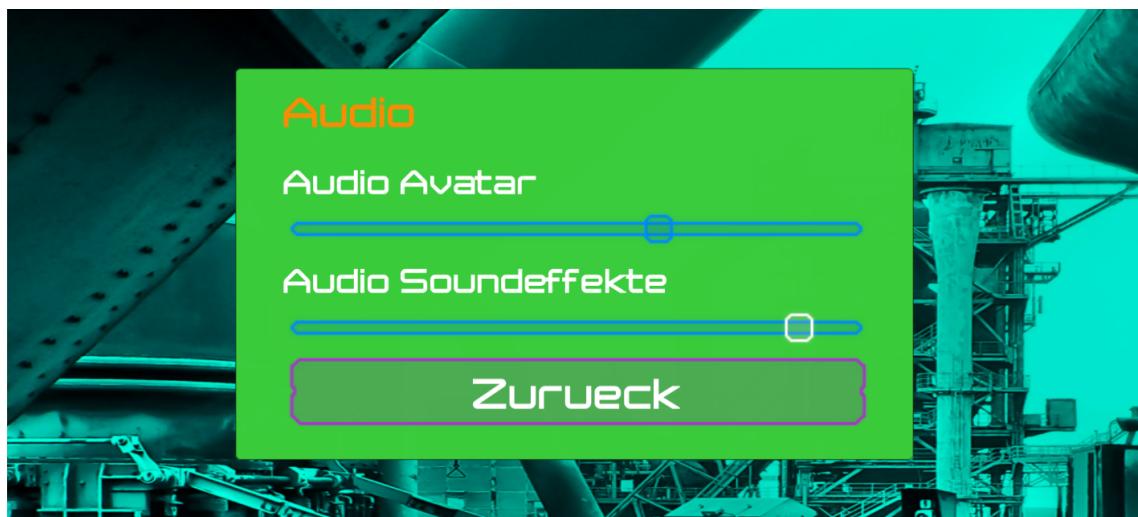


Abbildung 61: AudioPanel mit den Sliders zur Regelung der Lautstärke



Abbildung 62: Beschreibungspanel

## 4 Marketing und Präsentation

### 4.1 Logo

Beim Entwerfen des Logos lag der Fokus auf einer klaren Kernbotschaft sowie der Funktion unserer App. Es sollte einfach sein und funktionieren. Zudem sollte das Logo variabel einsetzbar sein, weshalb wir auf zusätzliche Elemente in Verbindung zu einem Museum verzichtet haben. Als hervorhebendes Element wurde der letzten Buchstaben zu einem Navigationssymbol abgeändert, welches zugleich im Bild Logo integriert wurde. Im Gegensatz zu den üblichen Farben für Navigationssymbole fiel unsere Wahl auf ein helles Grün. Im Sinne unserer App soll dies ein „GO!“ bzw. „Hier bist du richtig“ symbolisieren.

Da unsere Applikation für mobile Endgeräte ausgelegt ist haben wir uns für eine seriflose Schriftart (*Yu Gothic UI*) entschieden, zumal diese auch in verkleinerter Darstellung gut lesbar ist und dennoch eine dynamische Wirkung erzielt.



Abbildung 63: Indoorino Logo

### Slogan

Der Slogan für *indoorino* sollte kurz, passend und einprägsam sein. Zusammen haben wir einige Vorschläge für einen angemessenen Slogan erarbeitet.

- *Mehr entdecken*
- *Frustfrei finden*
- *Navigieren und Informieren*
- *Die App die navigiert, nicht frustriert*
- *Durch unser Führen öffnen sich Türen*
- *Viele Räume, viele Ziele, eine App*
- *Zielführend*
- *Zügig zum Ziel*
- *Führend zum Ziel*
- *Wir zeigen wo's lang geht*
- *Wir wissen wo's lang geht*

Aufgrund seiner prägnanten Aussagekraft ging der Slogan „*Durch unser Führen öffnen sich Türen*“ als klarer Favorit hervor.

Zum einen besteht sofort ein Bezug zu unserem Logo da alle Elemente darin vorkommen. Durch den Reim wird der Slogan einprägsamer. Die Doppeldeutigkeit vom „Türen öffnen“ fanden wir in Bezug auf ein Museum sehr passend. Somit wird der User mithilfe unserer App geführt und ihm wird zugleich die Tür zu neuem Wissen geöffnet.

## 4.2 Plakatdesign

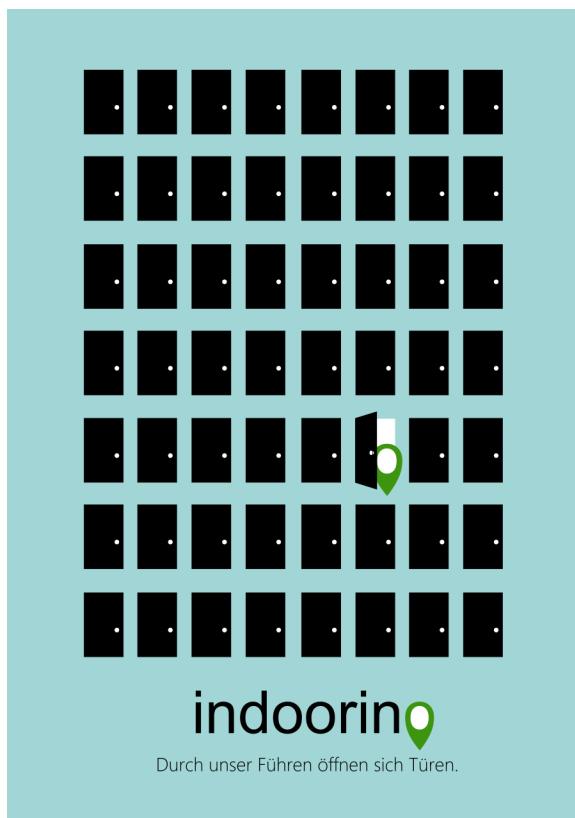


Abbildung 64: Indoorino Poster

kurzer Zeit die Kernaussage des beworbenen Produkts zu übermitteln. Um diesen Effekt zu erzielen, haben wir ein minimalistisches Designkonzept mit den essentiellen Wirkelementen für unsere App erarbeitet, rein nach dem Prinzip „Keep it short and simple“. Zu Beginn lenken wir den Blick des Betrachters auf den massiven Block aus Türen der sich trotz seiner schwere durch den intensiven Kontrast-unterschied vom hellen Hintergrund abhebt. Diese Anordnung wird durch das *indoorino*-Logo unterbrochen, welches zugleich den Blick des Betrachters durch sein farbiges Attribut zum Schriftzug und dem *indoorino*-Slogan führt.

Alle grafischen Elemente für das Corporate Design wurden von uns mit GIMP 2 und Inkscape eigenhändig erarbeitet.

Eine weitere Aufgabe für den 1. Projektabschnitt war die Gestaltung eines Plakats für unsere Projekt-präsentation. Bei der Plakat-gestaltung sollte man immer im Hinterkopf behalten, dass die Betrachtungszeit in den meisten Fällen nur wenige Sekunden beträgt. Daher sei die Gestaltung mit Grafiken, Text und Farben wohl durchdacht, um dem Betrachter in



Abbildung 65: Logo Gimp & Inkscape [60]

## **4.3 Video**

Funktion:

Das Video dient als Intro für die Präsentation und soll auf mögliche Einsatzbereiche der Innenraumnavigation deuten.

Exposé :

Das kurze Werbevideo zeigt die Problematik einiger Menschen in Alltagssituationen sich in Innenräumen zurecht zu finden.

Es existieren drei Storylines:

1. „Student sucht den richtigen Raum in der Hochschule“

Der Student irrt verzweifelt durch die Hochschule und sucht vergeblich nach dem richtigen Raum.

2. „Bewerberin die sich verspätet“

Die Bewerberin muss erst durch eine menge ähnlicher Büroräume laufen, bevor Sie viel zu spät zum Vorstellungsgespräch erscheint, was ihre Chance auf einen neuen Job kostet.

3. „eine Person die keine Innenraumpläne lesen kann“

Eine Person betrachtet einen Innenraumplan und noch bevor die Suche nach dem Raum losgehen kann, wird der Innenraumplan vor Verzweiflung zerrissen.

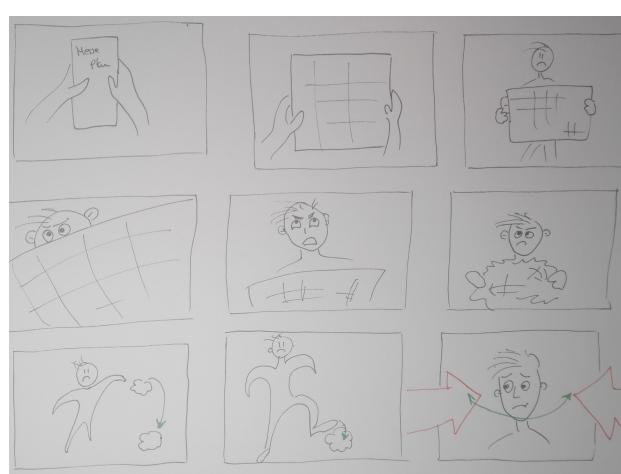
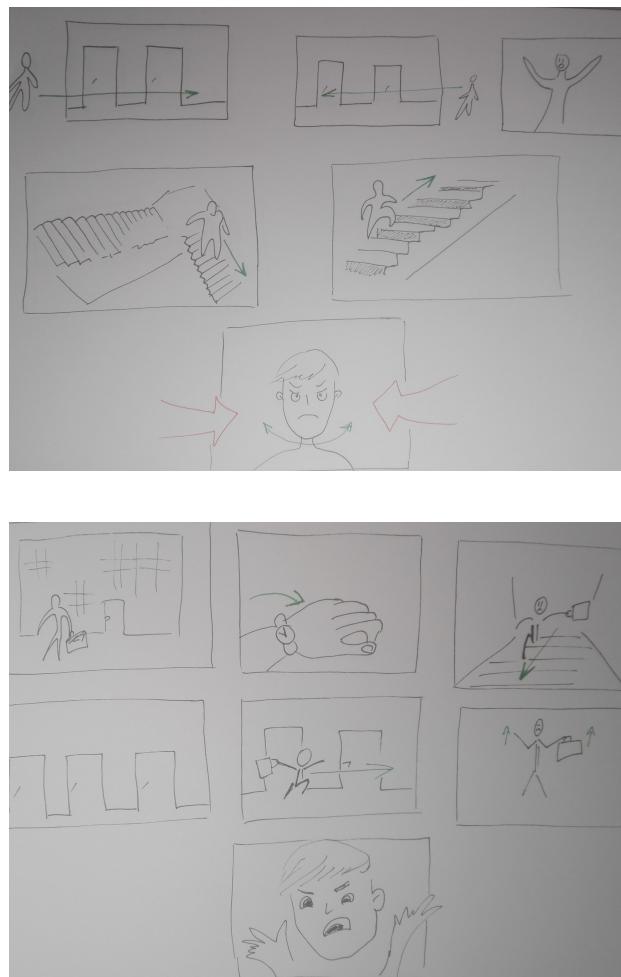
Als Lösung zu diesen Problemen wird der Produktnname „Indoorino“- Die Innenraumnavigation via Logo eingeblendet.

Stil:

Das Werbevideo ist im Infomercial- Stil umgesetzt. Das ermöglicht den Einstieg in die Präsentation mit Humor.

Außerdem wird auf die Problematik der Innenraumsuche eingegangen und alles etwas dramatisch und übertrieben (so wie es in den Infomercials gemacht wird) dargestellt.

Storyboard:



Abbildungen 66: Storyboard Video

## Schnitt:

Die Postproduktion ist in Premiere Pro realisiert. Premiere ist ein intuitives Programm, das einwandfrei bei kleineren Projekten funktioniert. Das Video ist nach dem Drehbuch geschnitten und mehrfach gekürzt.

Auch das Colorgrading und Motion Graphics lassen (Bauchbinden) sich in Premiere anfertigen.

Anschließend ist die Datei als MPEG2 Exportiert.s

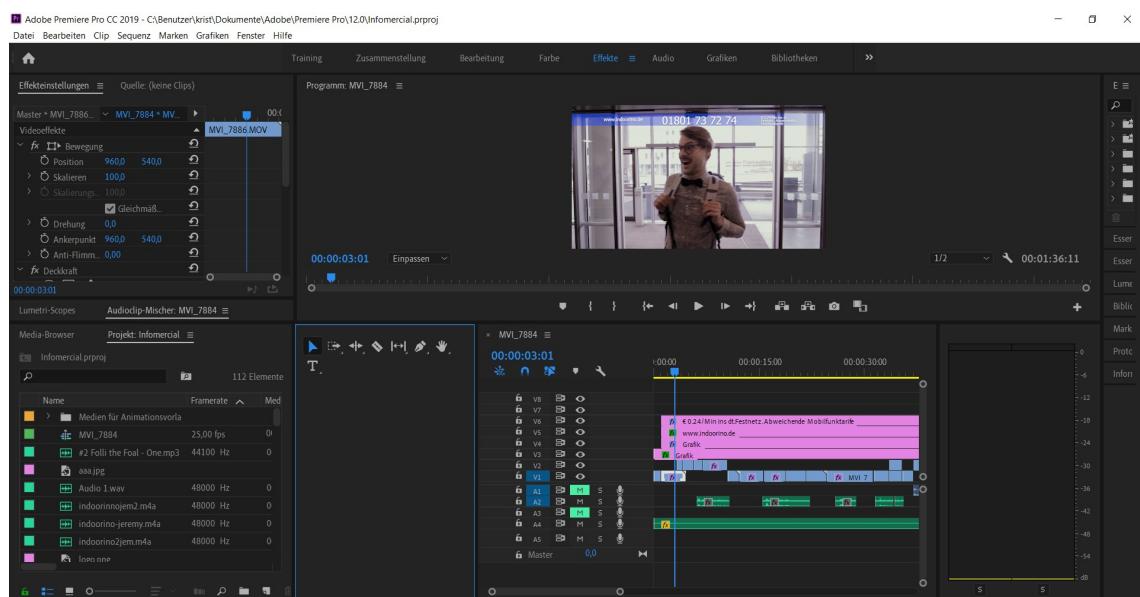


Abbildung 67: Adobe Premiere Pro Videoschnittübersicht

## **5 Zusammenfassung und Ausblick**

### **5.1 Resumé der Projektarbeit**

Das erfolgreiche Absolvieren einer Projektarbeit setzt sich aus den Kernpunkten der Projektorganisation sowie der Bereitschaft im Team zielorientiert zusammen zu arbeiten, um daraufhin eine Projektpräsentation verwirklicht zu haben. Dabei werden diverse Fertigkeiten im Bereich der sozialen- sowie organisatorischen Kompetenz gefordert. Unser Projekt durchlief hierbei verschiedene Entwicklungsstufen, die durch einige Rückschläge, jedoch auch von Fortschritten zu skizzieren sind.

Da wir zu Beginn wenig Erfahrung im Bereich des Projektmanagements sowie der Teamarbeit aufwiesen, wurden wir schon zu Beginn vor eine Herausforderung gestellt.

Uns fiel es schwer sich an bestimmte Themen oder Aufgaben anzunähern um einen geeigneten Weg der Problemlösung zu finden.

Eine kleinere Hürde die dazu kam, war die Unwissenheit über den technischen Kenntnisstand und Arbeitsverhalten jedes einzelnen innerhalb der Gruppe. Das erschwerte zu Beginn das Arbeiten an unserem Projekt, da man sich zunächst öfter untereinander abstimmen musste, um einen gemeinsamen Workflow zu erlangen.

Fortführend kam es bei der allgemeinen Kommunikation zu Anlaufschwierigkeiten, da nicht ausreichend miteinander kommuniziert wurde. Dies beeinträchtigte die Entwicklung des Projektes, da es zu Missverständnissen innerhalb, als auch außerhalb des Teams kam. Fortlaufend bleibt auch zu erwähnen, dass bis dato auch einige technische Rückschläge, die Entwicklung des Arbeitsprozess verlangsamt.

Zwar gab es wie oben genannt einige Rückschläge die die Projektentwicklung beeinträchtigten, allerdings wurde dadurch der Zusammenhalt des Teams gestärkt. Wir haben uns gegenseitig unterstützt und versucht bei Problemen den besten Lösungsweg zu finden. Durch das Projekt konnte jeder einzelne, sein Durchhaltevermögen unter Beweis stellen und seine kreativen und organisatorischen Qualitäten zeigen.

Wir entwickelten uns nicht nur auf sozialer Ebene sondern auch technischer Ebene weiter und können aus diesem Grund nun motiviert an weitere Aufgaben für das nächste Semester angehen.

Wir möchten nun für die Zukunft an den oben beschriebenen Problemen arbeiten und an den positiven Fortschritten festhalten um gemeinsam unser Projektziel nächstes Semester zu erreichen.

## **5.2 Ausblick: Zoorino**

### **5.2.1 Idee**

Ziel dieses Semesters war eine Indoornavigation für das Industriemuseum in Lauf zu implementieren, welche die Technologie des RTT verwendet. Da zu Beginn des Projektes das Releasedate des RTT Softwareupdate für die Google Wifi Stationen auf den Winter 2018 gefallen ist, haben wir uns zunächst dafür entschieden GPS für den Prototypen zu verwenden.

Nachdem nun einige Zeit vergangen war hat sich schlussendlich herausgestellt, dass sich der Releasetermin für das Update noch ein weiteres mal verzögert. Auf Grund dessen haben wir uns dazu entschieden uns nun auf GPS zu konzentrieren und unsere Anwendung auch darauf auszulegen. Zwar bietet das Industriemuseum Lauf auch einen Außenbereich an, allerdings ist dieser mit

wenigen Ausstellungspunkten bestückt weshalb wir zu der Idee gekommen sind unsere App speziell für Tiergärten auszulegen. Hierbei wollten wir besonderen Fokus auf den Nürnberger Tiergarten legen.

Wir möchten hierbei das Konzept der Navigation beibehalten und die Möglichkeit bieten verschiedene Routen zusammen zu erstellen um die gewünschten Tiere zu besuchen. Außerdem möchten wir auch an der Idee des Augmented Reality festhalten und eine kleine Anwendung in die App einbauen.

### **5.2.3 Konzeptausblick**

#### **Avatar „Zoorino“**

Unser Avatar „Zoorino“ soll eine Erweiterung der aktuellen Standortinformation des Users sein und mit dieser interagieren. Das Konzept der Interaktion mit dem User wird im Folgenden unter 2. *Navigation* genauer erläutert.

## 1. Starten der App und Basismenü

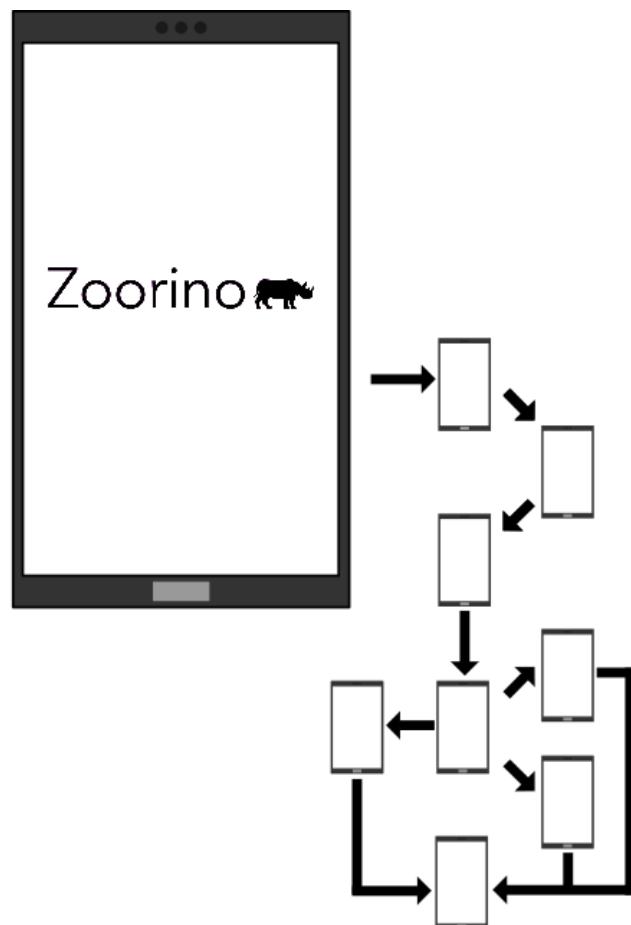


Abbildung 68: Starten der App

### 1.1.1 Starten der Applikation

App wird durch den User gestartet. *Zoorino* Logo erscheint auf dem Display.

### 1.1.2 Standortlokalisierung

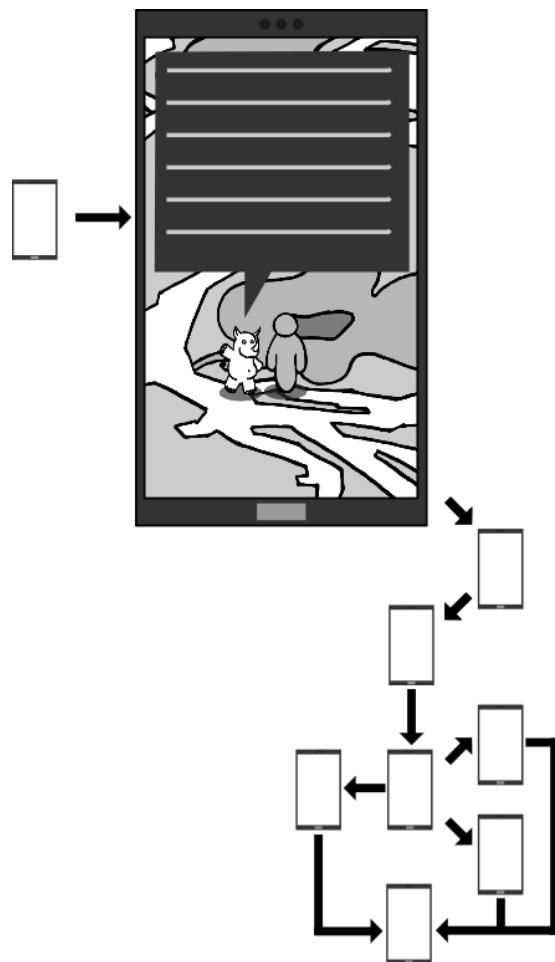


Abbildung 69: Standortlokalisierung der App

Sobald der Standort des Users lokalisiert wurde, erscheint neben seinem Icon sein persönlicher „Zoorino-Avatar“ und begrüßt ihn in einem Dialogfenster

**Zoorino: „Hallo und herzlich Willkommen im Nürnberger Tiergarten. Mein Name ist Zoorino. Ich bin dein persönlicher Tour Guide und werde dich heute durch den Nürnberger Tiergarten führen.“**

-> \* *Einwilligung des Users auf den Zugriff seiner Standortinformation.*

### 1.1.3 Hinweis auf AR-Aktivitäten

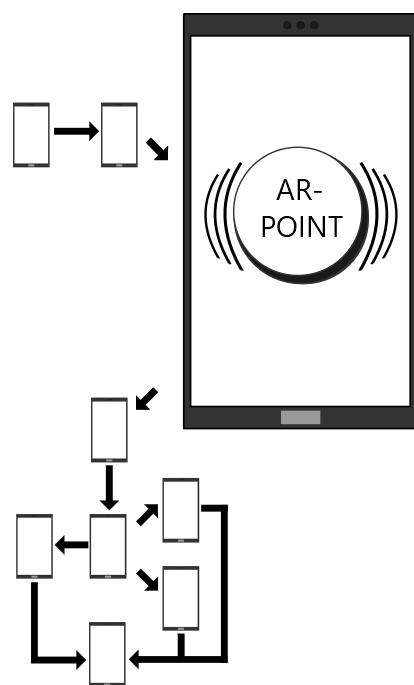


Abbildung 70: Hinweis AR - Funktion

>>>**Vibrationsalarm<<<**

**> AR-Button erscheint ->> User drückt auf Button**

Dialogfenster erscheint

Zoorino: „**Während unserer Führung wirst du sogenannte AR-Points entdecken. Nutze deine Kamera, sobald du auf solche AR-Points hingewiesen wirst, und sieh dich in deiner Umgebung um. Dann kannst du noch viel mehr spannende Dinge im Tiergarten erleben und mehr über seine Bewohner zu erfahren!**“

->\* *Einwilligung des Users auf den Zugriff der Kamera für die AR Funktion*

->\* *Erlaubnis für Benachrichtigung, wenn sich der User in der Nähe eines AR-Points befindet*

#### 1.1.4 Weiterleitung zum Hauptmenü



Abbildung 71: 'Weiterleitung zum Hauptmenü'

**Zoorino: „Was möchtest du sehen?“**

Der User wird durch tippen auf das Dialogfenster in das Hauptmenü weitergeleitet.

### 1.2.1 Hauptmenü

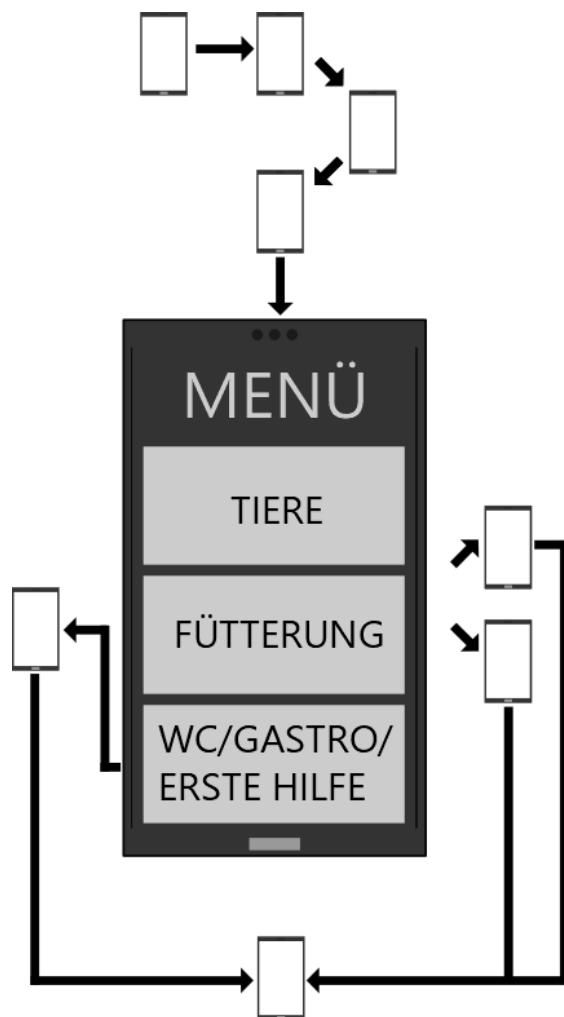


Abbildung 72: Hauptmenü der App

Der User hat die Auswahl zwischen 3 Menüpunkten:

**-Tiere**

**-Fütterung**

**-WC/Gastro/**

**Erste Hilfe**

Das Hauptmenü lässt sich zu jeder Zeit Aufrufen um eine neue

Auswahl zu treffen. Ein entsprechender Button befindet sich in der rechten unteren Bildschirmecke.

## 1.2.2 Tiere

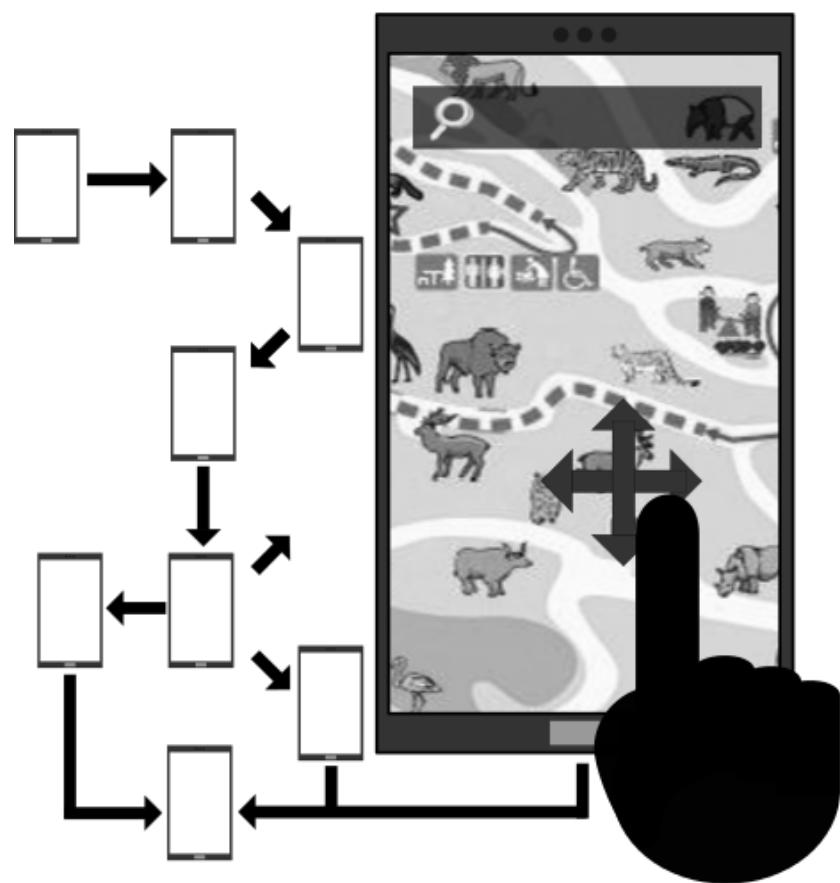


Abbildung 73: Tiere der App [61]

Dem User wird ein Kartenausschnitt des Nürnberger Tiergartens und seiner Bewohner gezeigt.

Durch das Verschieben der Karte in diverse Richtungen hat er die komplette Übersicht des Nürnberg Tiergarten und seiner Bewohner und kann sich aussuchen welchen er Besuchen möchte.

Drückt der User auf ein entsprechendes Tiersymbol kann er dadurch eine Führung zu diesem auswählen.

Eine manuelle Suche über eine Suchleiste im oberen Bildschirmbereich wäre eine weiter Idee.

### 1.2.3 Fütterung

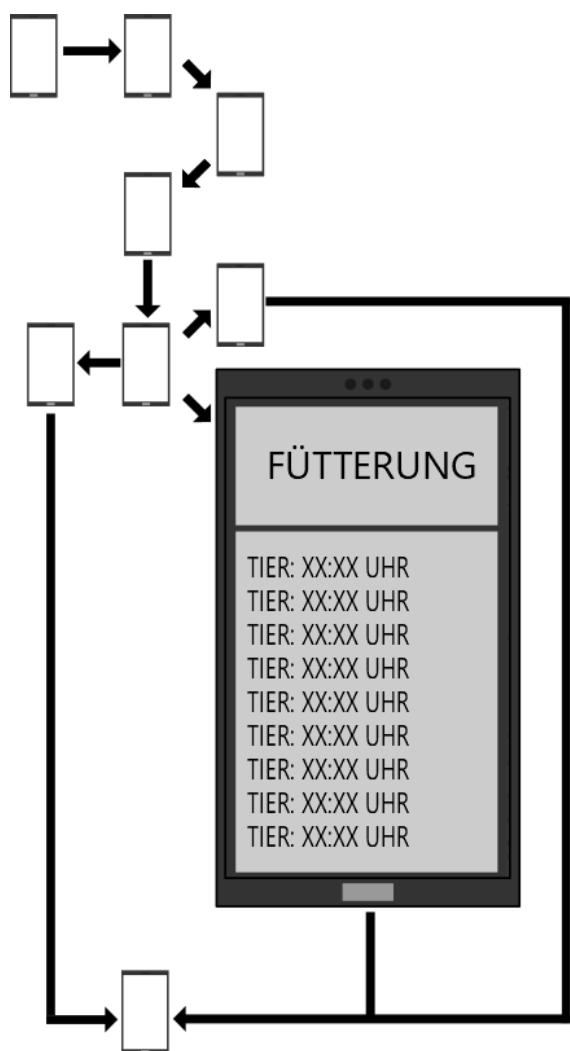


Abbildung 74: Fütterungsscreen

Der User bekommt eine Liste für die Fütterungszeiten diverser Tiere des Nürnberger Tiergartens angezeigt.

Durch drücken auf den entsprechenden Tiernamen kann er eine Führung zu der gewünschten Tierfütterung auswählen.

#### 1.2.4 Gastro/ WC / Erste Hilfe

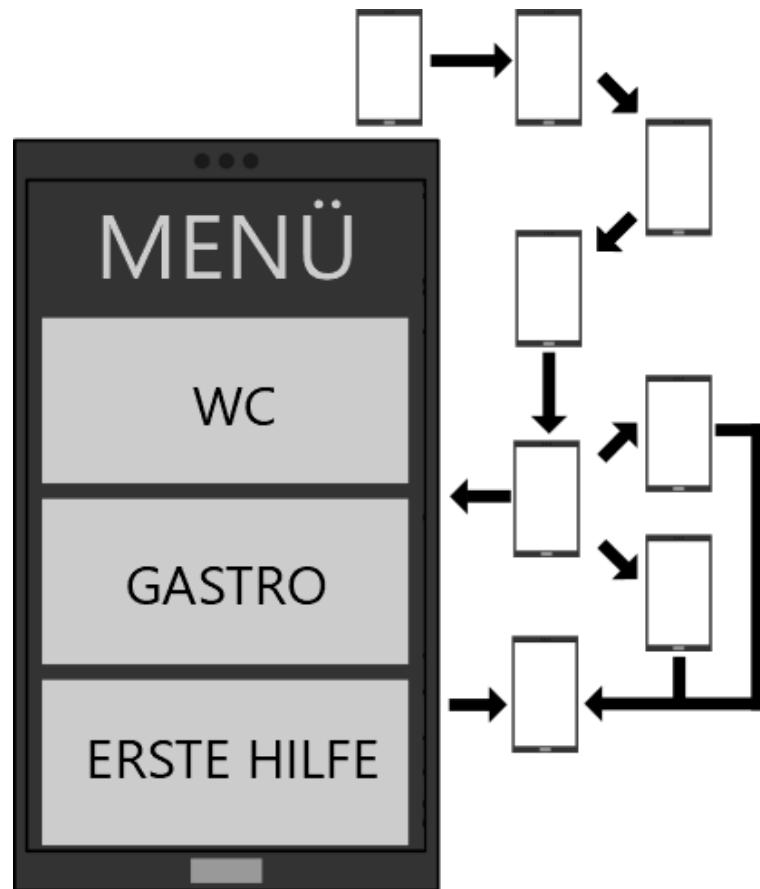


Abbildung 75: Infomenü

Der User wird zu einem neuen Menü weitergeleitet.

Hier kann er Eine Führung zur Gewünschten Örtlichkeit auswählen.

Die Menüpunkte sollen später durch die entsprechenden Symbole ersetzt werden.

Bei den Menüpunkten WC/ Erste Hilfe soll die App sofort die Route zur nächstgelegenen Örtlichkeit aufzeigen.

Der Menüpunkt Gastro funktioniert wie der Menüpunkt 1.2.2 mit einer Karte des Nürnberger Tiergartens. Hier soll man eine Führung zu den verschiedenen Restaurants und Bistros des Tiergartens erhalten.

### 1.2.5 Weiter zur Führung

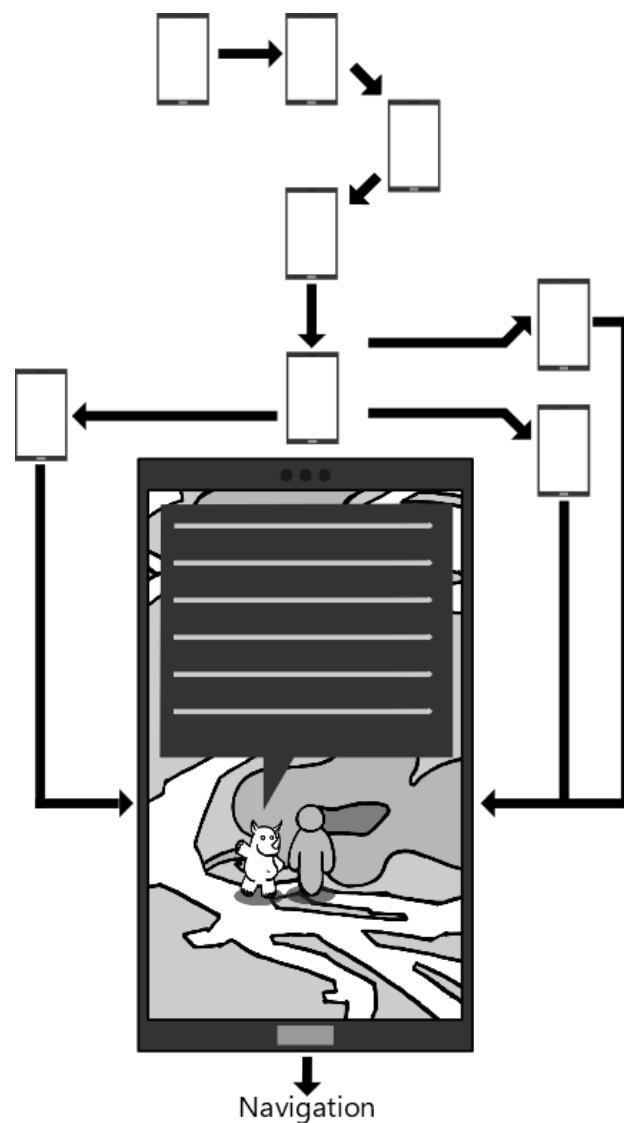


Abbildung 76: Führungsscreen

Nach der Auswahl des Users wird er zum Dialogfenster zurückgeführt.

**Zoorino: „Alles klar. Ich führe dich nun zu den  
„>>AUSWAHL<<. Lass uns starten!“**

Bestätigung durch User.

Weiterleitung zur Navigation.

## 2. Navigation

### 2.1.1 Führung Starten

**Situationsbeispiel:** Unser User möchte heute die Löwen besuchen. Über das Hauptmenü hat er über den Menüpunkt Tiere die Löwen auf dem Plan des Nürnberg Tiergartens ausgewählt und hat bestätigt, dass er seine Führung starten möchte.

### 2.1.2 Beginn der Führung

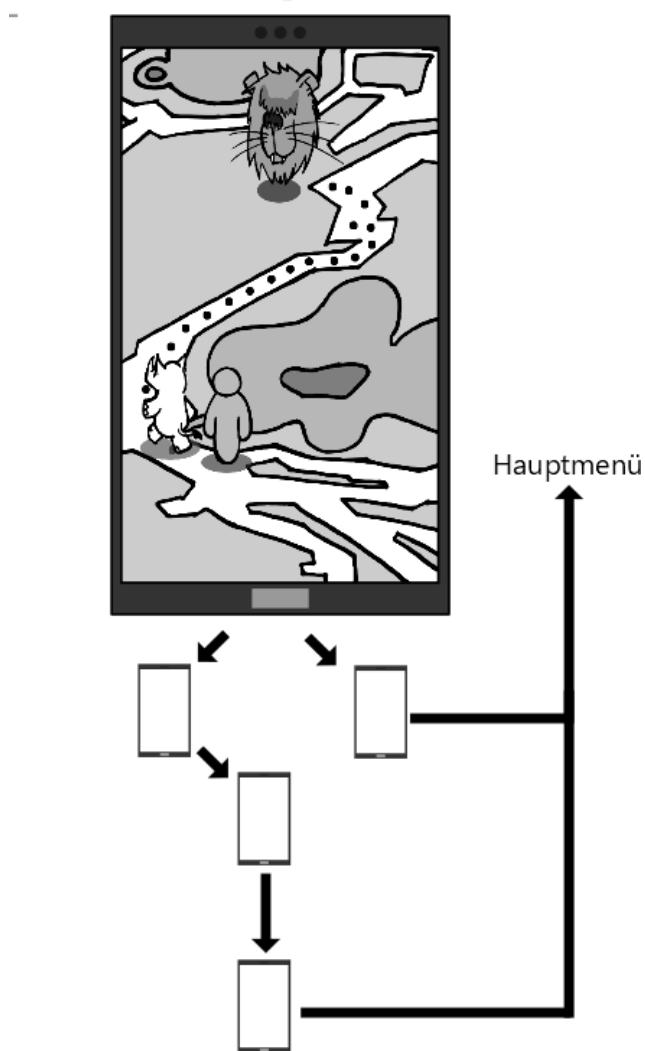


Abbildung 77: Beginn der Führung

Ziel (Tiersymbol) wird auf der Karte angezeigt.

Route zum Ziel wird auf der Karte farbig hinterlegt.

Zoorino fängt an sich gen Ziel zu bewegen.

**>> Der Avatar soll auf die aktuelle Position des Users reagieren. <<**

- > Befindet sich der User in unmittelbarer Nähe des Avatars bewegt dieser sich auf der Route weiter.
- > Die aktuelle Position des Users soll in bestimmten Intervallen vom System abgefragt werden.

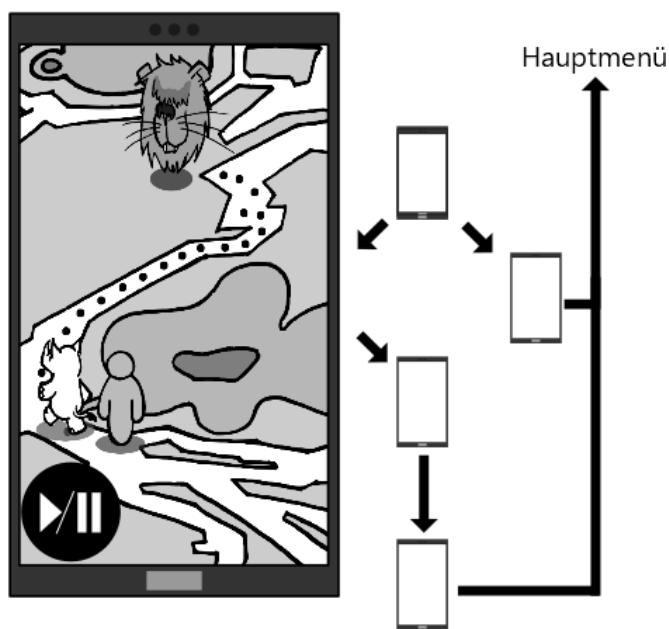


Abbildung 78: Pause während der Führung

Falls der User die Tour pausieren möchte soll es einen Pause-Button geben welcher – wenn vom User betätigt – auch zum Fortsetzen der Führung genutzt werden soll.

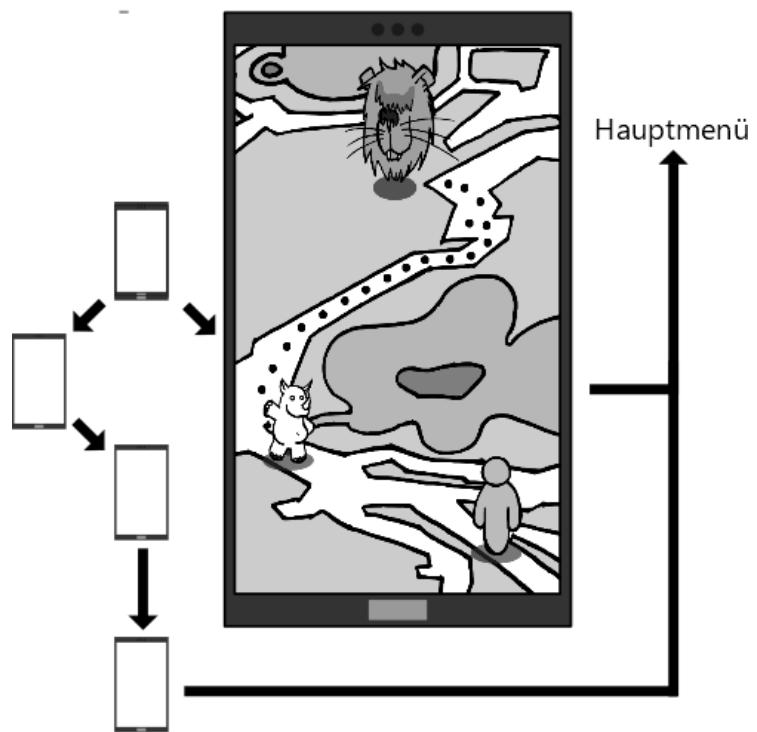


Abbildung 79: Während der Führung

Falls der User während einer Führung für längere Zeit inaktiv ist bzw. sich für längere Zeit außerhalb des Bereichs seines Avatars befindet, wird er automatisch ins Hauptmenü weitergeleitet und muss die Führung neu starten.

### 2.1.3 Ziel erreicht

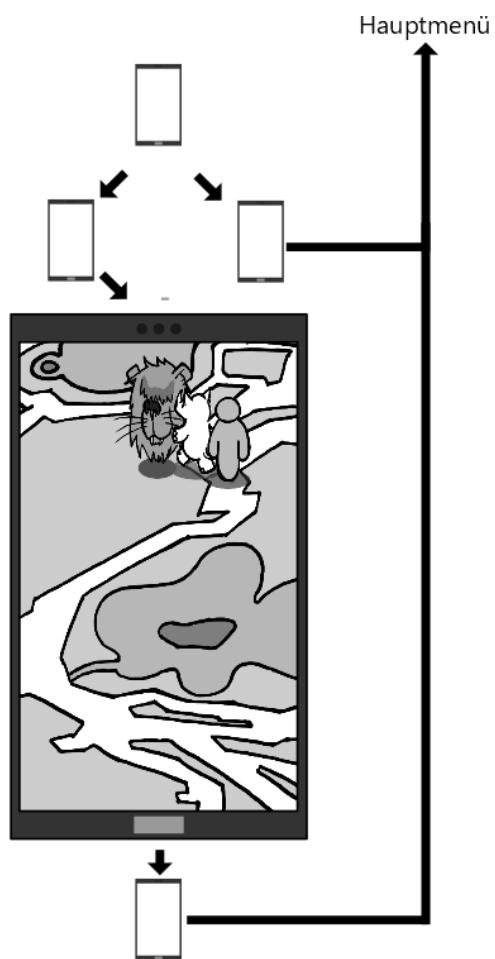


Abbildung 80: Ziel erreicht

Hat der User sein Ziel erreicht, wird er über einen Vibrationsalarm und einem PopUp Fenster darüber informiert.

*Zoorino: „**Wir haben unser Ziel erreicht!**“*

## Information über das Tier

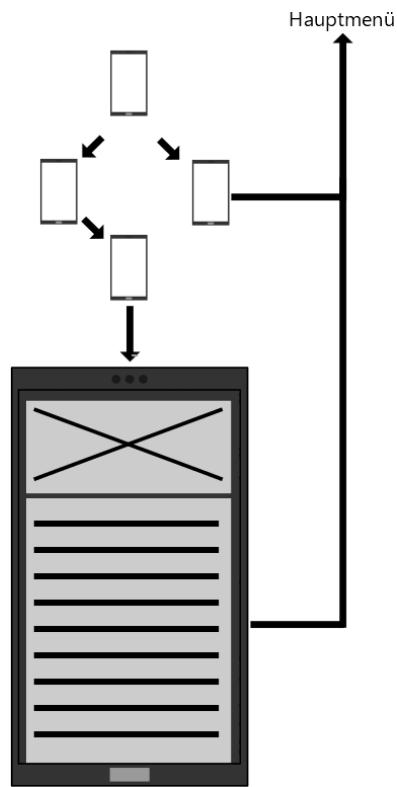


Abbildung 81: Informationen über Tier

Der User erhält in einem neun Fenster Informationen über das Tier wie z.B. Name, Geburtsdatum, allgemeine Informationen über die Gattung, natürlicher Lebensraum, Fun Facts, „Hast du gewusst dass...“ etc.

Eventuell einbinden eines 3D Modells des Tiers (siehe AR-Funktion).

Über den Menü-Button gelangt der User wieder ins Hauptmenü und kann eine neue Führung auswählen.

### **3. AR-Funktion**

Während sich der User auf Entdeckungstour im Nürnberger Tiergarten befindet, soll er während der Führung auf sogenannte AR-Points aufmerksam gemacht werden. Diese AR-Points befinden sich an diversen Stellen im Nürnberger Tiergarten. Eine Überlegung ist es die AR-Points auf den Führungs Routen oder an bestimmten Zielen zu verteilen. Der User soll hierbei durch einen Vibrationsalarm auf einen solchen AR-Point aufmerksam gemacht werden.

#### **3.1 Ideen für AR-Aktionen**

##### **Mini-Games (mit Highscore)**

- Fang den Dieb

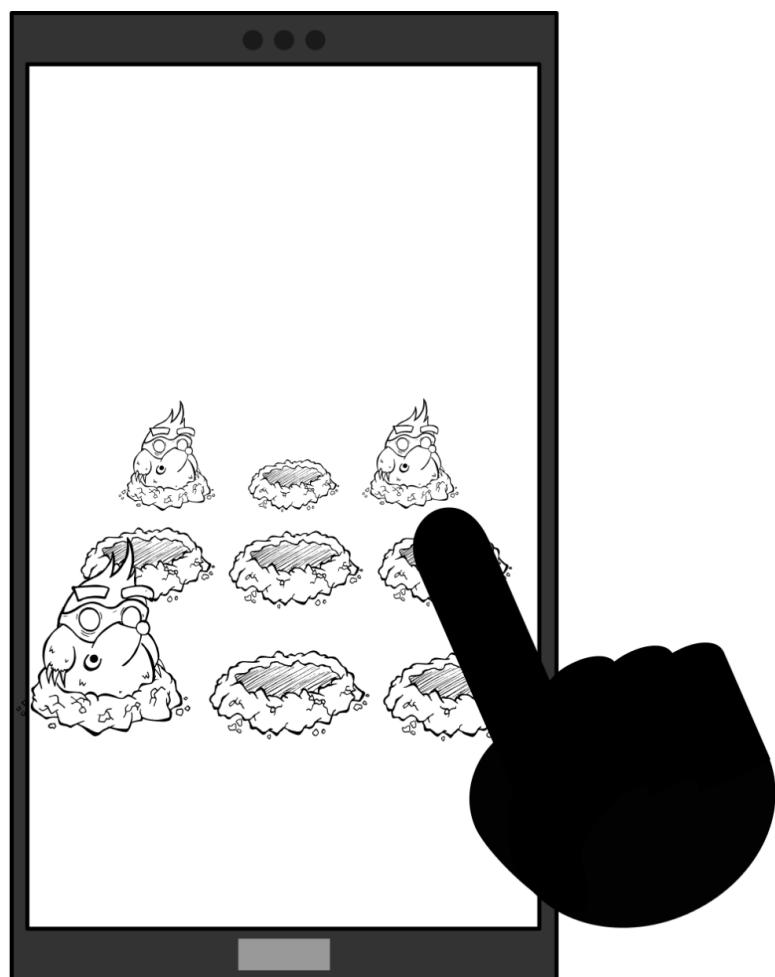


Abbildung 82: Minigame AR - Maulwurf

Ziel des Spiels ist es in 30 Sekunden so viele diebische Maulwürfe zu treffen um einen hohe Punktestand zu erreichen. Am Ende wird der Punktestand in einer Highscoreliste eingetragen.

## **Interaktion mit der Umgebung**

### **Schnappschuss mit Tieren**

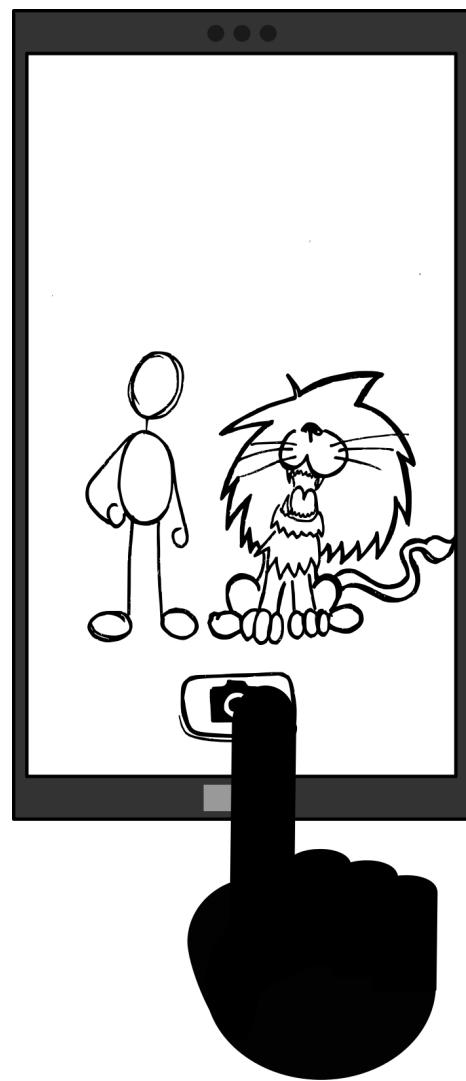


Abbildung 83: Schnappschuss mit Tieren

- 3D-Modelle der Tiere laufen außerhalb des Geheges herum
- Hier können die Tiere gefahrlos näher betrachtet werden

- Zudem besteht die Möglichkeit ein Schnappschuss mit dem Tier zu erstellen

Überlegung:

- AR Points also zusätzliche Option im Hauptmenü integrieren.
- Anpassung der Avatars an den User für alternativ Routen.
  - \* Für die optimale Nutzung der App ist die Einwilligung des Users auf seine Standortinformation, Kamera und die Erlaubnis für Benachrichtigungen durch die Zoorino App erforderlich.

## 7 Literatur- und Quellenverzeichnis

- [1] „Location based Game“. Verfügbar unter: [https://de.wikipedia.org/wiki/Location-based\\_Game](https://de.wikipedia.org/wiki/Location-based_Game)
- [2] „Pokemon-Go“. Verfügbar unter: [https://de.wikipedia.org/wiki/Pok%C3%A9mon\\_Go](https://de.wikipedia.org/wiki/Pok%C3%A9mon_Go)
- [3] „Mapbox“. Verfügbar unter: <https://www.mapbox.com/>
- [4] „Asana“. Verfügbar unter: <https://app.asana.com/>
- [5] „Monday“. Verfügbar unter: <https://monday.com/lang/de/>
- [6] „Github“. Verfügbar unter: <https://github.com/>
- [7] „Trello“. Verfügbar unter: <https://trello.com/>
- [8] „git-Der einfache Einstieg“. Verfügbar unter: <https://rogerdudler.github.io/git-guide/index.de.html>
- [9] „Wireless Local Area Network“. Verfügbar unter: [https://de.wikipedia.org/wiki/Wireless\\_Local\\_Area\\_Network#cite\\_note-1](https://de.wikipedia.org/wiki/Wireless_Local_Area_Network#cite_note-1)
- [10] „API-Level“. Verfügbar unter: <https://www.droidwiki.org/wiki/API-Level>
- [11] „Definition Beacon“. Verfügbar unter: <https://onlinemarketing.de/lexikon/definition-beacon>
- [12] „Visible Light Communications“. Verfügbar unter: [https://de.wikipedia.org/wiki/Visible\\_Light\\_Communications](https://de.wikipedia.org/wiki/Visible_Light_Communications)
- [13] „Transmission Control Protocol“. Verfügbar unter: [https://de.wikipedia.org/wiki/Transmission\\_Control\\_Protocol/Internet\\_Protocol](https://de.wikipedia.org/wiki/Transmission_Control_Protocol/Internet_Protocol)
- [14] „Android RTT Referenz“. Verfügbar unter: <https://developer.android.com/reference/android/net/wifi/rtt/package-summary.html>
- [15] Abbildung 3. Verfügbar unter: <https://slidex.tips/download/infsoft-gmbh-indoor-positions-bestimmung-navigation-ein-handbuch-zu-technologien>
- [16] Zitat aus: Indoor Positions Bestimmung & Navigation. Ein Handbuch zu Technologien und Use Cases, Insoft GmbH 20.01.2016. Verfügbar unter: <https://slidex.tips/download/infsoft-gmbh-indoor-positions-bestimmung-navigation-ein-handbuch-zu-technologien>. Zugriff am 24.02.2019
- [17] „Positionsermittlung anhand von Signalstärke“. Verfügbar unter: <https://www.gpsworld.com/how-to-achieve-1-meter-accuracy-in-android/>
- [18] „Compulab Wild Router“. Verfügbar unter: <https://fit-iot.com/web/products/wild/>
- [19] „Google Wifi Router“. Verfügbar unter: [https://store.google.com/de/product/google\\_wifi](https://store.google.com/de/product/google_wifi)
- [20] „Google Pixel 2“. Verfügbar unter: <https://www.amazon.de/Google-Pixel-XL-64GB-schwarz>

- [21] „Wifi RTT Verfahren“ übersetzt und zusammengefasst. Verfügbar unter: <https://www.gpsworld.com/how-to-achieve- 1-meter-accuracy-in-android/> Zugriff am 07.02.2019.
- [22] „FTM“ übersetzt und zusammengefasst. Verfügbar unter: [http://people.csail.mit.edu/bkph/ftmrtt\\_intro](http://people.csail.mit.edu/bkph/ftmrtt_intro) Zugriff am 07.02.2019 ]
- [23] „FTM Verfahren“. Verfügbar unter: [http://people.csail.mit.edu/bkph/ftmrtt\\_intro](http://people.csail.mit.edu/bkph/ftmrtt_intro)
- [24] „Android p and wi-fi rtt“. Verfügbar unter: <https://www.metaload.com/blog/android-p-and-wi-fi-rtt>
- [25] „Lateration“. Verfügbar unter: <https://de.wikipedia.org/wiki/Lateration>
- [26] „Lateration Bild“. Verfügbar unter: <https://de.wikipedia.org/wiki/Lateration#/media/File:Trilateration.png>
- [27] „Ftm rtt issues“. Übersetzt und zusammengefasst. Verfügbar unter: [http://people.csail.mit.edu/bkph/ftmrtt\\_issues](http://people.csail.mit.edu/bkph/ftmrtt_issues)
- [28] „Snapdragon 835 Chip“. Verfügbar unter: <https://www.qualcomm.com/products/snapdragon-835-mobile-platform>
- [29] „GPS“. Verfügbar unter: [https://de.wikipedia.org/wiki/Global\\_Positioning\\_System](https://de.wikipedia.org/wiki/Global_Positioning_System)
- [30] „Logo Unity“. Verfügbar unter: <https://unity3d.com/>
- [31] „Unity Learn“. Verfügbar unter: <https://unity3d.com/learn>
- [32] „Manual Unity“. Verfügbar unter: <https://docs.unity3d.com/Manual/index.html>
- [33] „Asset Store“. Verfügbar unter: <https://assetstore.unity.com/>
- [34] „Unity GPS Anfrage“. Verfügbar unter: <https://docs.unity3d.com/Manual/index.html>
- [35] „LocationService API“. Verfügbar unter: <https://docs.unity3d.com/ScriptReference/LocationService.Start.html>
- [36] „UnityPlayerActivity“. Verfügbar unter: <https://docs.unity3d.com/Manual/AndroidUnityPlayerActivity.html>
- [37] „Erdreferenzellipsoid“. Verfügbar unter: <https://technofab.uservoice.com/knowledgebase/articles/235104-perchè-l-altimetro-da-i-numeri>]
- [38] „Geographic Coordinate conversion“. Verfügbar unter: [https://en.wikipedia.org/wiki/Geographic\\_coordinate\\_conversion](https://en.wikipedia.org/wiki/Geographic_coordinate_conversion)

- [39] „Masseschwerpunkt der Erde“. Verfügbar unter: <https://www2.htw-dresden.de/~bilaj/vortrag/Dresden%2004.pdf>
- [40] „Koordinatensysteme“. Verfügbar unter: [https://upload.wikimedia.org/wikipedia/commons/thumb/6/6a/ECEF\\_ENU\\_Longitude\\_Latitude\\_Upwardness\\_relationships.svg/538px-ECEF\\_ENU\\_Longitude\\_Latitude\\_Upwardness\\_relationships.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/6/6a/ECEF_ENU_Longitude_Latitude_Upwardness_relationships.svg/538px-ECEF_ENU_Longitude_Latitude_Upwardness_relationships.svg.png)
- [41] AIAA Education Series; Applied Mathematics in Integrated Navigation Systems, Second Edition; Robert M. Rogers, 2003; S.42f
- [42] „Konvertierung zu ECEF“. Verfügbar unter: <http://www.joachim-bolz.de/projekte/gps/lhxyz.gif>
- [43] „Ellipsoid Referenzmodell“. Verfügbar unter: <http://support.virtual-surveyor.com/en/support/solutions/articles/1000261351-what-is-wgs84>
- [44] Verfügbar unter: <http://astronomy.swin.edu.au/cosmos/E/Ellipse>
- [45] „Krümmungsradius“. Verfügbar unter: <https://www.goruma.de/erde-und-natur/astronomie/erdumlaufbahn-keplersche-gesetze-exzentritzaet>
- [46] Verfügbar unter: <http://www.joachim-bolz.de/projekte/gps/gps.html>
- [47] „Download Libgdx“. Verfügbar unter: <https://libgdx.badlogicgames.com/download.html>
- [48] „Getting context of an app“. Verfügbar unter: <https://stackoverflow.com/questions/19301405/getting-context-of-an-app-launched-by-a-launcher>
- [49] „Verhalten API Android“. Verfügbar unter: <https://developer.android.com/reference/android/location/LocationManager>
- [50] „Dokumentation LibGDX“. Verfügbar unter: <https://xoppa.github.io/blog/loading-models-using-libgdx/>
- [51] „Konvertierung 3Ds zu .obj“. Verfügbar unter: <http://www.greentoken.de/onlineconv/>
- [52] „FBX-Converter“. Verfügbar unter: <https://github.com/libgdx/fbx-conv>
- [53] „StarUML“. Verfügbar unter: <http://staruml.io>
- [54] „3Ds Max Features“. Verfügbar unter: <https://reviews.financesonline.com/p/autodesk-3ds-max/>
- [55] „Blender Features“. Verfügbar unter: <https://reviews.financesonline.com/p/blender/>
- [56] „Hunderassen“. Verfügbar unter: <https://retro-vintage-design.de/hunderassen/>

[57] Zitat. Spiegel online, Samstag, **24.06.2006** 16:19 Uhr Ilka Lehnen- Beyer. Verfügbar unter: <http://www.spiegel.de/wissenschaft/natur/orientierung-tiere-haben-gps-ersatz-im-kopf-a-423234.html>. Zugriff am 20.01.2019

[58] „Grundfunktionen Unity“. Verfügbar unter: <https://unity3d.com/de/learn/tutorials/topics/user-interface-ui/creating-main-menu>

[59] „Asset Store“. Verfügbar unter: <https://unity3d.com/de/quick-guide-to-unity-asset-store>

[60] „Gimp/Inkscape Logo“. Verfügbar unter: <https://i.imgur.com/PZZqs.png>

[61] Karte Tiergarten Nürnberg - [https://tiergarten.nuernberg.de/typo3temp/\\_processed/\\_csm\\_Strecke\\_seit\\_2017\\_9910a7d1d9.png](https://tiergarten.nuernberg.de/typo3temp/_processed/_csm_Strecke_seit_2017_9910a7d1d9.png))

## **7 Abbildungsverzeichnis**

Abbildung 1: Zeitleiste Trello	15
Abbildung 2: Board Trello	15
Abbildung 3: Android RTT Referenz [14]	20
Abbildung 4: Verfahren zu Ermittlung des Innenraumstandortes [15]	21
Abbildung 5: Positionsermittlung anhand von Signalstärke [17]	22
Abbildung 6: Compulab Wild Router [18]	23
Abbildung 7: Google Wifi Router [19]	23
Abbildung 8: Google Pixel 2 [20]	24
Abbildung 9: FTM Ablauf [23]	25
Abbildung 10: Trilateration [26]	26
Abbildung 11: Unity Logo [30]	28
Abbildung 12: Unity Oberfläche	29
Abbildung 13: Unity „Add Tag“	32
Abbildung 14: Ordnerstruktur „src“	38
Abbildung 15: Kompilieren des Moduls	39
Abbildung 16: Inhalt entpackter .aar	39
Abbildung 17: Inhalt entpacktes .jar	39
Abbildung 18: Unity Zielplatform wählen	41
Abbildung 19: Library einbetten	41
Abbildung 20: Erdreferenzellipsoid [37]	43
Abbildung 21: Geländeumriss des Industriemuseum Lauf	44
Abbildung 22: Tabelle Ergebnisse Ellipsoid/Geoid Höhendifferenzberechnung	45
Abbildung 23: Koordinatensysteme [40]	46
Abbildung 24: Konvertierung zu ECEF	47
Abbildung 25: ECEF Berechnung in Android Studio	47
Abbildung 26: 2. ECEF Berechnung in Android Studio	48
Abbildung 27: ENU Berechnung in Android Studio	49

Abbildung 28: ENU Formel	50
Abbildung 29: Konvertierungsablauf ENU	50
Abbildung 30: LibGDX Setup Menü	51
Abbildung 31: LibGDX Struktur Android Studio	52
Abbildung 32: Verhalten API über Android	54
Abbildung 33: Verhalten API über Android	54
Abbildung 34: libGDX Koordinatensystem mit Achsenbezeichnung	57
Abbildung 35: Sequenzdiagramm des Prototypen	59
Abbildung 36: Sequenzdiagramm des Prototypen nach Implementierung	61
Abbildung 37: Messgerät	61
Abbildung 38 & 39: Testgelände BB Gebäude	62
Abbildung 40: Ausgemessen Testgelände BB Gebäude	63
Abbildung 41: Prototyp Testgelände BB Gebäude	64
Abbildung 42: Unity Prototyp	64
Abbildung 43: 3Ds Max Overview Features	65
Abbildung 44: Blender Overview Features	66
Abbildung 45: 3Ds Max Meter Setup	68
Abbildung 46: 3Ds Max System Unit Scale	69
Abbildung 47: 3Ds Max Übersicht	70
Abbildung 48: 3Ds Max Übersicht Kontrolle	70
Abbildung 49: Erste Skizzen Charakterdesign	72
Abbildung 50: Erste Skizzen spezifisches Charakterdesign	74
Abbildung 51: Finale Skizze spezifischer Charakter	75
Abbildung 52: Die im Asset-Package enthaltenen Rahmen	78
Abbildung 53: Der Button im Normalzustand	79
Abbildung 54: Der Button hervorgehoben bei Berührung	79
Abbildung 55: Der Button veränderten Farbzustand Klicken	79
Abbildung 56: Beschreibungs-Text für das Label	81
Abbildung 57: Beschreibungs-Text im Play-Mode beim Abspielen der Szene	81

Abbildung 58: Beispiel für Hierarchie des Beschreibungs-Panels	82
Abbildung 59: Unity Game Engine mit der Ansicht der fertigen Menü-Version	85
Abbildung 60: MainMenu Panel mit den verschiedenen Buttons	86
Abbildung 61: AudioPanel mit den Sliders zur Regelung der Lautstärke	87
Abbildung 62: Beschreibungspanel	87
Abbildung 63: Indoorino Logo	88
Abbildung 64: Indoorino Poster	90
Abbildung 65: Logo Gimp & Inkscape [60]	90
Abbildung 66: Storyboard Video	92
Abbildung 67: Adobe Premiere Pro Videoschnittübersicht	93
Abbildung 68: Starten der App	97
Abbildung 69: Standortlokalisierung der App	98
Abbildung 70: Hinweis AR - Funktion	99
Abbildung 71: Weiterleitung zum Hauptmenü	100
Abbildung 72: Hauptmenü der App	101
Abbildung 73: Tiere der Appanzeige	102
Abbildung 74: Fütterungsscreen	103
Abbildung 75: Infomenü	104
Abbildung 76: Führungsscreen	105
Abbildung 77: Beginn der Führung	106
Abbildung 78: Pause während der Führung	107
Abbildung 79: Während der Führung	108
Abbildung 80: Ziel erreicht	109
Abbildung 81: Informationen über Tier	110
Abbildung 82: Minigame AR - Maulwurf	111
Abbildung 83: Schnappschuss mit Tieren	112

## **8 Einzelnes Mitwirken am Projekt**

**1 Abstract** (Chantal)

**2 Konzeption**

2.1 Brainstorming (alle)

2.2 Konzeption und erste Ausarbeitung(Chantal, Jeremy)

2.3 Funktionalitäten der Anwendung(alle)

2.4 Projektname (Chantal, Inga, Jeremy)

2.5 Organisation

2.5.1 Planungstool (Philipp, Inga, Jeremy)

2.5.2 Versionskontrolle (Philipp, Inga)

2.5.3 Gruppeninterne Organisation (Philipp,Kristina,Inga)

2.6 Recherche

2.5.1 Android RTT(Philipp,Kristina, Chantal)

2.5.1 GPS - Global Positioning System(Philipp, Inga)

**3 Umsetzung**

3.1 Unity

3.1.1 Grundfunktionen (Philipp, Jeremy)

3.1.2 Objekt auf einer Route bewegen lassen

(Philipp, Jeremy)

3.1.3 Objektradius auf andere Objekte prüfen

(Philipp,Jeremy)

3.1.4 Unity GPS Abfrage(Philipp, Inga, Jeremy)

3.2 Android Studio

3.2.1 Android Library Plug-In für Unity(Philipp, Inga)

3.2.2 Android GPS Abfrage(Philipp, Inga)

3.2 Ortsabfrage und Berechnungen

3.2.1 Geoid/Ellipsoid Berechnung(Philipp, Jeremy)

3.2.2 Koordinatensystem Global to Local(Philipp, Jeremy)

3.2.3 Umwandlung GPS zu ECEF Koordinaten

(Philipp, Jeremy)

### 3.3 LibGDX

- 3.3.1 Einbindung LocationManager(Philipp)
- 3.3.2 Importieren von 3D Modellen(Philipp, Inga)
- 3.3.3 Drehung des Testgeländes Nordausrichtung(Philipp)
- 3.3.4 Positionierung des Spielerobjekts(Philipp)
- 3.3.5 Prototyp(Philipp)

### 3.4 Design

- 3.4.1 Erstellung 3D Modelle in 3D Max(Kristina)
- 3.4.2 Charakterdesign(Kristina)
- 3.4.3 Erstellung des App Menüs in Unity (Chantal)

## 4 Marketing und Präsentation

- 4.1 Logo (Jeremy)
- 4.2 Plakatdesign(Kristina, Jeremy)
- 4.3 Video(alle)

## 5 Zusammenfassung und Ausblick

- 5.1 Resümee der Projektarbeit (Inga)
- 5.2 Ausblick: Zoorino (Inga)
  - 5.2.1 Idee (Inga)
  - 5.2.3 Konzeptausblick (Jeremy)