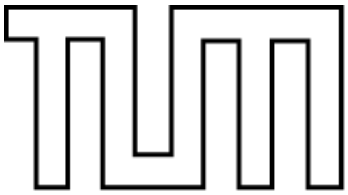


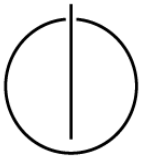
FAKULTÄT FÜR INFORMATIK

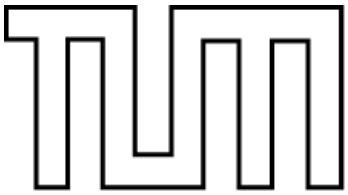
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Einsatz von Augmented Reality auf mobilen Endgeräten zur Visualisierung von Steuergerätefunktionen im Automotive Bereich

Maximilian Schüler





FAKULTÄT FÜR INFORMATIK

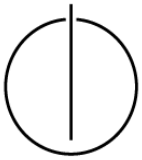
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Einsatz von Augmented Reality auf mobilen Endgeräten zur Visualisierung von Steuergerätefunktionen im Automotive Bereich

Use of augmented reality on mobile devices for visualization of electronic control unit functionality in the automotive domain

Bearbeiter:	Maximilian Schüler
Aufgabensteller:	Prof. Dr. Uwe Baumgarten
Betreuer:	M. Sc. Nils Kannengießer (TUM)
	Dipl.-Inf. Markus Reschka (BMW Group)
Abgabedatum:	15. Oktober 2012



Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Olching, den 15. Oktober 2012

_____

Maximilian Schüler

Zusammenfassung

Augmented Reality ist eine relativ junge Technologie mit hohem Potential. Auch im Kontext zur Erklärung von Steuergerätefunktionen im Automobil soll ermittelt werden, wie Augmented Reality als Hilfsmittel eingesetzt werden kann. Diese Masterthesis beschäftigt sich in diesem Zusammenhang mit verschiedenen Einsatzmöglichkeiten von Augmented Reality und beschreibt zusätzlich dessen Anwendung anhand eines Prototyps. Insbesondere wird dabei auf unterschiedliche Verfahren und deren Verwendung auf mobilen Endgeräten eingegangen. Aufgrund der begrenzten Ressourcen auf diesen Geräten und der vorhandenen Sensorik werden ausschließlich Verfahren und Werkzeuge beschrieben, die in diesem Rahmen ohne zusätzliche Hardware eingesetzt werden können. Der Ansatz zur Vereinigung von realer Manipulation der Fahrzeug-Aktuatorik und der optischen Präsentation virtueller Bestandteile durch Augmented Reality bildet dabei den Kern dieser Arbeit.

Abstract

Augmented Reality is a relative young technology with high potential. Regarding the context of explaining electronic control unit functionality in the automotive domain, one should analyze how Augmented Reality can be used as a technological facility. In this context, this Master's Thesis addresses different possible applications of Augmented Reality and, additionally, describes its applications by means of prototypes. Especially, different methods and their applications on mobile devices are elaborated. Due to the limited resources on these devices and the available sensors, only methods and tools are analyzed, which can be used without additional hardware. Essence of this work is the approach of merging the manipulation of physical actuators with the optical presentation of virtual components through Augmented Reality in the automotive domain.

Inhaltsverzeichnis

1 Thema und Hintergründe

Durch ständige Weiter- und Neuentwicklungen im Automobilsektor entsteht für Automobilhersteller zunehmend das Interesse die einzelnen Funktionen, die das Kraftfahrzeug bietet, zu erläutern, um somit bei potentiellen Kunden das Kaufinteresse zu wecken. Von BMW wurden beispielsweise Begriffe wie Dynamic Drive und Adaptive Drive geprägt. Dabei werden diese Begriffe meist nicht ausreichend durch das Unternehmen gegenüber dem Kunden erklärt. BMW bietet die Möglichkeit sich selbst über einzelne technische Aspekte und Funktionen, welche in den Fahrzeugen ihre Anwendung finden, zu informieren. In dem Techniklexikon, welches über das Internet eingesehen werden kann, ist es möglich einzelne Begriffe nachzuschlagen. Dabei handelt es sich allerdings um eine passive Art des Informationsaustausches, in welcher der Kunde dazu angehalten ist, aktiv nach den Erklärungen zu den einzelnen Funktionen zu suchen. Zusätzlich wird dabei Personen mit naturwissenschaftsfremden Hintergrund eine erste Hürde in Bezug auf das Verständnis gesetzt. Formulierungen wie, „Im Lenkstrang, der Lenkrad und Lenkgetriebe verbindet, sitzt ein Planetengetriebe. …“[BMWTL1] bieten bereits einen hohen Detailgrad und sind dem Verständnis der Gesamtfunktion des Teilsystems nicht zwingend zuträglich, bzw. für Leser ohne naturwissenschaftlichen Hintergrund nur schwer nachzuvollziehen. [BMWTL2][BMWTL3]

Bisher wurden von BMW Texte als auch Videofilme verwendet um Funktionen am Automobil dazustellen und dem Kunden zu erklären.[1] Für einige Konzepte, wie Dynamic Drive, existiert an dieser Stelle keine weitere Präsentationsform außer einem Artikel. Eine besondere Herausforderung stellen gerade Steuergeräte und Konzepte dar, die nicht unmittelbar erfahren werden können. Die Aktive Rollstabilisierung (ARS) beispielsweise, die genauer in Kapitel 4.2.3.3 behandelt wird, zeichnet sich dadurch aus, dass während der Fahrt kein Eingriff durch das System spürbar ist. Der Fahrer wird durch das System unterstützt, ohne dass dieser davon Kenntnis nimmt. Gerade diese Ausprägung des Systemverhaltens erschwert es dem Kunden die Notwendigkeit dieser Funktionen zu vermitteln.

Durch eine interaktive Präsentation der Steuergeräte im Fahrzeug kann die passive Art des Informationsaustausches zu einem aktiven Erlebnis erweitert werden. Dabei soll es dem Kunden ermöglicht werden, durch eigene Aktionen direkt den Effekt der Funktionen spürbar nachvollziehen zu können. Innovative Technologien sollen dabei auf den Kunden ansprechend wirken und das Interesse an dem Objekt der Schulung als auch an der Art und Weise des Lernens fördern. Auf diese Weise soll das Kaufinteresse eines Kunden verstärkt werden.

Diese Arbeit behandelt die Umsetzung einer innovativen Form der Darstellung von Systemfunktionen. Darüber hinaus wurde ein Demonstrator entwickelt, der auf dem BMW-Familientag 2012 bei BMW am 01.07.2012 internen Mitarbeitern und deren Familien vorgestellt wurde. Mittels dieses Prototyps konnten einigen Führungskräften die Vorteile dieser Form der Funktionserklärung erfolgreich vermittelt werden.

1.1 Begriffserläuterung und Hintergrundwissen

In dem weiteren Verlauf werden einige Begriffe verwendet, die an dieser Stelle definiert, bzw. erklärt, werden. Dazu zählen unter anderem Begrifflichkeiten aus dem Umfeld des Unternehmens BMW als auch Fachbegriffe aus den Themengebieten Computer Vision und mobile Entwicklungen.

[1] Beispielsweise zur Erklärung der Aktivlenkung, Siehe [BMWTECLX]

1.1.1 Android

Die Android Plattform ist eine quelloffene Entwicklung für mobile Endgeräte, die durch die Open Handset Alliance, eine Gruppe von 84 Unternehmen der Bereiche Technologie und mobile Endgeräte, unterstützt wird. Android ist die erste vollständige, offene und frei verfügbare mobile Plattform, die entwickelt wurde. [OHSALL]

Diese Plattform erlaubt es, Entwicklern anhand der öffentlich zugänglichen API ein tieferes Verständnis der Funktionsweise der Plattform zu erlangen und ermöglicht aufgrund der Barrierefreiheit bezüglich fehlender Lizenzgebühren innovative Entwicklungen schnell vorantreiben zu können. Ein weiteres Beispiel für die Barrierefreiheit von Android ist, dass zur Installation einzelner Anwendungen kein App-Store notwendig ist. Auch Software unbekannter Quellen kann installiert werden. [OHSFAQ]

Seit November 2007 wird die Android Plattform ständig weiterentwickelt und verfügt bereits über mehr als acht verschiedene Versionen. Zur Übersicht zeigt Abbildung 1 ein Diagramm, welches die Verbreitung der einzelnen Versionen auf Geräten grafisch darstellt. [GOBLOG][DEVAND]

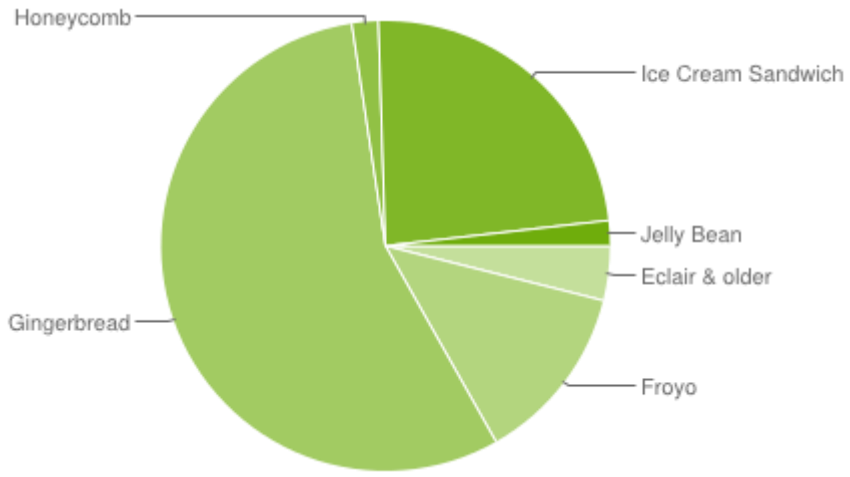


Abbildung 1 Verbreitung der einzelnen Android-Versionen, Stand: Oktober 2012 [DEVAND]

Die Android-Plattform basiert auf einem Linux-Kernel, verfügt über verschiedene C/C++-Bibliotheken und umfasst eine Android Laufzeitumgebung mit der *Dalvik Virtual Machine*. Darauf setzten weitere Schichten, wie das Applikationsrahmenwerk und die Applikationen auf, welche in der Programmiersprache JAVA verfasst wurden. Die *Dalvik Virtual Machine* ist ein Interpreter, der ausschließlich *Dalvik Executable*-Dateien ausführen kann, welche unter anderem bezüglich der Speichereffizienz optimiert sind. Insbesondere können JAVA-Klassen in dieses Format transformiert und anschließend durch den Interpreter ausgeführt werden. [DALVIK][ITSECA]

Für die Programm-Entwicklung sind lediglich das JAVA SDK und das Android SDK erforderlich. Ein Android Endgerät ist für die Entwicklung nicht zwingend erforderlich, da diese auch per Software simuliert werden können. Als besondere Technik im Unterschied zu beispielsweise dem iOS wird an dieser Stelle noch auf das *Android-Manifest* verwiesen, das den Inhalt einer App bezüglich der Zugriffsregeln und erforderlichen System-Berechtigungen auf verschiedene Ressourcen spezifiziert. Somit kann der Benutzer selbst einsehen, welche *Permissions*, bzw. Privilegien, von der App erfragt werden und kann dabei entscheiden, ob die App installiert werden soll oder nicht. [ITSECA]

Auf weitere Mechanismen im Umgang mit Android und auch auf verwendete Komponenten wird im Zuge dieser Masterthesis nur eingegangen wenn dies zwingend erforderlich ist. Dieses Dokument beschreibt nicht den Umgang mit Android während der Entwicklung im Detail, sondern dessen Einsatz im Projekt.

1.1.2 Augmented Reality

Frau Prof. Gudrun Klinker, Ph.D. beschreibt auf der Internetseite zum Lehrstuhl für Augmented Reality der Technischen Universität München den Begriff Augmented Reality wie folgt:

“Augmented Reality (AR) is a newly emerging technology by which a user‘s view of the real world is augmented with additional information from a computer model. With mobile, wearable computers, users can access information without having to leave their work place. They can manipulate and examine real objects and simultaneously receive additional information about them or the task at hand. Using Augmented Reality technology, the information is presented three-dimensionally integrated into the real world. Exploiting people’s visual and spatial skills to navigate in a three-dimensional world, Augmented Reality thus constitutes a particularly promising new user interface paradigm. “[CAMP]

Somit beschreibt Augmented Reality die Anreicherung der realen Welt um virtuelle Informationen, die zusätzlich auf einem Display eingeblendet werden und dabei die Realität um diese Informationen erweitern. Von Paul Milgram wurde ein Modell eingeführt, welches die reale Umgebung mit der virtuellen Umgebung in Verbindung setzt, um bei der Begriffsdefinition zu einem Konsens zu gelangen. Dabei beschreibt das Modell, welches als Reality-Virtuality (RV) Continuum bezeichnet wird, nicht nur Augmented Reality (AR) sondern auch Augmented Virtuality (AV). Welche dieser Formen im Einzelnen Fall vorliegt, wird innerhalb des Bereichs der Mixed Reality (MR) anhand des Ähnlichkeitsmaßes zur Realität bestimmt. Mit zunehmender Distanz zur Realität aufgrund des verstärkten Einsatzes von virtuellen Objekten, bzw. Bestandteilen, bewegt man sich von links nach rechts auf der in Abbildung 2 dargestellten horizontalen Linie des RV Continuums. Im weiteren Verlauf wird auf den Begriff der Augmented Virtuality nicht genauer eingegangen und wurde hier nur der Vollständigkeit halber genannt. [ARRVC]

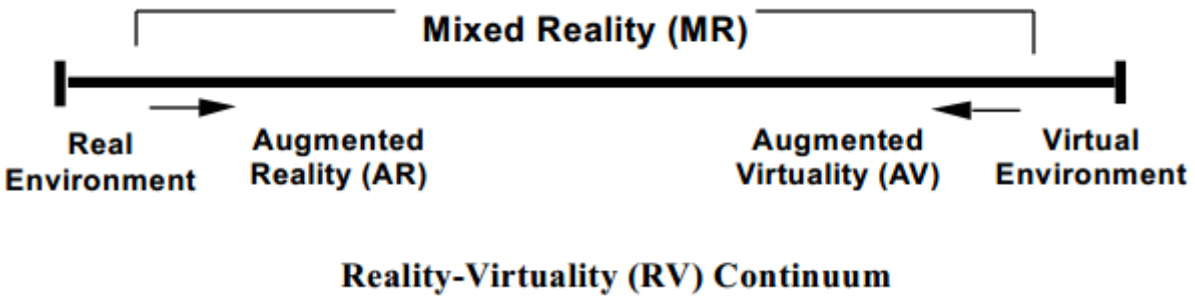


Abbildung 2 Reality-Virtuality (RV) Continuum, Zusammenhang zwischen Realität und Virtualität [ARRVC]

Für die Verwendung von Augmented Reality werden zur Darstellungen unterschiedliche Verfahren eingesetzt. Um Anwendungen von Augmented Reality genauer differenzieren zu können, werden grundsätzlich Anwendungen anhand der eingesetzten Displays unterschieden. Diese lassen sich in zwei Kategorien unterteilen. Wird als Anzeigegerät, bzw. Display, beispielsweise eine Brille eingesetzt, welche die zusätzlichen Informationen in das Sichtfeld des Trägers einblendet, so wird dies als „optical see-through“ AR Verfahren bezeichnet (optical ST). Eine solche Brille wird in diesem Zusammenhang als head-mounted display (HMD) bezeichnet, da diese direkt an den Kopf des

Benutzers angebracht wird und dessen Kopfbewegung den Beobachtungspunkt[2] verändert. Im Gegensatz dazu steht der Ansatz des monitor-based displays oder „window-on-the-world“ (WoW), auf das in dieser Arbeit nicht genauer eingegangen wird.[3] [ARRVC]

Zur Erklärung des Unterschieds kann man hier festhalten, dass bei einem monitor-based Verfahren das Display nicht am Benutzer angebracht ist und sich somit auch die Perspektive nicht durch dessen Bewegungen manipulieren lässt. Da für die Anwendung in der Masterarbeit ein Tablet einzusetzen ist, welches der Benutzer in beiden Händen hält, kann man hier von einer abgewandelten Form eines HMD sprechen. Somit wird in diesem Zusammenhang von einem ST AR System gesprochen. Im Unterschied zur erwähnten Brille, bei welcher allein die zusätzlichen Informationen generiert und eingeblendet werden, kann auch der Anteil der realen Welt künstlich eingeblendet werden. Bei diesem Ansatz handelt es sich um ein closed-view HMD, da der Benutzer keine direkte Sicht auf die reale Welt erhält, sondern diese komplett generiert wird. Zu diesem Zweck ist es notwendig, dass die Realität mittels einer Videokamera optisch aufgezeichnet wird. Erst die Kombination aus diesem Video und den zusätzlichen Informationen wird dem Benutzer eingeblendet, weshalb von einer Art video-see-through HMD gesprochen werden kann. [AZUMA]

Abbildung 3 stellt den Aufbau des video see-through HMD konzeptuell in einem Diagramm dar, welches in [AZUMA] verwendet wurde. Der Bestandteil des Head Trackers ist notwendig, um den Beobachtungspunkt der Kamera zu erhalten, der notwendig ist, um eine realistische Darstellung der virtuellen Elemente zu ermöglichen. Im Gegensatz dazu wird in der Anwendung im Demonstrator zu diesem Zweck nicht der Kopf des Benutzers, sondern das Tablet an sich verwendet. In dem Diagramm wird ersichtlich, dass die Daten der Videokamera, welche den Anteil der Darstellung der realen Welt beinhalten, zusammen mit den generierten virtuellen Bestandteilen aus dem *Scene generator* im *Video compositor* zusammengeführt werden. Die Kombination dieser beiden Anteile ergibt das Ergebnisbild, welches dem Benutzer auf dem Monitor angezeigt wird.

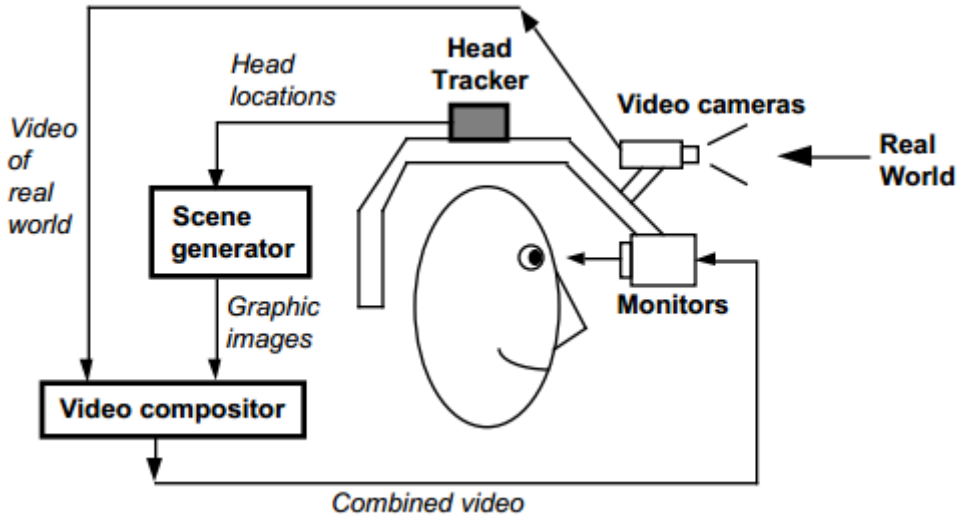


Abbildung 3 Konzeptionelles Diagramm eines video see-through HMD[AZUMA]

Für den zweiten Teil der Masterarbeit ist somit ein video see-through AR System umzusetzen. Dabei wird die reale Welt nicht unmittelbar dargestellt, wie es beispielsweise bei der Sicht durch eine Brille der Fall wäre, sondern vorher als Video eingelesen und unmittelbar wiedergegeben. Zur Darstellung

[2] Der Beobachtungspunkt, bzw. Eyepoint, stellt dabei den Ursprung des pyramidenförmigen Sichtbereichs dar.
[3] Siehe [ARRVC] Kapitel 5 für eine detaillierte Klassifizierung von MR anhand einer Taxonomie.

wird in der Masterarbeit als Anzeigegerät das Display des Tablets verwendet, welches als eine Art Fenster zur Betrachtung der AR Szene verwendet wird. Für die Aufnahme der optischen Daten wird die Kamera auf der Rückseite des Tablets eingesetzt. Dadurch ist die Darstellung bei dieser Anwendung abhängig von der Position und Orientierung des Tablets und damit auch abhängig von den Armbewegungen des Benutzers. Der Aufbau der AR Anwendung wird im Zusammenhang mit dem Anwendungsfall zur Niveauregulierung eines Fahrzeugs von BMW in Kapitel 5 detailliert beschrieben.

1.1.2.1 Tracking

Bereits im vorherigen Kapitel wurde angesprochen, dass es für die korrekte Darstellung von Zusatzinformationen notwendig ist, den Beobachtungspunkt der Kamera in Zusammenhang mit den Bilddaten zu ermitteln. Diese Funktionalität wird bei einer Anwendung von einem video see-through AR System, wie in Abbildung 3 dargestellt, durch den *Head Tracker* erbracht.

Um virtuelle dreidimensionale Objekte perspektivisch korrekt in dem Kontext von Bilddaten der realen Welt anzeigen zu können, ist es notwendig, dass die Pose der Kamera abhängig von diesen optischen Daten bestimmt wird. Die Pose der Kamera setzt sich zusammen aus der Position der Kamera als auch der Orientierung der Kamera im dreidimensionalen Raum. Die Position wird dabei anhand eines dreidimensionalen Vektors beschrieben, während die Orientierung der Kamera meist anhand eines Quaternions [4] definiert wird. Auf diese Weise werden alle Freiheitsgrade im dreidimensionalen Raum bezüglich der Translation und Rotation eindeutig festgelegt.

Ist die Pose der Kamera im dreidimensionalen Raum bekannt, ist dementsprechend auch die Perspektive auf das virtuelle Objekt bekannt. Mit Hilfe dieser Informationen können grafische Daten erzeugt werden, die das virtuelle Objekt darstellen, als wäre dieses real und aus der bekannten Perspektive von der Kamera erfasst, bzw. aufgenommen worden. Somit wirkt die Darstellung auf den Benutzer zumindest hinsichtlich der Perspektive natürlich und scheinbar real. Wird das Kamerabild zusammen mit den errechneten Bilddaten des virtuellen Objektes auf dem Display dargestellt, entsteht für den Benutzer der Eindruck, dass das virtuelle Objekt in der dargestellten Szene real existiert.

Der Begriff des Trackings bezeichnet in diesem Zusammenhang den Vorgang der Verfolgung eines Objektes im Raum, um dessen Position und Ausrichtung bestimmen zu können. Tracking-Verfahren können aufgrund der verwendeten Eingabedaten in verschiedene Kategorien eingeteilt werden. Im Verlauf der Masterthesis wird ausschließlich auf optische Verfahren des Trackings eingegangen, da die verwendeten mobilen Geräte nicht über die Sensorik, bzw. Hardware, verfügen, um andere Verfahren anwenden zu können. Alternative Tracking-Verfahren verwenden beispielsweise Ultraschall oder inertiale Sensorik, wie Gyroskope und Beschleunigungssensoren, um eine Bestimmung der Pose eines getrackten Objektes durchführen zu können. [VTFAR]

Für ein Tracking, welches rein auf optischen Daten basiert, ist lediglich eine Videokamera notwendig, welche die Eingabedaten liefert, aus welchen die Pose des getrackten Objektes errechnet werden kann. Grundsätzlich sind zu einer solchen Berechnung die kameraspezifischen Parameter notwendig. Dabei wird von den inneren und äußeren Kameraparametern gesprochen. Anhand einer Kamera-Kalibrierung müssen diese Werte zusammen mit den Parametern zur Beschreibung der Bildverzerrung, ermittelt werden: Durch die extrinsischen Parameter wird die Abbildung der Punkte

[4] Ein Quaternion besteht aus vier komplexen Einheiten und wird zur eindeutigen Beschreibung der Orientierung im Raum eingesetzt werden.

des dreidimensionalen Weltkoordinatensystems auf das dreidimensionale Kamerakoordinatensystem beschrieben. Intrinsische Parameter beschreiben den Zusammenhang zwischen den dreidimensionalen Kamerakoordinaten und den zweidimensionalen Pixelkoordinaten der Bildebene. Die Kenntnis dieser Parameter ist entscheidend für die Genauigkeit bei einem optischen Tracking-Verfahren. Aus diesem Grund wird meist vor dem Einsatz einer Kamera für optisches Tracking eine Kalibrierung vorausgesetzt, welche die entsprechenden Parameter liefert. Beispielsweise wird durch die ARToolKit[5] Bibliothek ein Verfahren zur Verfügung gestellt, um diese Parameter für eine beliebige Videokamera zu ermitteln. [ARTOOL]

Im Rahmen dieser Arbeit wird lediglich auf optisches Tracking eingegangen. Aufgrund des Umfangs werden nicht-optische Trackingverfahren, als auch IR-Tracking Verfahren, von einer Beschreibung ausgeschlossen. An dieser Stelle wird auf einschlägige Fachliteratur verwiesen. Teil der Arbeit sind ausschließlich Verfahren, die mittels der zur Verfügung gestellten Hardware eingesetzt werden können.

1.1.2.2 Feature Tracking

Eine Form des Trackings von Objekten anhand von Bilddaten wird vom Feature Tracking abgedeckt. Ein Feature ist ein markanter Bildpunkt, welcher entsprechende Eigenschaften hat, die es ermöglichen diesen Punkt, bzw. Bereich, in nachfolgenden Bildern eines Videos wieder zu erkennen. Im Unterschied zu anderen optischen Verfahren ist beim Feature Tracking kein Referenzmodell notwendig, über das bereits Informationen vorliegen. Um Features, bzw. Merkmale, im Bild extrahieren zu können, wurden bereits unterschiedliche Algorithmen entwickelt, die dies ermöglichen. Beispiele solcher Algorithmen sind KLT/SIFT oder SURF. [KLTSIFT][FTSURF].

Mit Hilfe dieser Algorithmen lassen sich Bildmerkmale extrahieren und die relative Bewegung dieser Feature zwischen den einzelnen Bildern berechnen (Optischer Fluss). Der optische Fluss von Features wird in Abbildung 4 (links) dargestellt. Somit kann die relative Posen-Änderung der Kamera bestimmt werden. Um Kenntnis über die absolute Pose zu gewinnen, muss bei einem solchen Verfahren eine Feature Map[6] erstellt werden, die zu Beginn als Bezugspunkt dient. Aufgrund der immer höheren Leistung von Rechnern werden heute bereits Verfahren wie PTAM[7] eingesetzt, welche die Feature Map zur Laufzeit des Trackings dynamisch anpassen und erweitern, um genauere Ergebnisse zu erzielen. [ARPTAM]

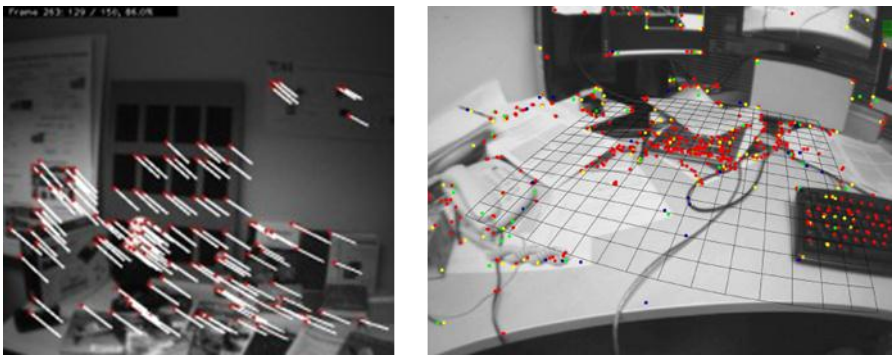


Abbildung 4 Optischer Fluss der Feature (links) [IMUKLT], Feature-Erkennung mittels PTAM (rechts) [ARPTAM]

[5] ARToolKit ist eine Software-Bibliothek zur Erstellung von AR Anwendungen.
[6] Ein Modell, das die räumliche Umgebung und die optischen Merkmale in Verbindung setzt. Siehe [VTFAR]
[7] PTAM, Abk. für Parallel Tracking and Mapping

1.1.2.3 Marker Tracking

Wie bereits erwähnt ist es bei einem Feature Tracking nicht notwendig, optische Hilfsmittel in der Umgebung bereitzustellen, damit ein Tracking durchgeführt werden kann. In diesem Punkt besteht der Hauptunterscheid zum Marker Tracking. Hier wird als Hilfsmittel ein Marker eingesetzt, der sich vorzugsweise stark optisch von der Umgebung abhebt. Zusätzlich sind Informationen über die Geometrie dieses Markers der Anwendung a priori bekannt. Bei einem Marker handelt es sich in den gängigen Anwendungen meist um eine optische Kodierung einer Ganzzahl, die mittels schwarzer und weißer Flächen gleicher Größe binär dargestellt wird. Abbildung 5 stellt einen solchen Marker der Firma Metaio mit ID 1 dar. Jeder Marker wird von seiner Umgebung mittels eines breiten weiß/schwarz/weiß Rahmens abgegrenzt. Wichtig dabei ist, dass die Größe des Markers mit 60 Millimetern bekannt ist. Der optische Aufbau der Kodierung ist entscheidend für ein robustes Tracking. So sollten beispielsweise keine symmetrischen Kodierungen gewählt werden, da sich somit die Pose des Markers im Raum, abhängig von der Kameraansicht, nicht eindeutig bestimmen lässt. Aus diesem Grund wurden von Metaio bereits markante Marker festgelegt. An dieser Stelle wird nicht weiter auf die Kriterien zur Wahl von Markern eingegangen. Allgemein beschäftigt sich [ARGFTT] mit der Frage der Beschaffenheit von optimalen Features für ein Tracking. Insbesondere Marker können als Bildbereich, bzw. Features, betrachtet werden.

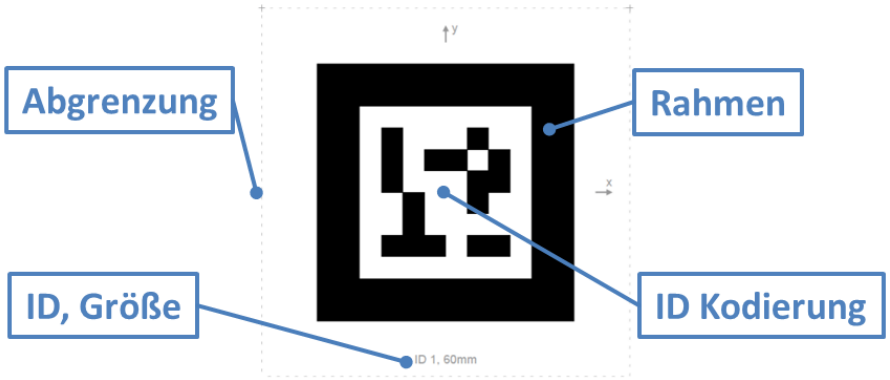


Abbildung 5 Metaio ID-Marker mit ID 1 in 60 mm

1.1.2.4 Markerless Tracking

Das entscheidende Merkmal eines Markerless Tracking Verfahrens ist die Tatsache, dass keine künstlichen optischen Hilfsmittel in der realen Umgebung angebracht werden müssen, die dem optischen Verfahren als Referenzpunkte dienen. Dabei lassen sich diese Verfahren in zwei Kategorien unterteilen:

Bei der modellbasierten Methode wird bereits a priori Wissen über die Umgebung vorausgesetzt. Dies kann in unterschiedlichen Formen vorliegen. Beispielsweise mittels eines digitalen Referenzmodells oder anhand von Texturen eines Bereichs der Szene, welche getrackt werden soll. Ein digitales Referenzmodell kann dabei ein 3D Modell sein, welches verwendet wird, um anhand eines Vergleichs mit den Bilddaten auf die Pose der Kamera schließen zu können. Dabei werden beispielsweise von dem virtuellen Referenzmodell einzelne Kontrollpunkte entlang einer Kante auf das aufgenommene Bild projiziert und anhand eines Matching-Algorithmus verglichen, um Korrespondenzen zu finden. Dieser Vorgang wird in Abbildung 6 dargestellt. In der Darstellung ist die Vergleichskante in roter Farbe abgebildet. Anschließend kann mit Hilfe der Ausgabe dieses Algorithmus die Pose der Kamera abgeschätzt werden. [AR3DML]

Eine andere Technik beruht darauf, dass anhand der Bewegung der Kamera während der Aufnahme die Topologie der Umgebung abgeschätzt werden kann. Diese Methode wird als Structure from Motion (SfM) bezeichnet. Diese Form des Trackings erfordert eine höhere Rechenleistung und ist oft komplexer als der modellbasierte Ansatz. Im Gegensatz zu dem modellbasierten Ansatz wird bei dieser Methode kein Wissen über die Umgebung, wie beispielsweise die Kenntnis eines Referenzmodells, vorausgesetzt. [ARMTSW]

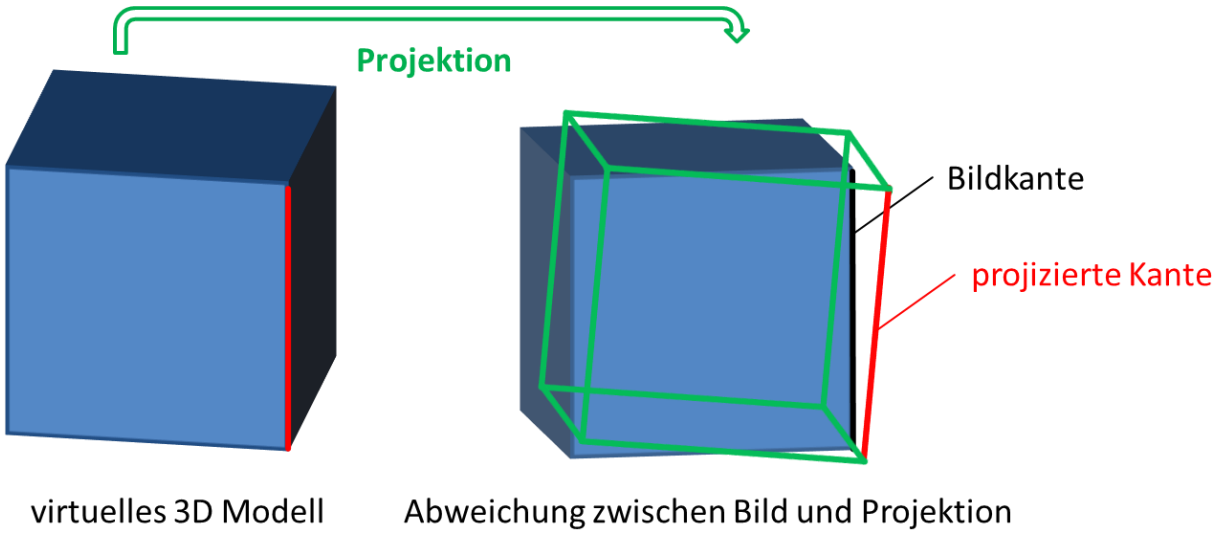


Abbildung 6 Konzeptzeichnung von Projektion und Matching einer Kante des 3D Modells

Bisher wurden alle notwendigen Begriffe erläutert, welche für das Verständnis der Masterthesis erforderlich sind. Für eine umfassendere Auflistung verschiedener Tracking-Verfahren, auch außerhalb der optischen Verfahren, wird auf [FRIEMT] verwiesen. Diese detaillierte Begriffsbetrachtung bekommt in Kapitel 5, das sich mit dem Thema Augmented Reality und Tracking befasst und deren Anwendung in der Masterarbeit erklärt, eine besondere Bedeutung. Im nächsten Kapitel 1.2 wird die Fragestellung der Masterthesis in das Gesamtthema eingeordnet, um zu vermitteln, welche Bereiche von dieser Arbeit im Einzelnen thematisch abgedeckt werden müssen.

1.2 Einordnung der Fragegestellung ins Gesamtthema

Um Erklärungen von Funktionen am Automobil auch bei komplexen oder verdeckten Abläufen zu unterstützen, ist es vorteilhaft, den Lernenden die Möglichkeit zu bieten, durch eigene Interaktion Bereiche zu beleuchten, die beispielsweise durch ein Video nicht abgedeckt werden. Durch ein Video ist die Perspektive für den Betrachter fest vorgegeben. Auch werden die Informationen über das vorgestellte System in einem festen Ablauf präsentiert. Somit ist es nicht möglich Informationen über einzelne Bereiche gezielt nach Bedarf abzufragen.

In der vorliegenden Abschlussarbeit werden Einsatzmöglichkeiten von innovativen Technologien diskutiert, welche Verbesserungen der eben genannten Nachteile herkömmlicher Methoden nach sich ziehen. Ein Beispiel ist unter anderem der Einsatz von Augmented Reality zur Darstellung von für die Funktionserklärung relevanten Bauteilen im Fahrzeug. Mittels AR kann durch den Benutzer die Perspektive auf das betrachtete Objekt beeinflusst werden. Durch diese Methode kann der Kunde frei von Einschränkungen der räumlichen Darstellung Einfluss auf den Präsentationsverlauf nehmen und am realen Automobil eigenständig die Funktionsweise entdecken.

Um die Inhalte den Kunden vermitteln zu können, wurden als Medien tragbare Rechner gewählt, die im weiteren Verlauf auch als Tablets bezeichnet werden. Gerade aufgrund der steigenden Popularität von Tablet-PCs wird diese Form der Hardware als Präsentationsmedium verwendet. Dabei spielt die Vertrautheit der Kunden im Umgang mit einem solchen Gerät eine entscheidende Rolle. Der Benutzer muss nun selbst mit der Handhabung des Tablets vertraut sein, um die Steuerung eigenständig übernehmen zu können. Eine intuitive Steuerung der Präsentation ist dabei immens wichtig, um erste Barrieren bei der Benutzung zu verhindern. Aufgrund eines Touch-Displays kann auf Berührungen des Benutzers reagiert werden.

Im Zuge der Masterthesis wurde ein Demonstrator entwickelt, der verschiedene Technologien vereint, um die Funktion verschiedener Steuergeräte zu vergegenwärtigen. Für die Demonstration und Validierung dieses Prototyps wurde der BMW-Familientag gewählt. Am BMW-Familientag wurde einem breiten Publikum, bestehend aus unterschiedlichen Altersklassen und Personen mit verschiedenem professionellem Hintergrund, der aktuelle Stand des Demonstrators präsentiert. Wichtig für den BMW-Familientag war dabei besonders die Umsetzung der Anwendungsfälle zur Erklärung der Aktiven Rollstabilisierung, die in Kapitel 4.2.3.3 detailliert beschrieben werden. Aus diesem Grund wurden mehrere Anwendungsfälle für die Vermittlung der ARS Funktionalität eingeplant und mussten bis zum 01.Juli.2012 implementiert und getestet werden.

Durch den Umfang der Implementierungsaufgaben im Hinblick auf den Familientag wurde das Thema Augmented Reality bis zum genannten Termin lediglich angerissen. Gerade der Kontext des Familientags, bezüglich des Standortes zur Präsentation, erschwerte die Anwendung von optischen Verfahren als Grundlage für den Einsatz von Augmented Reality im Vorhinein. Optimale Umstände für die Verwendung dieser Verfahren werden in Kapitel 5.1.2 diskutiert.

Erst im weiteren Verlauf der Masterarbeit konnte genauer auf das Thema AR eingegangen werden. Der entwickelte Demonstrator behandelt demnach aktuell keine Bereiche der Augmented Reality. Dieses Thema wird im zweiten Teil der Masterthesis im Kapitel 5 beschrieben und kann als Grundlage für weitere konsekutive Abschlussarbeiten eingesetzt werden. Dazu wurden in der zweiten Hälfte des Bearbeitungszeitraums alle Funktionen implementiert und getestet, die für den Einsatz von AR, gemäß den Anforderungen der BMW, notwendig sind, um in den Demonstrator integriert zu werden. Anhand dieses zweiten Prototyps wurde bereits der „proof-of-concept“ erbracht, um das gewählte Verfahren für den vorgegebenen Einsatzzweck von BMW zu verwenden.

Aufgrund des Umfangs der verschiedenen Anwendungsfälle als auch der Rahmenbedingungen, die durch die Schnittstelle zum Server bereits vorgegeben waren, konnte der ursprünglich geplante Umfang an Anwendungsfällen nicht umgesetzt werden. Organisatorische Aufgaben, die ständige Weiterentwicklung der Serverseite als auch der Fahrzeugzugang zu bestimmten Steuergeräten auf der physikalischen Ebene, stellten dabei bezüglich des Zeitaufwands eine besondere Herausforderung dar.

Anhand des nächsten Kapitels werden die bisherigen Anwendungsgebiete von Augmented Reality bezüglich der verschiedenen Disziplinen betrachtet und dazu einige Beispiele aufgezählt.

1.3 Spektrum des Einsatzes von Augmented Reality

Die Anwendung von Augmented Reality zur flexiblen Darstellung von Informationen wird bereits in unterschiedlichen Disziplinen erfolgreich eingesetzt. Im weiteren Verlauf werden einige Beispiele aus den verschiedenen Domänen dieser Anwendungen gegeben, um einen ersten Eindruck zu erhalten welche Möglichkeiten durch Augmented Reality eröffnet werden.

Medizin

In der Medizintechnik wird Augmented Reality bereits zur Darstellung der Anatomie eines Patienten eingesetzt. In Abbildung 7 links sind beispielsweise die Knochen des Patienten mittels Augmented Reality mit dem Originalbild des Beins überlagert dargestellt. [TUMAR2]

Chemie

In der Domäne der Chemie ist es mittels Augmented Reality möglich, komplexe dreidimensionale Strukturen, wie etwa große Moleküle, durch drehen eines Würfels von allen Seiten zu betrachten. Während der Manipulation der Würfelpose ist es lediglich notwendig, dass durch die Hand des Benutzers der Würfel, bzw. dessen Marker, nicht abgedeckt werden. [TUMAR1]

Luftfahrt

Für Piloten von militärisch als auch bei zivil verwendeten Luftfahrzeugen wurde bisher Augmented Reality eingesetzt um diese mit Zusatzinformationen zu versorgen. Im militärischen Bereich wurden dabei Anwendungen entwickelt, die den Piloten bei einem Luftkampf, der Navigation als auch der Zielerfassung unterstützen. Ein Beispiel einer solchen Entwicklung ist das Head Up Display (HUD) von Lockheed Martin für die F-16 C. Dabei werden zusätzliche Informationen auf der Cockpitscheibe des Piloten dargestellt. In anderen militärischen Flugzeugen werden Informationen auch in einem Helmet-Mounted Display direkt auf dem Helmvisier des Piloten angezeigt. Auch bei Transportflugzeugen und zivilen Strahlflugzeugen wie beispielsweise der Boeing 737-832 wird mittels eines Head Up Display der Pilot über wichtige Parameter informiert. [FRMADER]

Ebenso werden an Flugplätzen im Bereich des Towers AR Systeme eingesetzt um unabhängig von der aktuellen Wetterlage und Helligkeit die Standorte der Luftfahrzeuge visuell ermitteln zu können. Bei dieser Form eines AR Systems wird von einem Augmented Reality Tower Tool (ARTT) gesprochen. In diesem Bereich wurden bereits in den 80er Jahren erste Konzepte erarbeitet und von der NASA 1998 ein Prototype mit dem Namen Situation Awareness Virtual Environment (SAVE) vorgestellt. [ARNASA]

Industrielle Anlagen

In industriellen Anlagen werden AR Systeme unter anderem dazu eingesetzt Diskrepanzen, bzw. Abweichungen, beim Anlagenbau aufzudecken. In Abbildung 7 rechts wird eine Anlage in diesem Zusammenhang dargestellt. [ARINDU]

Zudem werden AR Systeme im Bereich der Wartung von Anlagen als auch der Produktentwicklung eingesetzt. Ein Beispiel dafür ist das entwickelte Testszenario im Rahmen des EU-Forschungsprojekts AMIRE. Dabei werden Mitarbeiter bei der Lokalisierung von Anlagenteilen zur Wartung komplexer Anlagen unterstützt. [ARCLAB]

An dieser Stelle wird zusätzlich an das ARVIKA Projekt verwiesen, welches durch ein Konsortium aus verschiedenen Firmen der Industrie, darunter BMW, und Einrichtung des Bildungswesens, wie auch der Technischen Universität München, getragen wurde. Das Projekt erforschte und realisierte AR-Technologien für den Einsatzzweck in Arbeitsprozessen der Entwicklung, Produktion und dem Service. [ARVIKA] Weitere verschiedene Anwendungsfälle von AR Systemen im industriellen Umfeld sind auch zu finden in [ARVIKAb].

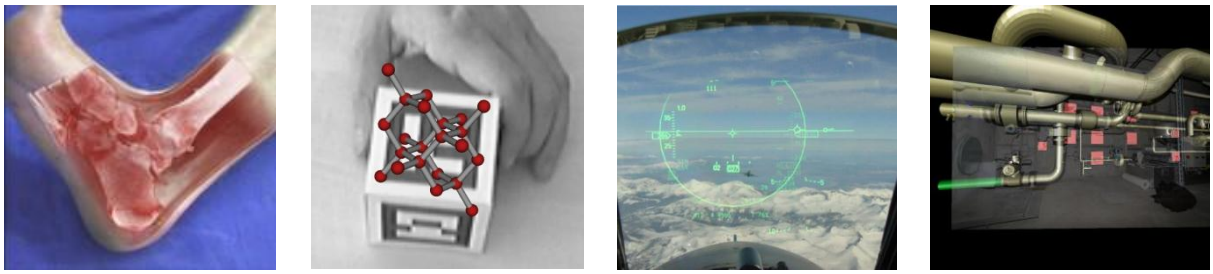


Abbildung 7 Anwendungen von AR in verschiedenen Domänen, (von links nach rechts) Medizin, Chemie, Avionik und Anlagenbau [TUMAR2] [TUMAR1] [FRMADER] [ARINDU]

Sport

Bei Übertragungen von Sportereignissen wird Augmented Reality eingesetzt, um virtuelle Inhalte in die Übertragungen einzublenden. Bei Fußballübertragungen im Fernsehen wird häufig zur Unterstützung der Zuschauer mittels einer virtuellen Linie die Abseitsposition angezeigt oder etwa der Abstand zwischen dem Schützen und der Mauer bei einem Freistoß dargestellt. [GUFAR1]

Automobilindustrie

Bei der BMW Group wird Augmented Reality bereits in der Personalgewinnung, bzw. Personalmarketing, eingesetzt, um Fachkräfte mittels innovativer Anwendungen zu begeistern und zudem einen breiteren Informationskanal bereitzustellen. [HORNET1] Auch bei der Wartung wird von BMW Augmented Reality eingesetzt, um dem Personal Wartungsvorgänge bezüglich Ein- und Ausbau von Bauteilen zu erleichtern. Dabei wird eine Art Bauanleitung zum jeweils aktuellen Bearbeitungsschritt in der Datenbrille des Mechanikers eingeblendet. [BMWSV1]

Bisher noch nicht beleuchtet ist der Einsatz von Augmented Reality zum Kundenerleben für komplexe Fahrzeugfunktionen in Steuergeräten. In diesem Umfeld leitet das nachfolgende Kapitel unter anderem die Aufgabenstellung dieser Arbeit ab und grenzt diese ein.

2 Projektbeschreibung und Aufgabenstellung

Der Projektumfang für den BMW-Familientag und damit die Aufgabenstellung für die vorliegende Masterthesis werden in diesem Kapitel detailliert beschrieben. Um ein Verständnis für die Motivation dieser Arbeit zu erlangen, wird zudem auf Missstände bezüglich der Erklärung von Steuergerätefunktionen eingegangen.

Das folgende Kapitel befasst sich mit der Problematik von BMW, die als Anlass für das Thema dieser Masterarbeit dient. Insbesondere werden dabei Funktionen von Steuergeräten erklärt, die für den Demonstrator am BMW-Familientag eingesetzt wurden.

2.1 Projektauslöser bei BMW und Motivation

Durch das Voranschreiten der technologischen Entwicklungen wird zunehmend Aktuatorik und Sensorik in modernen Fahrzeugen verbaut, um kontextsensitiv den Fahrer mit wichtigen Daten für die Fahrzeugführung versorgen zu können, bzw. den Fahrkomfort und die Sicherheit zu steigern. Die Anzahl an Steuergeräten nimmt somit ständig zu und formt immer komplexere Systeme, die für den Kunden letztlich unter Umständen nur noch in geringen Zügen verständlich sind. Während früher ein PKW fast ausschließlich aus mechanischen Komponenten bestand, werden heute verstärkt elektrotechnische Komponenten eingesetzt. Auch die Anzahl an Quellcodezeilen und damit der Softwareanteil im Automobil nimmt immer weiter zu. Während in einem F-22 Raptor[8] circa 1,7 Millionen Programmzeilen, bzw. Lines of Code (LoC), an Software vorhanden sind, werden in modernen PKWs der Premium-Klasse heutzutage ungefähr 100 Millionen LoC eingesetzt. Grund dafür ist die Anzahl der Steuergeräte, die sich in aktuellen Modellen auf 70 bis 100 Stück beläuft und deren Vernetzung über verschiedene Bussysteme realisiert wird. Im Jahre 1977 belief sich diese Zahl noch auf ein einziges Steuergerät, wie das Beispiel des Oldsmobile Toronado von General Motors zeigt. [IEEESP]

Allein bei der Bedienungsanleitung für Kunden zu einem modernen BMW beläuft sich die Seitenzahl auf mehrere Hundert Stück. Beispielsweise hat eine BMW 528i Limousine aus dem Jahr 2012 335 Seiten.[9] Dabei handelt es sich natürlich nicht um die Beschreibung der Steuergeräte an sich, sondern vielmehr der Funktionen am Automobil. Damit erhält man ein grobes Verständnis dafür, welchen Beitrag diese Systeme in einem modernen PKW leisten müssen. Bei dieser Menge an Informationen ist die Funktionsweise einzelner Teilsysteme in dem komplexen Gesamtsystem des Premiumwagens für Kunden teilweise nicht erkennbar. Dieser Umstand wurde von Mitarbeitern der BMW als ein aktuelles Problem bezeichnet. Zur Problemlösung wurde das Thema dieser Masterarbeit ausgeschrieben. Durch diese Arbeit soll besonders für bestimmte Steuergerätefunktionen eine ansprechende Form der Erklärung gefunden werden. Bisher wurden zur Erklärung von Steuergerätefunktionen lediglich Videos oder kurze schriftliche Erklärungen von BMW für den Kunden eingesetzt. In dem BMW Techniklexikon können so einzelne Begriffe nachgeschlagen werden. Dabei wird außer der Funktionswirkung meist kein technischer Hintergrund angeboten oder zu detailliert auf einzelne Bereiche eingegangen.

Die Funktionsweise der ARS wird im Zusammenhang mit Dynamic Drive erläutert, wobei nicht auf das einzelne Steuergerät, bzw. Teilsystem, eingegangen wird, sondern das Konzept Dynamic Drive in

[8] Lockheed Martin F-22 Raptor, Kampfflugzeug der U.S. Air Force

[9] Siehe [BMWUSA] für Bedienungsanleitungen zu BMW Modellen des amerikanischen Marktes die frei zugänglich sind

seiner Gesamtheit beschrieben wird. Ein Video, welches die Funktionsweise genauer erklärt, wird vom BMW Techniklexikon nicht angeboten. [BMWTL2]

Das ARS ist ein Teil des aktiven Fahrwerkregelungssystems Dynamic Drive, welches dazu dient die Wankbewegungen des Fahrzeugs in Kurven und bei schnellen Ausweichmanövern auf ein Minimum zu reduzieren. Zu diesem Zweck werden an beiden Achsen Stabilisatoren eingesetzt, die entgegen dem Trägheitsmoment M_x, um die Längsachse des Fahrzeugs, ein Ausgleichsmoment $M_{ARS,x}$ erzeugen um die Wankbewegungen auszugleichen. In Abbildung 8 werden die relevanten Einflussfaktoren dargestellt. Zusätzlich wird eine Wankrantenverteilung zwischen Vorder- und Hinterachse durchgeführt, um differenziert auf unterschiedliche Fahrsituationen eingehen zu können. Als Eingabeparameter werden Werte der Beschleunigungs- und Positionssensoren erfasst, um in Echtzeit auf die Fahrsituation bezüglich der Querbeschleunigung a_y reagieren zu können und die Aktuatorik entsprechend ansprechen zu können. [BMWTL2][BMWARS]

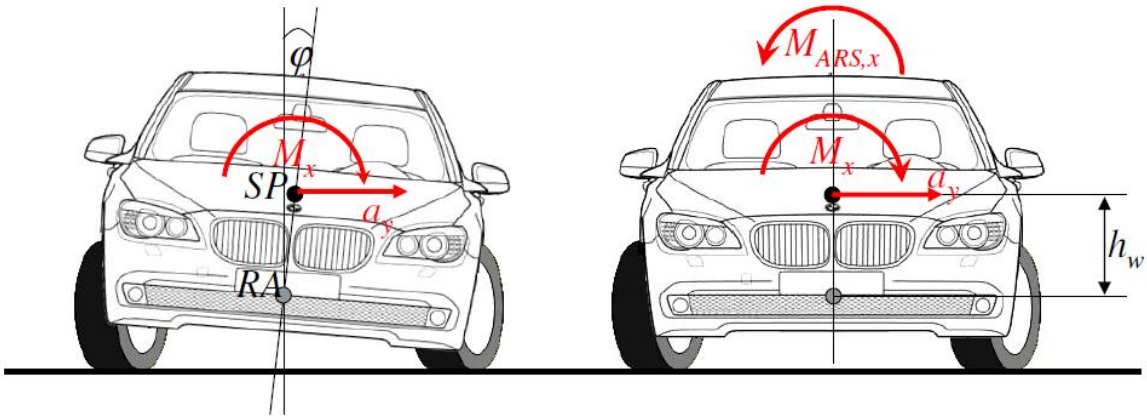


Abbildung 8 Einflussfaktoren bei einem Fahrzeug ohne ARS (links) und mit ARS (rechts) während der Kurvenfahrt [BMWARS]

Für eine detaillierte Beschreibung des ARS im Zusammenhang mit der Kommunikations-Komponente wird auf [BMWRTD] verwiesen. Eine exakte Erläuterung der Vorgänge im Fahrzeug bezüglich der Mechanik befindet sich in [BMWARS].

Der Umstand, dass dieses System nicht unmittelbar für den Fahrzeughalter spürbar ist, erweckt unter Umständen beim Kunden den Eindruck, dass durch das System keine Veränderung zugunsten des Fahrverhaltens entsteht. Um den Zusammenhang zwischen dem Einsatz dieses Steuergeräts und der Fahrsituation zu vergegenwärtigen, wurde ein Anwendungsfall im Demonstrator integriert. In Kapitel 4.2.3.3 wird dieser zusammen mit der Umsetzung im Demonstrator erklärt.

Auf die Steuergeräte für die anderen beiden Anwendungsfälle wird bezüglich der technischen Hintergründe nicht näher eingegangen. Beschrieben werden deren Nutzen und die Umsetzung der Funktionserklärung bezüglich ihrer Wirkung. Die Erklärungen zum ARS wurden hier lediglich als Beispiel gewählt, um die Problematik des Themas zu verdeutlichen. Die Abschlussarbeit [BMWRTD] von Robert Trampler beschäftigt sich auf technischer Ebene detaillierter mit den Steuergeräten der genannten Anwendungsfälle. Dabei wird die Umsetzung zur freien Manipulation der Aktuatorik erläutert, um die Steuergerätefunktionen anschaulich erklären zu können.

Zur Verbesserung der Präsentation von Steuergerätefunktionen ergibt sich die folgende Aufgabenstellung, dieser Masterthesis, die in dem nachfolgenden Kapitel diskutiert wird.

2.2 Aufgabenstellung und Abgrenzung

Um ein Verständnis für den Aufgabenumfang zu schaffen, wird im folgenden Kapitel die Aufgabenstellung detailliert erläutert. Zudem wird in einem weiteren Abschnitt das Thema dieser Masterthesis von den anderen Anteilen abgegrenzt, die für die Umsetzung des Prototyps notwendig waren, aber nicht Bestandteil dieser Ausarbeitung sind.

2.2.1 Aufgabenstellung

Im Zuge der Masterarbeit sollte für die Erklärung einzelner Steuergerätefunktionen im Automobil, insbesondere am Beispiel des BMW F11[10], eine Anwendung entwickelt werden, die auf innovativen Technologien beruht und am BMW-Familientag vorgestellt werden konnte. Die Anwendung sollte dabei unter anderem AR auf einem mobilen Endgerät zum Einsatz bringen. Aufgrund der umfangreichen Arbeiten im Hinblick auf den Familientag konnte diese Anforderung erst nach dem Familientag in der zweiten Hälfte des Bearbeitungszeitraums erfüllt werden. Dabei wurde zu Beginn des Projekts der Aufwand bezüglich der zu leistenden Arbeiten von der Betreuung bei BMW erheblich unterschätzt. Somit sind Bestandteile von AR im herkömmlichen Sinne nicht in dem Demonstrator enthalten.

Grundsätzlich konnten die Anforderungen von BMW grob mittels einiger Anwendungsfälle festgehalten werden, wobei zu diesem Zeitpunkt noch unklar war, welche dieser Anwendungsfälle letztendlich für den Familientag umgesetzt werden. Entscheidend bei der Realisierung der einzelnen Fälle waren zum Beispiel einerseits die Machbarkeit, als auch der Aufwand, der für die Entwicklung und organisatorische Arbeiten betrieben werden musste.

Für die Entwicklung der Anwendung sollte eine Android App implementiert werden, welche die entsprechenden Anwendungsfälle abdeckt. Diese App sollte auf einem Tablet-PC ausgeführt werden und die Schnittstelle zu einer Kommunikations-Komponente verwenden, um indirekt die Aktuatorik des PKW manipulieren zu können. Die Kommunikation zwischen der Kommunikations-Komponente und dem Tablet-PC sollte dabei über Funk, und somit kabellos, stattfinden.

Im weiteren Verlauf werden die Anwendungsfälle beschrieben, die als Grundlage für die Arbeiten dieser Masterarbeit verwendet wurden. Des Weiteren werden alle Anforderungen aufgeführt, die in diesem Zusammenhang und der Rücksprache mit dem Betreuer bei BMW im Rahmen einer Analyse aufgedeckt wurden. Zusätzlich wurden die Analyseergebnisse aufgrund der bereits vorhandenen Kommunikations-Komponente und deren Weiterentwicklung während der Bearbeitungszeit im Verlauf der Arbeit zunehmend detailliert.

2.2.1.1 Umgesetzte Anwendungsfälle

Für den BMW-Familientag sollten die folgenden Teilsysteme, bzw. Steuergeräte, im Zusammenhang mit ihrer Funktion im Gesamtautomobil erklärt werden.

1. Aktivlenkung

   Mittels der Aktivlenkung werden bei unterschiedlichen Geschwindigkeiten des Fahrzeugs die Lenkbewegungen des Fahrers entsprechend der Situation angepasst. Somit ist die effektive Auslenkung der Reifen variabel und von der Geschwindigkeit des Fahrzeugs abhängig. Auch die Auslenkung der Hinterachse wird von dieser Regelung beeinflusst. [BMWTL1]

[10] bezeichnet die unternehmensinterne Baureihenbezeichnung des BMW 5er Touring (Kombivariante) seit 2010

2. Wankstabilisierung

   Fährt das Fahrzeug mit erhöhter Geschwindigkeit in eine Kurve ein, erfährt es durch die Fliehkräfte eine Rollbewegung entlang der Längsachse. Durch die Wankstabilisierung wird dieser Bewegung aktiv entgegengewirkt. Der Effekt ist, dass sich das Automobil aktiv entlang der Längsachse entgegen der Fliehkraft neigt und somit die Rollbewegung ausgleicht. [BMWTL2]

3. Niveauregulierung

   Bei schwerer Beladung (Kofferraum) erfährt das Fahrzeug eine Absenkung im Bereich der Hinterachse. Mittels der Niveauregulierung wird der Fahrzeug-Höhenstand an der Hinterachse durch die Luftfederung ausgeglichen. Auch Schiefbeladung kann anhand von Einzelrad-Regelung der Federung ausgeglichen werden. [BMWTL4]

Die Umsetzung der jeweiligen Anwendungsfälle 1 und 2 wird in Kapitel 4 erklärt. In diesem Kapitel werden auch die erarbeiten Konzepte und deren Implementierung detailliert erläutert. Die Realisierung des Anwendungsfalls 3 wird dagegen in Kapitel 5 beschrieben.

Im folgenden Kapitel wird auf die Anforderungen von Seiten der BMW für die Durchführung des Projekts eingegangen.

2.2.1.2 Funktionale Anforderungen

Funktionale Anforderungen an das System waren zu großen Teilen im Vorherein nicht bekannt, da es zu Beginn der Masterarbeit noch keine klaren Vorstellungen zum Systemverhalten gab. Zu den Anwendungsfällen passende Use-Cases in Bezug auf die Nutzung des Systems waren dabei noch nicht vorgegeben. Lediglich ein logisches Datenmodell in Bezug auf die Kommunikation zwischen der Kommunikations-Komponente und der Tablet-Anwendung war bereits größtenteils vorhanden. Das Datenmodell wird in Kapitel 5.3.1 in [BMWRTD] erläutert. Allerdings wurde dieses Modell noch während der Entwicklung angepasst und weiterentwickelt. Ein Prozessmodell bezüglich der Kommunikation war zu diesem Zeitpunkt lediglich informell vorhanden. An die Benutzerschnittstelle, bzw. Benutzeroberfläche, wurden zu diesem Zeitpunkt noch keine festen Anforderungen gestellt.

2.2.1.3 Nichtfunktionale Anforderungen und Rahmenbedingungen

Überwiegend waren zu Beginn der Arbeit bereits einige nichtfunktionale Anforderungen festgelegt worden, die in diesem Kapitel beschrieben werden.

In erster Linie sollte die zu entwickelnde App für jeden Anwender am BMW-Familientag nutzbar sein. Zu diesem Zweck galt es die Steuerung, bzw. Interaktion mit der App so einfach und intuitiv wie nur möglich zu gestalten.

Aufgrund des Umfangs der Besucherzahlen am BMW-Familientag war es zudem erforderlich das die App eine gewisses Maß an Zuverlässigkeit gewährleitet. Diese Anforderung entstand aufgrund der Tatsache, dass die App auch von anderen Mitarbeitern der BMW als den Entwickler vorgestellt werden musste.

Eine weitere Anforderung war die Sicherheit bei der Verwendung der App im Zusammenhang mit der Aktuatorik des PKW. Hierbei sollten keine Systemzustände erreichbar sein, welche die Gesundheit der Benutzer gefährden könnten. Besonders kritisch für die Personensicherheit war dabei die Aktive Rollstabilisierung. Bei geöffneter Türe am PKW dürfte es nicht möglich sein den Rollwinkel des Fahrzeugs zu verändern. Formal wurden diese Anforderungen nicht festgehalten. Diese

Anforderungen bezüglich der Sicherheit wurden allerdings auf Seite der Kommunikations-Komponente umgesetzt, die in Kapitel 2.2.2 erläutert wird.

Rahmenbedingungen für die Entwicklung richteten sich an die verwendete Technik und Hardware, als auch an das Vorgehen aufgrund der Zeitvorgaben. Als Plattform für die Anwendung sollte ein mobiles Endgerät vom Type ASUS Eee Pad Transformer Prime TF201 eingesetzt werden. Dieses Gerät wurde bereits vorher aufgrund der relativ hohen Leistung für ein Tablet-PC von BMW eingekauft. Im weiteren Verlauf der Arbeit wurden zwei weitere Geräte vom Typ ASUS Transformer Pad TF300T für die Präsentation am BMW-Familientag eingekauft. Das von diesen Geräten verwendete Betriebssystem ist Android 4.0.3 Ice Cream Sandwich. Aus diesem Grund handelt es sich bei der verwendeten Programmiersprache zur Entwicklung der App um JAVA. Zur Kommunikation sollte WLAN mittels TCP/IP eingesetzt werden, da eine entsprechende Schnittstelle bereits auf Seiten der Kommunikations-Komponente implementiert worden war.

Auch das Nachrichtenformat war bereits aufgrund der Anforderung der direkten Lesbarkeit von Nachrichten vorgegeben. Zu diesem Zweck wurden Nachrichten als JSON-Objekte gekapselt. Dabei handelt es sich um ein leichtgewichtiges Textformat zum Datenaustausch, welches menschenlesbar und unabhängig von der verwendeten Programmiersprache ist. [INJSON]

Das Projekt in Bezug auf die Entwicklung eines funktionsfähigen Demonstrators musste bis zum 01.07.2012 abgeschlossen sein, damit dieser auf dem BMW-Familientag präsentiert werden konnte. Somit war auch der zeitliche Rahmen bezüglich des Abschlusses der Implementierungs- und Testarbeiten vorgegeben.

2.2.2 Abgrenzung

Im folgenden Kapitel werden die einzelnen Aufgabengebiete erläutert, die von Robert Trampler und mir, Maximilian Schüler, abgedeckt wurden, als auch die Grenzen zwischen diesen Bereichen definiert. Zusätzlich wird beschrieben welche Themen durch diese Arbeit im Allgemeinen in Bezug auf Augmented Reality behandelt werden und welche Gebiete davon abgegrenzt werden.

In dieser Masterthesis wird die Entwicklung eines Prototypen zur Darstellung von Steuergerätefunktionen beschrieben, als auch weitere Ideen beschrieben, deren Umsetzung im Rahmen der Masterarbeit nicht mehr möglich waren. Des Weiteren wird das Thema Augmented Reality mit Bezug auf die Erklärung von Steuergerätefunktionen diskutiert. Dabei kommen bestehende Rahmenwerke zum Einsatz, die für mobile Geräte entwickelt wurden. Diese Frameworks stellen Funktionen bereit, welche für die Realisierung einer Augmented Realtiy Anwendungen notwendig sind. In der Arbeit werden keine Neu-, bzw. Eigenentwicklungen in Bezug auf Augmented Reality diskutiert. Für den Umfang der Arbeit ist die Entwicklung eines eigenen Frameworks zu umfangreich und daher nicht Teil dieser Masterarbeit.

Insbesondere befasst sich diese Masterthesis mit dem, auf der rechten Seite von Abbildung 9 dargestellten Bereich, des Gesamtkonzepts, welcher mit roter Farbe umrandet wurde. Thematisch lassen sich die Diplomarbeit von Robert Trampler und diese Masterthesis gemäß der physikalischen Darstellung bei der Funkübertragung mittels WLAN zwischen dem Tablet und der Kommunikations-Komponente abgrenzen.

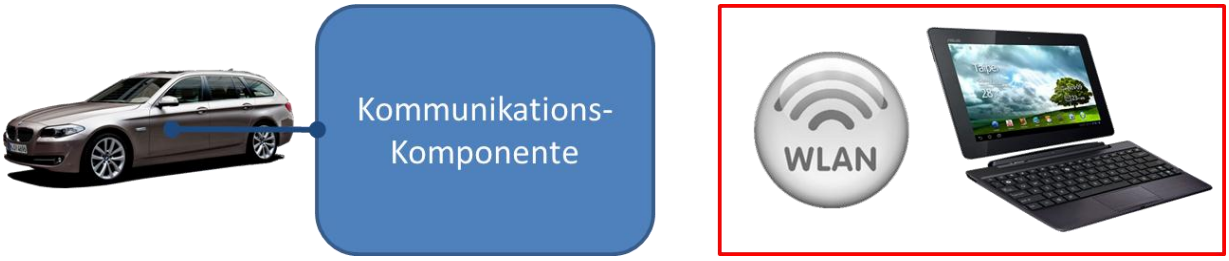


Abbildung 9 Konzeptuelle Darstellung des Demonstrators

Der Zugriff auf das Bordnetz, bzw. die Manipulation der Bussignale, des BMW F11 wurde von Robert Trampler durchgeführt. Dabei wurden Methoden erarbeitet bestimmte Effekte der Aktuatorik per Anfrage an der Kommunikations-Komponente bereitzustellen. Diese Funktionen sollten anschließend von der Tablet-Anwendung verwendet werden um die Steuergerätefunktionen anschaulich am realen Fahrzeug darstellen zu können. Um das notwendige Maß an Personensicherheit gewährleisten zu können musste dieser Teil des Demonstrators Sicherheitsmechanismen umfassen, die kritische Systemzustände im Vorhinein ausschließen.

Um die Nachrichten der Kommunikations-Komponente auf Seiten der Tablet-Anwendung verarbeiten zu können musste die entsprechende Schnittstelle gemäß den Vorgaben implementiert werden. Während des Projektes wurden dabei noch einige Veränderungen vorgenommen, die somit auch auf Seiten der Tablet App angepasst werden mussten. Auch die Übertragung via WLAN zwischen Tablet und Kommunikations-Komponente war Teil dieser Arbeit. Dies umfasste die Konfiguration des WLAN Netzwerks als auch organisatorische Tätigkeiten im Hinblick auf den BMW-Familientag um eine Kommunikation zu ermöglichen. Besonders kritisch dabei waren die Bedingungen bezüglich der Latenzzeiten, die an einige Anwendungsfälle gekoppelt waren. Beispielsweise durften keine großen Verzögerungen bei dem Anwendungsfall zur Erklärung der Aktivlenkung auftreten, da hier sonst ein Versatz beim Lenkverhalten aufgetreten wäre. Diese Bedingungen werden in Kapitel 4.2.3.2 erläutert, das sich detailliert mit dem Anwendungsfall der Aktivlenkung beschäftigt.

Gerade für den Anwendungsfall der Aktivlenkung konnte auf Vorarbeiten des Praktikanten Lukas Obkircher zurückgegriffen werden, der bereits vor Beginn der Masterarbeit erste Beispiele zur Umsetzung eines Prototyps implementierte. Besonders hilfreich waren Methoden zur Berechnung der Lage-Erkennung des Tablets, die in abgewandelter Form für verschiedene Anwendungsfälle eingesetzt werden konnten.

Die Architektur des Demonstrators wurde in diesem Kapitel lediglich konzeptuell erfasst, während in Kapitel 4.1 und 4.2 diese genauer erläutert wird. Der Einfachheit halber wurde hier lediglich das Konzept in groben Zügen erläutert um die Aufgabenstellungen der beiden Arbeiten besser voneinander abgrenzen zu können.

Im folgenden Kapitel wird auf das Vorgehen während der Bearbeitung des Projektes eingegangen. Dabei bezieht sich das Kapitel vorwiegen auf den Zeitraum der die erste Hälfe der Masterarbeit umfasst. In dieser Phase der Arbeit wurden Inhalte im Team mit Herrn Robert Trampler erarbeitet und umgesetzt. Aufgrund der Abhängigkeiten der beiden Arbeiten hinsichtlich der Kommunikation zwischen Tablet und Fahrzeug und aufgrund des relativ kurzen Entwicklungszeitraums waren kurze Kommunikationswege ein klarer Vorteil. Tests als auch Änderungen konnten aufgrund des gemeinsamen Arbeitsplatzes bei BMW kurzfristig durchgeführt werden.

2.3 Vorgehen und Ablauf des Projekts

Für die Entwicklung des Prototyps im Hinblick auf den BMW-Familientag wurde zu Teilen eine angepasste Ausprägung des agilen Vorgehens angewandt. Aufgrund des Kontextes der Aufgabenstellung konnte zu Beginn der Arbeit noch kein klar definierter Zustand der vorgefertigten Kommunikations-Komponente zugesichert werden. Demnach mussten Entscheidungen schnell getroffen werden und unmittelbar im Team kommuniziert werden. Gerade bezüglich der Schnittstelle zwischen Tablet und Kommunikations-Komponente war es notwendig, dass zwischen Teammitgliedern jederzeit ein konsistentes Verständnis über Kommunikationsabläufe und Nachrichtenformate vorherrscht.

Bei diesem kleinen Projekt mit einer Teamgröße von genau zwei Entwicklern und wegen des explorativen Charakters des Prototypen, bzw. Demonstrators, konnte somit von einer unverhältnismäßigen Bürokratie bezüglich der Dokumentation im Vorfeld abgesehen werden. Lediglich die Schnittstelle bezüglich des Nachrichtenformats und der einzelnen Nachrichten wurde in einem Dokument festgehalten, wie bereits in Kapitel 2.2.1.3 erwähnt wurde. Besonders agile Vorgehensweisen und Extreme Programming[11] kommen in diesem Kontext zum Einsatz. Streng genommen wurde für das Projekt kein klares Vorgehensmodell verwendet, sondern das Vorgehen projektspezifisch angepasst. Am ehesten lässt sich auf die Arbeitsweise im Projekt wohl das SCRUM[12] Vorgehensmodell anwenden. Täglich wurden Absprachen über das weitere Vorgehen getroffen und diese Arbeitspakete auch entsprechend priorisiert behandelt. Dies entspricht im Grunde der Verwendung von Backlogs wie sie unter anderem in SCRUM eingesetzt werden. Auch wurden kurze Intervalle für die Umsetzung von bestimmten Funktionalitäten gewählt. Maximal wurde eine Woche bis zu einer erneuten Orientierung aufgewandt. Somit sind diese Intervalle mit den aus SCRUM bekannten Sprints vergleichbar. Die Rollen wurden an SCRUM angelehnt in folgender Weise verteilt: Als ScrumMaster kann man Herrn Markus Reschka nennen, welcher im Verlauf des Projekts für eine klare Aufteilung sorgte und darüber hinaus das Team mit Mitteln versorgte, die für die Umsetzung des Projekts notwendig waren, bzw. die Produktivität des Teams förderten. Als Product Owner nach SCRUM kann man die Personen der Entwicklungsteams einzelner Steuergeräte bezeichnen, die Ideen und Kritik in Bezug auf die Erklärung der Steuergerätefunktionen äußerten. Regelmäßige Treffen mit diesem Personenkreis war fester Bestandteil vor dem BMW-Familientag. Zu gegebener Zeit wurden auch andere Formen von Vorgehensmodellen eingesetzt. Zeitweise wurden Prototypen erstellt, die zur Klärung von Fragestellungen oder zum Test von Implementierungsideen eingesetzt wurden. Aus diesem Grund wurden somit auch Teile des explorativen und experimentellen Prototypings angewandt. [TUMSWT]

Aufgrund der Risiken bezüglich der technischen Durchführbarkeit des Projektes und des Projekterfolgs war der explorative Charakter des Vorgehens vorteilhaft. Dabei wurden schrittweise die technische Umsetzung erschlossen und am Demonstrator umgesetzt. Tests konnten dabei aufgrund der geringen Teamgröße unmittelbar nach kurzer Absprache durchgeführt werden. Wegen der Diskrepanzen bezüglich des Entwicklungsstands der Seite der Kommunikations-Komponente gegenüber der Tablet-Anwendung konnte erst zu einem relativ späten Zeitpunkt Tests in Form eines technischen Durchstichs durchgeführt werden. Um während der Entwicklung weitestgehend unabhängig von Arbeitsergebnissen des Teammitglieds arbeiten zu können, wurde von Robert Trampler ein Simulator entwickelt, der den Bereich der Kommunikations-Komponente zusammen mit

[11] Beim Extreme Programmierung steht die Programmierung im Vordergrund des Vorgehens.
[12] SCRUM ist eine Form der agilen Methoden mit besonders simplen Strukturen und klarem Rollenkonzept.

dem PKW nachbildet. Der simulierte Teil sollte dabei ein ähnliches Verhalten wie die Endanwendung der Entwicklung von Robert Trampler aufweisen. Wegen der Abweichungen im Verhalten zwischen Simulator und realer Komponente mussten schwerwiegende Änderungen bei der Implementierung vorgenommen werden. Gründe waren beispielsweise die Verarbeitungsgeschwindigkeit von Nachrichten auf Seiten der Kommunikations-Komponente und die Anforderung nach geringen Latenzzeiten bei Anfragen vom Tablet. Die Herausforderungen bei der Umsetzung dieser Komponente sind allerdings nicht Teil dieser Arbeit und werden hier auch nicht weiterführend erläutert. Eine Ausnahme dazu stellt der Implementierungsteil zur Verbesserung der Latenzzeiten dar, der in Kapitel 4.2.1 behandelt wird.

Nach Fertigstellung der Implementierungs- und Testarbeiten bezüglich der Kommunikation von einem realen Steuergerät mit entsprechender Aktuatorik bis zur Tablet-Anwendung konnten Tests unter ähnlichen Umständen wie für den BMW-Familientag vorgesehen durchgeführt werden. Weitere Änderungen der Schnittstelle wurden an dieser Stelle nahezu eingestellt, um die inkrementelle Weiterentwicklung der Tablet-Anwendung zu ermöglichen.

Da auch während der Bearbeitungszeit keine Dokumente zum Monitoring des Projektfortschritts verfasst wurden, wird an dieser Stelle nicht weiter auf Herausforderungen während des Projektverlaufs eingegangen. Diese werden an geeigneter Stelle bei der Beschreibung des entwickelten Systems erwähnt.

Abschließend wird noch eine kurze Zusammenfassung in Form eines Reviews über das Projektverlauf gegeben. Dabei sollen die „Lessons Learned“ und ein Fazit zur ersten Hälfte der Masterarbeit vermittelt werden.

Besonders für organisatorische Tätigkeiten in Bezug auf die Vorbereitungen des BMW-Familientages als auch für die Beschaffung der notwendigen Hardware wäre eine zusätzliche Person im Team sinnvoll gewesen. Des Weiteren hätte es zu einer Zeitersparnis in der Implementierung und den Tests geführt, wenn bereits zu Beginn der Masterarbeit eine finale und fixe Version der Kommunikations-Komponente vorhanden gewesen wäre oder deren Schnittstelle formal und final definiert worden wäre. Dabei wären UML-Sequenzdiagramme zum Ablauf der Kommunikation als auch Dokumente über das exakte Nachrichtenformat inklusive aller Frames und eingetragenen zulässigen Werte sinnvoll gewesen. Zwischenzeitliche Veränderungen, bzw. Diskrepanzen zwischen dem Austauschdokument und der Verwendung in der Realisierung der Kommunikation-Komponente, führten dabei zu Fehlersuchen und zu einem erhöhten Zeitaufwand bei Implementierung und Test. Die zeitliche Überlappung der Bearbeitungszeiträume für die Abschlussarbeit von Robert Trampler und der vorliegenden Arbeit, hatte den Vorteil, dass Abläufe anhand des Expertenwissens des jeweiligen Teammitglieds gegenseitig vermittelt werden konnten. Somit konnte ein gemeinsamer Konsens bezüglich der Inhalte hergestellt werden. Diese Besprechungen führten allerdings häufig zu Unterbrechungen bei der Bearbeitung von Arbeitspaketen. Um weiterhin ein leichtgewichtiges Vorgehen zu gewährleisten, aber die Produktivität der Bearbeiter zu unterstützen, wären demnach wenige finale Dokumente im Vorhinein zu erstellen gewesen, die nicht weiteren Veränderungen unterliegen und die als Basis für Implementierungen dienen. Das Projekt konnte am BMW-Familientag erfolgreich mit der Präsentation eines funktionsfähigen Demonstrators abgeschlossen werden. Die Umfänge des Demonstrators wurden aufgrund von Zeitmangel auf ein entsprechendes Maß gekürzt (Timeboxing). Eine Verschiebung der Deadline zugunsten weiterer Funktionen war aufgrund der Größe des BMW-Familientags mit circa 90.000 Besuchern und des damit verbundenen

Organisationsaufwands nicht möglich. Eine detaillierte Beschreibung der einzelnen Entwicklungsbestandteile des Demonstrators zeigt Kapitel 4 auf. [TUMSWT]

Das folgende Kapitel beschäftigt sich mit der Lösung der geschilderten Problematik und deren Umsetzung anhand eines Demonstrators.

3 Allgemeine Problemlösung und Innovative Technologien

Um eine geeignete Form der Präsentation zur Verfügung zu stellen, ist es notwendig den Zuhörer nicht nur zu informieren, sondern auch zu unterhalten. In diesem Sinne bestand die Aufgabenstellung in der Erklärung von technischen Funktionen für das breite Publikum des BMW-Familientages.

Das umfasst zum einen, eine ansprechende Präsentation und zum anderen eine geeignete Darstellungsform dieser Inhalte. Aus diesem Grund wurden die einzelnen Anwendungsfälle innerhalb einer App umgesetzt, die auf spielerischem Weg die Funktionen der Steuergeräte vermittelt. Die ständige Zunahme an mobilen Endgeräten in Form von Mobiltelefonen und Tablets zeigt, dass diese Geräte weithin bereits im Alltag verwendet werden und somit auch bei einer Präsentation die Akzeptanz der Benutzer, bzw. Zuhörer, beim Umgang mit den Präsentationsinhalten erhöhen. Laut der Marktforschung durch Gartner Inc. für das zweite Quartal 2012 zu den weltweiten Verkaufszahlen an Mobilfunktelefonen kann man nicht nur erkennen, dass diese Geräte weit verbreitet sind, sondern auch welche Betriebssysteme in diesem Zusammenhang meist verwendet werden. Auch durch die International Data Corporation (IDC) werden Zahlen für das zweite Quartal 2012 genannt, welche die Verbreitung von Android-Geräten auf dem Markt verdeutlicht. In Abbildung 10 ist eine Übersicht zu den weltweiten Marktanteilen der verschiedenen Betriebssysteme von Smartphones dargestellt. Bei den verschiedenen Marktforschungen kann man erkennen, dass sich das Android Betriebssystem immer stärker auf dem Markt verbreitet. Somit ist dieses Betriebssystem eine sinnvolle Wahl für den Einsatz im Demonstrator. Nicht nur, dass die Wahrscheinlichkeit relativ hoch ist, dass der Benutzer den Umgang mit einem Android-Gerät gewohnt ist. Zudem ist das Betriebssystem als Grundlage der Entwicklung des Demonstrators auch für zukünftige Projekte und Entwicklungen geeignet. [GARTWS][IDCIWS]

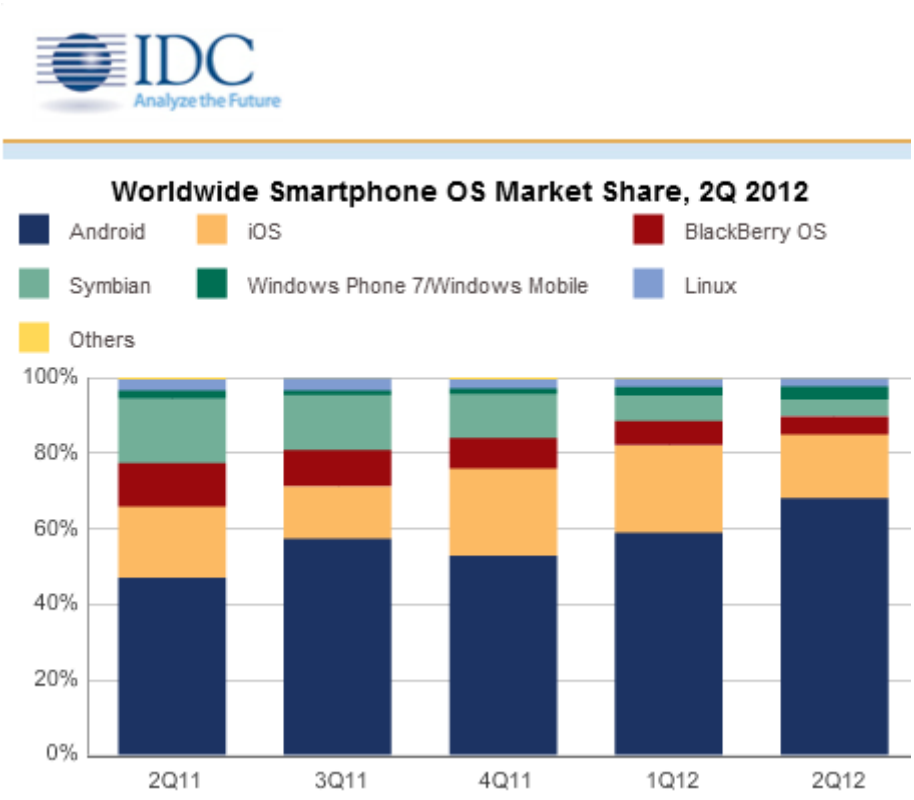


Abbildung 10 Weltweite Marktanteile von Smartphone-Betriebssystemen, [IDCIWS]

Um Inhalte in geeigneter Größe darstellen zu können ist es sinnvoll ein Tablet statt eines Smartphones für die Präsentation einzusetzen. Nichtsdestotrotz wäre eine Portierung des Präsentationsinhalts von einem Tablet auf ein Smartphone aufgrund des Android Betriebssystems ohne weiteren Entwicklungsaufwand möglich. Einzige Voraussetzung, damit alle Funktionen voll funktionsfähig sind, ist, dass die notwendige Sensorik auch auf dem Zielgerät vorhanden ist. Aufgrund der starken Präsenz von mobilen Geräten wie Tablets auf dem Weltmarkt passte das Marktforschungsunternehmen Gartner Inc. seine Prognosen für den Verkauf von PCs in den letzten beiden Jahren gegenüber anfänglicher Prognose bereits zweimal nach unten an. Auch diese Tatsache spricht für den Einsatz von Tablets bei der Präsentation von Inhalten, da es sich bei diesem Medium um ein zeitgemäßes Gerät handelt, welches zurzeit stark im Interesse der Bevölkerung steht. [GARTTA]

Um weiterhin das Interesse der Besucher des BMW-Familientags gegenüber den vorzustellenden Inhalten zu erhöhen, sollte auch ein gewisser Unterhaltungswert transportiert werden. Betrachtet man das jährliche Wachstum der Computerspiele-Industrie besonders im Sektor der mobilen Endgeräte, so wird deutlich, dass die Unterhaltung mittels Computer-Spiele längst auf den Smartphones und Tablets angekommen ist. Laut Gartner Inc. entfallen nach einer Schätzung circa 70 bis 80 Prozent aller heruntergeladenen mobilen Anwendungen auf mobile Computer-Spiele-Anwendungen. Bei dem Demonstrator wurde der Versuch unternommen, das Bedürfnis nach Spaß und Informationen miteinander in der Form eines Computer-Spiels zu verbinden. [GARTGA]

Auch die Augmented Reality Technologie verfügt über großes Potential und ist gleichzeitig noch eine relativ junge Technologie, die Kundschaft als auch Technikinteressierten durch entsprechenden Einsatz von virtuellen Komponenten begeistern kann. Dabei stellt auch der Umgang mit einer AR-Anwendung auf einem Tablet eine neue Erfahrung für viele Benutzer dar. Auf den Hype bezüglich Augmented Reality wird in Kapitel 7 genauer eingegangen. Weitere Vorteile von Augmented Reality werden in Kapitel 4.4 und Kapitel 5 in Bezug auf die Anwendung für die Erklärung von Steuergerätefunktionen beschrieben. [VTFAR]

Gerade die Vereinigung der verschiedenen Technologien, wie die Manipulation von Aktuatoren am Fahrzeug durch ein modernes mobiles Endgerät und die Erweiterung der Realität durch virtuelle Komponenten, stellen für Benutzer einen besonderen Reiz dar.

Im folgenden Kapitel wird der Demonstrator für den BMW-Familientag detailliert beschrieben. Darunter werden zudem die einzelnen Anwendungsfälle und die verwendeten Tools und Technologien erläutert, die zu deren Umsetzung beigetragen haben.

4 Demonstrator und BMW-Familientag

Für den BMW-Familientag am 01.07.2012 wurde ein Stand auf dem Gelände des Forschungs- und Innovationszentrums (FIZ) der BMW Group in München bereitgestellt, um die Funktionsweise des ARS zu demonstrieren. Aufgrund der hohen Besucherzahlen wurden auf dem Stand drei BMW F11 zur Verfügung gestellt, um das ARS System und Teile der Aktivlenkung zu erfahren. Für zwei Fahrzeuge war jeweils ein eigenes Tablet vom Typ ASUS Transformer Pad TF300T[13] vorgesehen während an dem dritten Fahrzeug ein ASUS Eee Pad Transformer Prime TF201[14] eingesetzt wurde. Die beiden verwendeten Tablets unterscheiden sich nur geringfügig voneinander und wurden keinem bestimmten Muster entsprechend eingesetzt.

Im weiteren Verlauf werden die Strukturierung, bzw. Aufbau, der Hardware als auch der Software detailliert erläutert und die Kommunikation der einzelnen Komponenten untereinander dargestellt. Abschließend werden die im Demonstrator umgesetzten Anwendungsfälle anhand der App beschrieben. Zudem wird ein Überblick über den Ablauf des Familientages gegeben und die Weiterentwicklung des Demonstrators diskutiert.

4.1 Hardware-Architektur des Demonstrators

Der Aufbau des Demonstrators wird anhand eines der drei Fahrzeuge in diesem Kapitel genauer erläutert. Dabei wird auf die technischen Grundlagen des Demonstrators und das Einrüsten des Standes am Familientag eingegangen.

Wie in Abbildung 9 auf Seite 17 abstrakt dargestellt, muss es zwischen dem Tablet und der Kommunikations-Komponente die Möglichkeit der Kommunikation via WLAN geben. Zu diesem Zweck musste ein WLAN-Netz, bzw. WLAN-Zugang, für die beiden Parteien bereitgestellt werden. Da es nicht erlaubt ist, fremde Geräte, wie ein Android-Tablet, in das BMW WLAN-Netzwerk zu integrieren, musste auf einen eigenen Router vor Ort ausgewichen werden. Auch zu Testzwecken wurden bereits vorher unterschiedliche Geräte zur Bereitstellung eines WLAN-Netzwerks eingesetzt. Besonderheiten beim Umgang mit der WLAN-Kommunikation werden im Kapitel 4.2.1 erläutert, welches besonders auf das Thema Kommunikation eingeht.

Die logische Einheit der Kommunikations-Komponente wird im weiteren Verlauf weiter in seine physikalischen Bestandteile untergliedert und beschrieben. In Abbildung 11 werden die einzelnen Teilkomponenten und Kommunikationsformen aufgeführt, die verwendet wurden.

Abbildung 11 Konzeptzeichnung, Verfeinerung der Kommunikations-Komponente, Fahrzeug (links), Midget (mittig) und Notebook (rechts)

[13] Siehe http://www.asus.de/Tablet/Transformer_Pad/ASUS_Transformer_Pad_TF300T/
[14] Siehe http://www.asus.de/Eee/Eee_Pad/Eee_Pad_Transformer_Prime_TF201/

Um die Aktuatorik am PKW verändern zu können müssen Bussignale, wie im konkreten Fall FlexRay-Nachrichten[15], manipuliert und an das Fahrzeug versandt werden. Zu diesem Zweck wurde eine Restbussimulation durchgeführt. Diese Simulation wird durch ein FlexXCon Midget der Firma Eberspächer [16] erledigt, welches zyklisch die entsprechend hinterlegten FlexRay-Nachrichten versenden kann. In jedem Fahrzeug werden zur Simulation der in den Anwendungsfällen verwendeten Steuergeräte jeweils drei Midgets verwendet. In Abbildung 11 wurde zur Vereinfachung lediglich ein einzelnes Midget dargestellt. Um die entsprechenden FlexRay-Nachrichten während der Laufzeit verändern zu können, werden die Midgets mittels Ethernet mit dem Notebook verbunden. Aufgrund der Anzahl der Midgets in einem PKW muss zusätzlich jeweils ein Hub, bzw. Switch eingesetzt werden, welches ein LAN-Netzwerk zwischen Notebook und den drei Midgets herstellt. Auch die Anbindung an das Notebook ist in Abbildung 11 nur konzeptionell abgebildet. Durch diesen Aufbau ist es möglich über eine Steuerungssoftware, die auf dem Notebook ausgeführt wird, die Steuergeräte für entsprechende Effekte an der Aktuatorik mit simulierten Daten zu versorgen. Für eine detaillierte Beschreibung zur Restbussimulation und zum Aufbau der Kommunikations-Komponente wird auf die Abschlussarbeit von Robert Trampler verwiesen. [BMWRTD]

Die gesamte Kommunikations-Komponente konnte zusammen mit der Verkabelung jeweils im Kofferraum eines BMW F11 transparent für den Benutzer und platzsparend untergebracht werden. Zudem wurde der Kofferraum der Fahrzeuge zur Präsentation geschlossen. Auch die Verkabelung der FlexRay-Kommunikation konnte unter Abdeckungen versteckt werden und wurde erst im Fußraum des Fahrers sichtbar. Aus diesem Grund war es den Zuschauern nicht erlaubt während des BMW-Familientages auf dem Fahrersitz Platz zu nehmen. Durch diese Maßnahmen sollte für den Benutzer der Eindruck entstehen, dass es sich bei den Fahrzeugen um serienmäßige Fahrzeuge handelt, die tatsächlich über die vorgeführten Funktionen verfügen.

4.2 Software-Architektur des Demonstrators

Insbesondere geht dieses Kapitel nicht nur auf die Struktur sondern auch auf das Verhalten des Demonstrators bezüglich der Software ein. Dabei werden verschiedene Diagramme der Unified Modeling Language (UML)[17] Version 2.0 eingesetzt um den Fokus auf bestimmte Aspekte der Modellierung zu lenken.

Auf dem in Abbildung 11 dargestellten Notebook auf der rechten Seite der Zeichnung wird, wie bereits in Kapitel 4.1 beschrieben, eine Steuerungssoftware benötigt, die es ermöglicht die Aktuatorik des Fahrzeugs zu manipulieren. Für einen gezielten Zugriff auf einzelne Funktionen der Aktuatorik und um die Vorgänge für die Manipulationen zu abstrahieren, wurde eine Client-Server Architektur zur indirekten und abgesicherten Kommunikation mit dem Fahrzeug gewählt. Nach dem Prinzip „Lose Kopplung – starke Bindung“ wird somit möglichst viel Komplexität von der Server-Komponente gekapselt und bietet wenige schmale Schnittstellen nach außen an. Dabei werden alle Teile in Verbindung mit der Kommunikation zum Fahrzeug aufgrund des engen Zusammenhangs in dieser Komponente zusammengefasst. [TUMSWA]

Für den genauen Aufbau der Server-Komponente wird auf die Abschlussarbeit [BMWRTD] von Robert Trampler verwiesen. Eine detaillierte Beschreibung dieser Komponente ist nicht Teil dieser

[15] FlexRay ist ein robustes, skalierbares, deterministisches und fehlertolerantes digitales serielles Bussystem, welches für Anwendungen im Automobil entwickelt wurde. [FLXRAY]
[16] Siehe http://www.eberspaecher-electronics.com/produkte/flexxcon-midget.html
[17] Siehe http://www.uml.org/

Abschlussarbeit. Lediglich die Schnittstellen zur Client-Komponente werden bezüglich der Interaktionen zwischen Client- und Server-Komponente erläutert. Im weiteren Verlauf wird Client und Server synonym mit Client- und Server-Komponente verwendet. Die Implementierung des Client wurde mittels eines Android-Projektes umgesetzt. Als Entwicklungsumgebung wurde dabei die Eclipse IDE eingesetzt. Die Versionen der einzelnen Tools werden in Kapitel 6.1 aufgeführt.

Besonders hilfreich bei der Entwicklung von Anwendungen mit mehreren Threads (Multithreading) im Android Kontext war der Dalvik Debug Monitor Server [18](DDMS), um zu ermitteln, welche Threads aktiv sind und ob alle Threads nach dem Beenden der Anwendung terminieren. Außerdem konnte anhand der Network Statistics der Nachrichtenverkehr überwacht werden. Diese Funktion wurde insbesondere zur Optimierung der Client-Server-Kommunikation eingesetzt.

Im folgenden Kapitel wird detailliert auf die Umsetzung der Kommunikation eingegangen. Zudem werden Alternativen zur aktuellen Umsetzung behandelt und Herausforderungen während der Entwicklung diskutiert.

4.2.1 Kommunikation

Zur Kommunikation zwischen Client und Server wird, wie bereits erwähnt, ein WLAN eingesetzt. Als Netzwerkprotokolle wurden zur Übertragung von Daten TCP/IP eingesetzt. Ein Vorteil bei der Verwendung dieser Technologie ist, dass der Implementierungsaufwand gegenüber anderen Protokollen relativ gering ist. In der zur Entwicklung des Demonstrators verwendeten Programmiersprache JAVA werden durch das Paket *java.net* Klassen bereitgestellt, welche die notwendige Funktionalität zur Kommunikation mittels TCP/IP[19] und UDP/IP[20] zur Verfügung stellen. Aufgrund der Vorgabe durch die Entwicklung des Servers wurde die Verwendung von TCP/IP zur Kommunikation gefordert. Die paketorientierte Kommunikationsform mittels UDP/IP wurde nicht unterstützt. UDP/IP garantiert weder, dass die Datenpakete tatsächlich beim Verbindungspartner ankommen, noch, dass die Nachrichten in der korrekten Reihenfolge ankommen. Allerdings bietet UDP/IP in JAVA die Möglichkeit einen Broadcast durchzuführen. Ein Broadcast wäre eine Möglichkeit die Kommunikation in einer 1:n-Beziehung zwischen Tablet und Kommunikations-Komponente durchzuführen. Zusätzlich kann hier beim Datenversandt Zeit durch eine Vorkonfiguration der Datenpakete mittels der *connect()*-Methode vorgenommen werden. Durchführbar wäre eine Vorkonfiguration aus dem Grund, da Daten grundsätzlich zwischen einer festen Gruppierung von zwei Geräten verschickt werden, bzw. eine feste Zieladresse im Vorhinein bekannt ist. Dabei wird dann entweder die Zieladresse des Kommunikationspartners bei einer 1:1-Kommunikation (Unicast) eingetragen oder die Broadcast -Adresse des Subnetzes. Eine weitere Alternative für eine effizientere Übertragung an mehrere Parteien des Subnetzes stellt die Klasse *java.net.MulticastSocket* dar. Diese Klasse ermöglicht es einen Multicast im Subnetz durchzuführen. Dies könnte zu einer Verbesserung bei der Kommunikation im Anwendungsfall des synchronen Wankens führen. Der Anwendungsfall zum synchronen Wanken wird in Kapitel 4.2.3.3 zusammen mit den bestehenden Herausforderungen beschrieben.

Auch die Umsetzung der Kommunikation mittels Bluetooth[21] könnte zu einer Verbesserung der Latenzzeiten bei den einzelnen Anwendungsfällen führen. Bluetooth wurde unter anderem im Hinblick auf einen geringen Stromverbrauch entwickelt. Diese Tatsache stellt einen Vorteil gegenüber

[18] Siehe http://developer.android.com/tools/debugging/ddms.html
[19] Java.net.Socket und java.net.ServerSocket siehe [JAVANE]
[20] Java.net.DatagramSocket siehe [JAVANE]
[21] Siehe http://www.bluetooth.com/Pages/Bluetooth-Home.aspx

der WLAN Technologie dar, da bisherige Tests gezeigt haben, dass die entwickelten Anwendungsfälle einen hohen Energieverbrauch aufweisen. Dieser Energieverbrauch entsteht zu einem großen Teil durch die Kommunikation über WLAN. Die geringe Teilnehmerzahl im Demonstrator kann auch in einem Piconetz mittels Bluetooth Vernetzung ausreichend abgedeckt werden. Die geringe Reichweite ist dabei irrelevant, da der Benutzer in kurzer Distanz vom Fahrzeug die Anwendung bedient. Aus Sicht der IT-Sicherheit handelt es sich bei der Sicherheitsarchitektur und dem Design der Protokolle von Bluetooth gegenüber dem WLAN Standard IEEE 802.11 um das geeignetere Übertragungsverfahren. Aufgrund der Personensicherheit im Umgang mit dem Fahrzeug muss auch der Aspekt der IT-Sicherheit bei der Übertragung berücksichtigt werden. Die geringere Datenübertragungsrate bei einer synchronen, verbindungsorientierten, leistungsvermittelnden Punkt-zu-Punkt Verbindung von 433,9 KBits/s, ist aufgrund des geringen Datenaufkommens zu vernachlässigen. In der Bluetooth Version 2.0[22] wird zudem die Datenrate auf bis zu 3 Mbit/s erhöht, während der Energieverbrauch gegenüber dem ursprünglichen Standard nochmals ungefähr halbiert wurde. Für die Anwendungsfälle steht die Rechtzeitigkeit der Datenübertragung im Vordergrund. Diese Anforderung erhebt den Anspruch, dass das Fahrzeug unmittelbar auf Eingaben am Tablet reagieren muss. Somit kann man den Demonstrator im Bereich der Echtzeitverarbeitung einordnen. Während der Entwicklung wurde diese Anforderung nicht genauer formal spezifiziert. In Kapitel 5.4.2 von [BMWRTD] wird bereits darauf hingewiesen, dass die Umsetzung der Kommunikation mittels WLAN im Demonstrator zu Schwierigkeiten bezüglich der Rechtzeitigkeit geführt hat. [BLUETO][ITSECB]

Für die Umsetzung der Kommunikation über TCP/IP musste auf Client-Seite die entsprechend von der Server-Seite vorgegebene Schnittstelle implementiert werden. Bei der Übertragung wurden, wie bereits erwähnt, JSON-Nachrichten versandt, um die direkte Lesbarkeit durch den Menschen zu unterstützen. Anhand dieser Nachrichten kann gezielt eine Anfrage bezüglich der Manipulation der Aktuatorik getätigt werden. Da es notwendig ist, dass in einem Anwendungsfall gleichzeitig mehrere Aktuatoren in kurzer Zeit mittels Anfragen manipuliert werden müssen, musste dafür ein Konzept erarbeitet werden. Für die Funktionalitäten von jeweils einem Aktuator wurde eine Gruppierung der Nachrichten in Form von Kanälen vorgenommen. Diese Kanäle werden unabhängig voneinander betrieben und ermöglichen eine nebenläufige Verarbeitung. Zudem muss bei diesem Vorgehen keine Sortierung der empfangenen Nachrichten bezüglich des Zielaktuators vorgenommen werden und vereinfacht damit den Rechenaufwand beim Empfang einer Nachricht. In Abbildung 12 unten werden die einzelnen Kanäle dargestellt, welche von der aktuellen Version des Servers verwaltet bzw. unterstützt werden. Aufgrund der Anzahl der Aktuatoren kann maximal ein Kanal für den gleichen Aktuator aufgebaut werden.

Im folgenden Kapitel wird der Verbindungsaufbau eines Kanals anhand des Verbindungsaufbaus für den Anwendungsfall bezüglich des ARS beschrieben. Auf den Inhalt der JSON-Nachrichten wird im Detail nicht eingegangen. Lediglich der strukturelle Aufbau einer Nachricht wird in Abbildung 12 dargestellt. Für die Kenntnis der Inhalte der einzelnen Nachrichten wird auf den Quellcode verwiesen. Im Anhang in Abbildung 37, Abbildung 38 und Abbildung 39 werden dagegen alle Nachrichten in Form von Java-Klassen zur Übersicht in einem UML-Klassendiagramm abgebildet. Um JSON-Nachrichten in JAVA verwenden zu können, wurde die Bibliothek *JSON4J* von IBM in Form einer JAR-Archivdatei in die Applikation eingebunden. Diese Bibliothek stellt alle notwendigen Methoden zur Verfügung, welche für den Umgang mit JSON-Nachrichten im Demonstrator benötigt

[22] Bei der Bluetooth Version 2.0 handelt es sich um den IEEE 802.15.1-Standard eine Weiterentwicklung des IEEE 802.15-Standards. [ITSECB]

werden. Bei JSON-Nachrichten, bzw. einem JSON-Objekt, handelt es sich um eine Menge von Schlüssel/Wert-Paaren, die beispielsweise wie in *JSONS4J* durch eine Erweiterung von *java.util.HashMap* verwaltet werden können. Der Datentyp des Schlüssels ist in diesem Fall immer ein *java.lang.String*. Werte können mit unterschiedlichen Datentypen versehen werden. Die verwendeten Datentypen der Werte einer JSON-Nachricht, wie sie im Demonstrator zum Einsatz kommen, werden in Abbildung 12 in der zweiten Spalte der Tabelle definiert. Als Datentyp für den „value“-Wert wird *java.lang.Integer* verwendet. Grundsätzlich wären auch andere Datentypen möglich gewesen, da *JSON4J* als Werte-Datentyp eines *com.ibm.json.java.JSONObject*-Objekts die abstrakte Klasse *java.lang.Number* zulässt. [JSON4JDO]

Schlüssel	Wert-Datentyp	Beschreibung	Beispiel-Werte
„target“	String	Anfragekanal	„car“
„action“	String	Angefragte Aktion	„start“
„value“	Integer	Spezifikation der Aktion	0
„control“	String	Nachrichtengruppe	„channel“

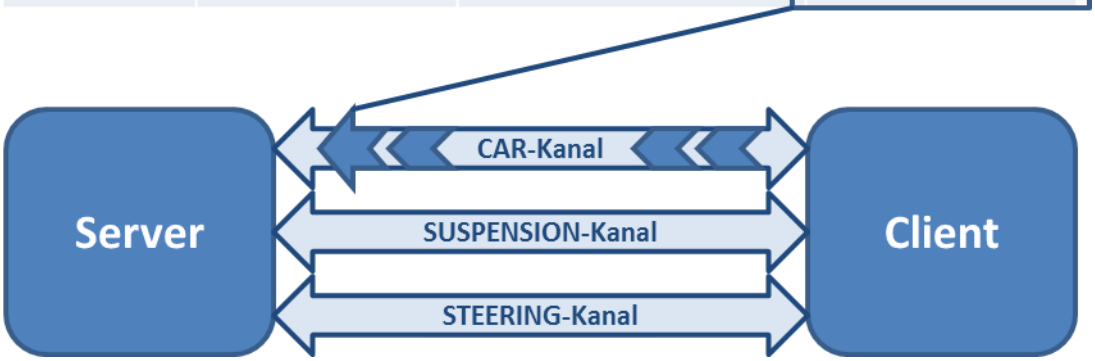


Abbildung 12 Allgemeiner Aufbau einer JSON-Nachricht im Demonstrator (oben) und Übertragungskanäle (unten)

Grundsätzlich gilt für alle Kanäle, dass diese von Server-Seite aus geschlossen werden, falls innerhalb einer vom Server festgelegten Zeit (*Keepalive time*) keine Nachricht empfangen wurde. Um dies zu verhindern, werden vom Client zyklisch *KeepAlive*-Nachrichten versandt, falls innerhalb der *KeepAlive time* keine andere Nachricht verschickt wurde. Zusätzlich wird jeder Kanal explizit durch eine bestimmte Nachricht auf- bzw. abgebaut. In Abbildung 12 wird in der Tabelle das Beispiel der JSON-Nachricht angegeben, welche für den Aufbau des CAR-Kanals zuständig ist und vom Client an den Server versandt wird. Bei dieser Nachricht wird als Anfragekanal, bzw. Nachrichtenziel, der CAR-Kanal definiert. Bei dem CAR-Kanal handelt es sich eher um die Gruppierung aller Steuergeräte, als tatsächlich um ein einzelnes Steuergerät wie bei anderen Kanälen. Die Kanäle SUSPENSION-Kanal und STEERING-Kanal betreffen jeweils genau ein Steuergerät. Der SUSPENSION-Kanal behandelt Anfragen an das ARS, während der STEERING-Kanal für die Aktivlenkung eingesetzt wird. Zusätzlich wird in Abbildung 12 mittels der Aktion „start“ vom Client ein neuer Kanal angefragt, bzw. das Öffnen des CAR-Kanals beantragt. Eine genaue Spezifikation der Aktion ist bei der derzeitigen Implementierung nicht möglich. Für Nachrichten, bei welchen keine genauere Spezifikation der Aktion notwendig ist, wird deshalb Null als Standardwert übertragen. Als Nachrichtengruppe wurde „channel“ festgelegt. Der „control“-Wert „channel“ wird für Aktionen eingesetzt, welche das Kanal-Management betreffen. Darunter fallen das Öffnen und Schließen von Kanälen.

Jede Nachricht, die an den Server geschickt wurde, wird von diesem mit einer entsprechenden Antwortnachricht quittiert. Somit kann auf Client-Seite ermittelt werden, ob eine Nachricht den

Server erreicht hat. Die Antwortnachricht auf die Anfrage aus Abbildung 12 enthält dabei den gleichen Inhalt, bzw. die gleichen Schlüssel-Wert-Paare, wie die auf Server-Seite empfangene Nachricht.

Der Ablauf des Verbindungsaufbaus zusammen mit dem implementierungstechnischen Hintergrund wird in Kapitel 4.2.2.2.1 beschrieben. Dabei wird besonders auf das Thema Multithreading im Zusammenhang mit der Kommunikation eingegangen.

Anhand der Network Statistics durch den DDMS konnte ein ungleichmäßiges Datenaufkommen bei der Übertragung festgestellt werden. Besonders kritisch ist dies bei Daten, die ohne großen zeitlichen Versatz einen Effekt der Aktuatorik bewirken sollen, wie beispielsweise bei dem Anwendungsfall zur Aktivlenkung, der in Kapitel 4.2.3.2 beschrieben wird. Bei unregelmäßigem Datenaufkommen wurden mehrere Nachrichten zurückgehalten und anschließend gebündelt auf einmal verschickt. Aufgrund der Nachrichtenverarbeitungszeiten des Servers führte dieses Aufstauen von Nachrichten dazu, dass veraltete Nachrichten unmittelbar vor der Verarbeitung verworfen werden müssen. Auf diese Weise kann die Anzahl der zu verarbeitenden Nachrichten auf ein Maß gesenkt werden, welches vom Server bearbeitet werden kann. Um eine kontinuierliche Datenübertragung mittels TCP/IP zu gewährleisten, wurden deshalb noch die Parameter bezüglich Type-of-Service (TOS) der Verbindung spezifiziert. Bei der Parametrierung wurde besonders Wert auf eine kurze Zeitverzögerung beim Nachrichtenversandt gelegt. Zu diesem Zweck konnten die Methoden *setTrafficClass()*[23] und *setTcpNoDelay()*[24] eingesetzt werden. In der Tabelle 1 ist der verwendete Quellcode abgebildet, der schließlich zu einer Verbesserung der Übertragung bezüglich des Aufstauens von Nachrichten führte.

```
1       // Tries to connect to the server
2       Socket newChannelSocket = new Socket(usedIP,usedPort);
3       // Disable segmentation via TCP settings
4       // TCP/IP Connection enriched by TOS Information
5       newChannelSocket.setTrafficClass(0x10); // IPTOS_LOWDELAY = 0x10
6       // disables Nagle's Algorithm, NO waiting for full data segments
7       newChannelSocket.setTcpNoDelay(true);
```

Tabelle 1 Quellcode, Setzen der TOS-Parameter eines Sockets

In der zweiten Zeile von Tabelle 1 wird unter Angabe der IP-Adresse und des Ports eine Socket-Verbindung von Seiten des Clients aufgebaut. Wird diese Socket-Verbindung von der Server-Seite akzeptiert, wird erfolgreich eine Verbindung zwischen Server und Client etabliert. In Zeile fünf wird der TOS-Parameter als Hexadezimalzahl 0x10 gesetzt, um die Verzögerung bei der Nachrichtenübertragung zu minimieren. Des Weiteren wird in Zeile sieben, laut der Java-Dokumentation, der Nagle-Algorithmus deaktiviert, der das Versenden von weiteren TCP-Segmenten verhindert, solange vorherige Segmente noch nicht quittiert worden sind. [NAGLEA]

In Abbildung 13 wird der Unterschied zwischen gesetzten TOS-Parametern und einer Übertragung ohne diese Veränderung sichtbar. Im oberen Teil der Abbildung wird deutlich, dass keine kontinuierliche Datenübertragung zwischen Client und Server stattfindet. Aus diesem Grund kommt es häufiger zu Spitzen im oberen Diagramm im Vergleich zum unteren Diagramm. Auch die Datenrate erreicht an diesen Stellen Spitzenwerte von 6,8 KB/s, während im unteren Diagramm lediglich maximal 3,9 KB/s eintreten.

[23] Siehe http://docs.oracle.com/javase/1.5.0/docs/api/java/net/Socket.html#setTrafficClass(int)
[24] Siehe http://docs.oracle.com/javase/1.5.0/docs/api/java/net/Socket.html#setTcpNoDelay(boolean)

Das Bildmaterial wurde im Zuge des dritten und fünften Testfalls aufgenommen. Zu diesem Zweck wurde der Reiter Network Statistics des DDMS innerhalb der Eclipse IDE während der Laufzeit der App im Lastfall aufgezeichnet. Die verschiedenen Tests zur Übertragung werden hier nicht aufgeführt, sondern wurden nur der Vollständigkeit halber erwähnt. Alle Tests befinden sich im Ordner „WLAN_Tests“ der beiliegenden CD des Anhangs.

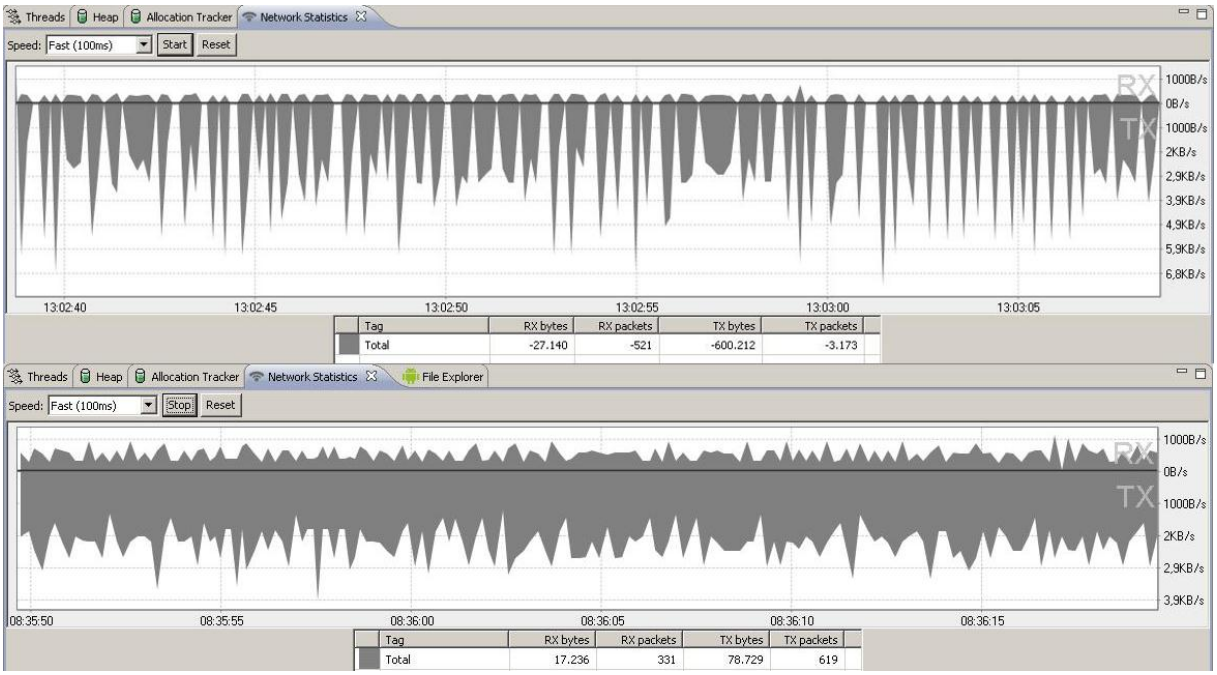


Abbildung 13 WLAN Übertragung, ohne (oben) und mit (unten) TOS Parametrierung

Aufgrund der Kompatibilität der Tablets mit der Bluetooth Version 2.1, bzw. Version 3.0, Technologie könnte diese Form der Kommunikation in einem konsekutiven Projekt implementiert und getestet werden. Auch die Verwendung von UDP/IP sollte in diesem Zusammenhang für den Einsatz am Demonstrator genauer überprüft werden. Wegen den bestehenden Anforderungen wurde in der vorliegenden Masterarbeit auf diese Punkte nicht weiter eingegangen.

Im darauffolgenden Kapitel wird auf die Struktur des Demonstrators bezüglich der softwaretechnischen Umsetzung, bzw. Implementierung und Modellierung, eingegangen. Dabei wird dann auch der Ablauf der Kommunikation im Zusammenhang mit den einzelnen Komponenten der Software diskutiert.

4.2.2 Struktur und Verhalten der Client-Komponente

Dieser Abschnitt geht auf die Anwendungsfälle des Demonstrators zusammen mit den einzelnen Klassen ein und beschreibt zudem, welche Aufgaben diese übernehmen. Darüber hinaus wird erläutert, welche Klassen, bzw. Komponenten, die Kommunikation zwischen Client und Server realisieren. Auch die Abläufe bezüglich des Systemverhaltens werden in diesem Kapitel ausführlich behandelt. Insbesondere das Thema Multithreading wird in diesem Zusammenhang mit der Kommunikation auf der Client-Seite betrachtet.

4.2.2.1 Logische Struktur des Demonstrators

Der logische Aufbau der Client-Anwendung wird in einem ersten Schritt anhand eines Paket-Diagramms in Abbildung 14 dargestellt und anschließend erklärt. Dabei wird auf die einzelnen Pakete im Projektverzeichnis eingegangen und die Paketstruktur zur allgemeinen Orientierung erläutert.

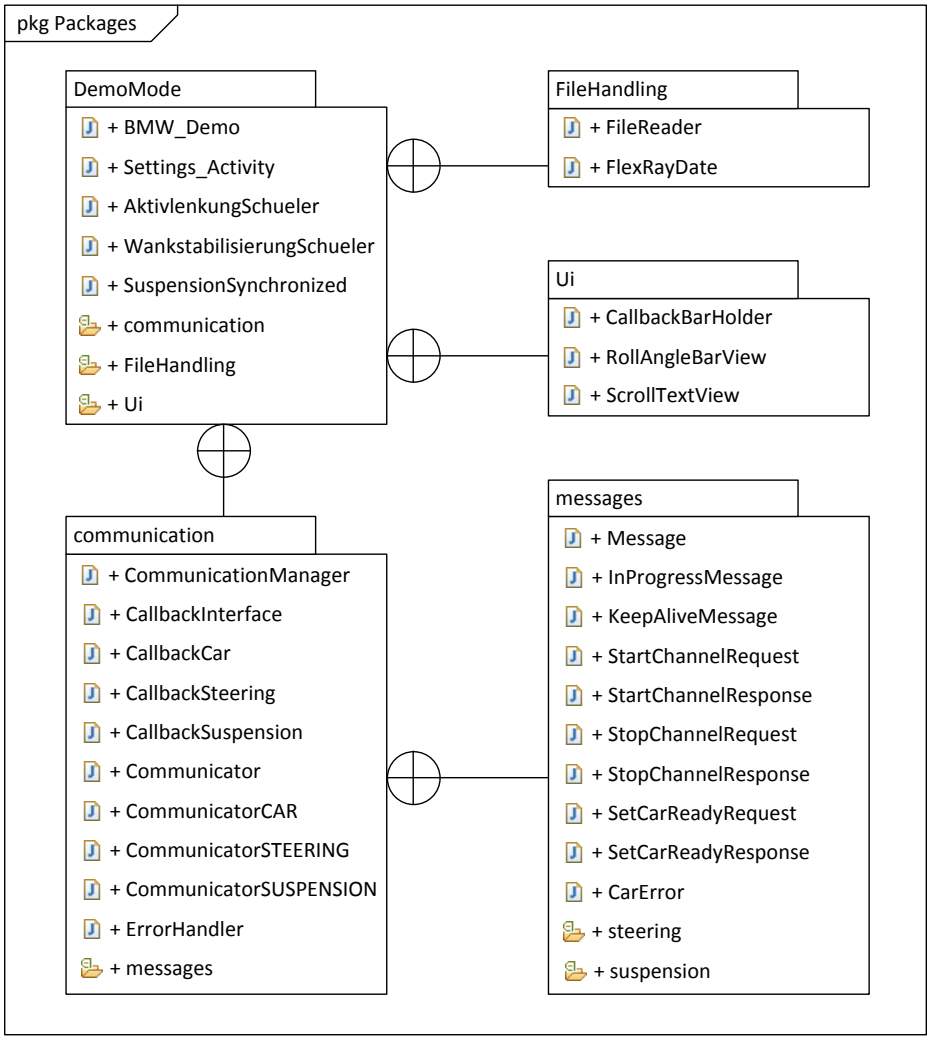


Abbildung 14 UML Paket-Diagramm zur Client-Anwendung

In Abbildung 14 wird die Hierarchie der Pakete des Android Projekts sichtbar. Im Paket *DemoMode* befindet sich die eigentliche Anwendung, bzw. die Activity, welche beim Start der Anwendung ausgeführt wird. Dabei handelt es sich um die Klasse *BMW_Demo*, die den Rahmen für alle Anwendungsfälle bereitstellt. Dabei werden die einzelnen Anwendungsfälle in jeweils einem Reiter präsentiert. In Abbildung 20 werden diese Reiter in der oberen Leiste mit den Namen der einzelnen Anwendungsfälle dargestellt. Je nach Anwendungsfall wechselt das Format der Darstellung zwischen Hoch- und Querformat (Portrait- und Landscape-Modus).

In Abbildung 20 wird der Anwendungsfall der Aktivlenkung dargestellt. Jeder Anwendungsfall, insbesondere der Einstellungsreiter, wird durch eine eigene Klasse repräsentiert. Auf die einzelnen Klassen wird bei der jeweiligen Beschreibung des Anwendungsfalls eingegangen. Zusätzlich beinhaltet das Paket *DemoMode* weitere Pakete. Das Paket *communication* besteht aus Klassen, welche die Funktionalität bezüglich der Kommunikation zwischen Client und Server zur Verfügung stellen. Für das Einlesen von externen Daten im Anwendungsfall der Wankstabilisierung wurde das Paket *FileHandling* vorgesehen. Des Weiteren wurden spezielle Komponenten der Benutzerschnittstelle in einem eigenen Paket, dem *Ui* Packet, gekapselt.

In erster Linie wird im weiteren Verlauf besonders auf das Paket *communication* und dessen darunterliegenden Pakete eingegangen. Dieser Anwendungsteil ist die Grundlage für eine Kommunikation zwischen einer Android-Anwendung und den BMW Steuergeräten. Aus diesem

Grund werden Teile des *communication* Pakets im Demonstrator und auch beim Prototyp in Zusammenhang mit Augmented Reality eingesetzt.

Das in Abbildung 14 abgebildete Paket *communication* beinhaltet die Klasse *CommunicationManager,* die als Kernkomponente betrachtet werden kann. Bei dieser Klasse handelt es sich um einen Thread [25], der innerhalb eines Anwendungsfalles gestartet wird. Der *CommunicationManager* dient als Schnittstelle zur Client-Server-Kommunikation und setzt das Entwurfsmuster des *Mediators* um. Diese Klasse kapselt und verdeckt somit die gesamte Funktionalität bezüglich der Kommunikation und stellt entsprechende Methoden zur Nutzung für andere Klassen zur Verfügung. Durch die Verwendung des *Mediator*-Entwurfsmusters[26] ergeben sich die folgenden Konsequenzen: Die Anzahl der Relationen zwischen den Klassen wird aufgrund dieser einzelnen Klasse stark reduziert. Aus diesem Grund wird die Struktur zur Kommunikation vereinfacht. Wiederverwendbarkeit in anderen Projekten ist aufgrund dieser zentralen Klasse einfach durchführbar. Der tatsächliche Ablauf der Kommunikation und die Interaktionen zwischen den Klassen werden mit Hilfe dieser Klasse abstrahiert und ähnlich einer Black-Box verdeckt zur Verfügung gestellt. Dies führt zu einer übersichtlichen Form der Benutzung durch die Klassen der Anwendungsfälle. Bei der aktuellen Implementierung wird der *CommunicationManager* allerdings nicht als *Observer*[27] eingesetzt um relevante Ereignisse an die entsprechende Klasse weiterzuleiten, sondern dies wird anhand von Callback-Aufrufen realisiert. Der *CommunicationManager* vermittelt lediglich während der Callback-Aufrufe direkt zwischen den betreffenden Klassen. [GAMMADP]

Zu diesem Zweck wurden die Klassen mit Präfix „Callback“ in deren Namen implementiert, welche die Schnittstelle *CallbackInterface* erweitern und für die Callback-Aufrufe bezüglich des jeweiligen Kanals eingesetzt werden. Verwendet ein Anwendungsfall einen bestimmten Kanal, muss diese Klasse zur Kommunikation die passende *CallbackInterface*-Erweiterung implementieren. Die aufrufenden Klassen dieser *CallbackInterface*-Implementierungen sind Erweiterungen der Klasse *Communicator*.

Bei der Klasse *Communicator* handelt es sich um einen Thread, der die TCP/IP-Nachrichtenübertragung über eine etablierte Socket-Verbindung behandelt. Da jeder Kanal mit spezifischen Nachrichten belegt wird, existiert für jeden Kanal eine eigene Erweiterung von *Communicator*. Diese Klassen tragen im Namen „Communicator“ als Präfix. Anhand der Implementierung einer *CallbackInterface*-Erweiterung kann sich eine Klasse für Benachrichtigungen, den entsprechenden Kanal betreffend, registrieren. Der Registrierungsvorgang wird durch den *CommunicationManager* vorgenommen. In Abbildung 17 wird unter anderem dieser Vorgang dargestellt. Aufgrund dieses Designs kann Logik, die in der Implementierung eines Anwendungsfalls zu einer bestimmten Nachricht hinterlegt ist, asynchron und ereignisgetrieben von einem anderen Thread aufgerufen werden. Auf diese Weise bleibt der Quellcode der Implementierungen der Anwendungsfälle bezüglich der Kommunikation übersichtlich. Darüber hinaus werden keine blockierenden Aufrufe in dem Thread der Anwendung ausgeführt, der von Android für die Darstellung der grafischen Benutzerschnittstelle eingesetzt wird.

[25] Bei einem Thread handelt es sich hier um eine Erweiterung der Klasse java.lang.Thread siehe http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/Thread.html

[26] Dieses Entwurfsmuster beschreibt eine Komponente, die geeignete Instanzen zusammenführt.

[27] Beschreibt das Entwurfsmuster zu einer Komponente, die automatisch Zustandsänderungen der überwachten Instanzen wiederspiegelt, indem diese unmittelbar benachrichtigt wird.

Eine weitere nennenswerte Klasse des Pakets *communication* ist die Klasse *ErrorHandler*. Entstehen während der Kommunikation auf einem Kanal serverseitig Fehler, werden diese anhand eines Fehlercodes der Client-Seite mitgeteilt. Somit ist es bei geeigneter Implementierung der Client-Seite möglich eine spezifische Fehlerbehandlung einzuleiten. Um dem Benutzer stichhaltige Hinweise zu einem Fehlercode liefern zu können, muss eine Abbildung, bzw. ein Mapping, zwischen Fehlercode und Fehlerhinweis für den Benutzer, durchgeführt werden. Diese Abbildung wird durch die Klasse *ErrorHandler* realisiert. Hierbei handelt es sich um eine zentrale Stelle für Einträge bezüglich zukünftiger Fehlercodes und deren korrespondierenden Fehlerhinweise. Die Verwendung der Klasse *ErrorHandler* kann somit bei zukünftigen Entwicklungen analog zu den bisherigen Implementierungen eingesetzt werden.

Um den Umgang im Quellcode mit JSON-Nachrichten zu vereinfachen, wurden diese in eigenen Klassen gekapselt. Somit wird die Repräsentation der Nachrichten als Java-Objekt ermöglicht. Diese entsprechen im weitesten Sinne POJOs[28], welche prinzipiell keinerlei Logik kapseln, sondern lediglich als Container für Daten dienen. Alle Klassen des *messages* Pakets, einschließlich der Klassen in den Subpaketen, erweitern die abstrakte Klasse *Message*. Im Wesentlichen besteht diese Klasse aus einer Sammlung von statischen Konstanten, welche als Auswahl für die Datenbelegung in einer Nachricht verwendet werden sollten. Damit ist die Klasse *Message* gleichzeitig als Austauschdokument zwischen Server-Entwickler und Client-Entwickler zu betrachten, da hier alle legitimen Ausprägungen der Schlüssel/Wert-Paar-Belegungen aufgelistet werden. Dieses Dokument sollte als Grundlage für zukünftige Entwicklungen am Nachrichtenumfang zum Konsentieren eingesetzt werden.

Alle genannten Klassen und die beschriebenen Zusammenhänge werden in Abbildung 41 im Anhang durch ein UML-Klassendiagramm dargestellt. Dabei werden die Abhängigkeiten zwischen Klassen der Android API nicht abgebildet. Auch wird aufgrund der Übersichtlichkeit überwiegend auf die Angabe von Attribute verzichtet. Methoden und Attribute werden lediglich angegeben, falls diese für das Verständnis zu den Abläufen relevant sind. Mechanismen der Android-Entwicklung werden wegen des Umfangs hier nicht erklärt. An dieser Stelle wird auf die entsprechende Fachliteratur verwiesen. Um den Fokus auf bestimmte Klassen lenken zu können werden im weiteren Verlauf Teile des gesamten Diagramms dargestellt und beschrieben.

[28] Der Begriff POJO wurde von Martin Fowler et al. im Datenbankumfeld geprägt. [FOWLER]

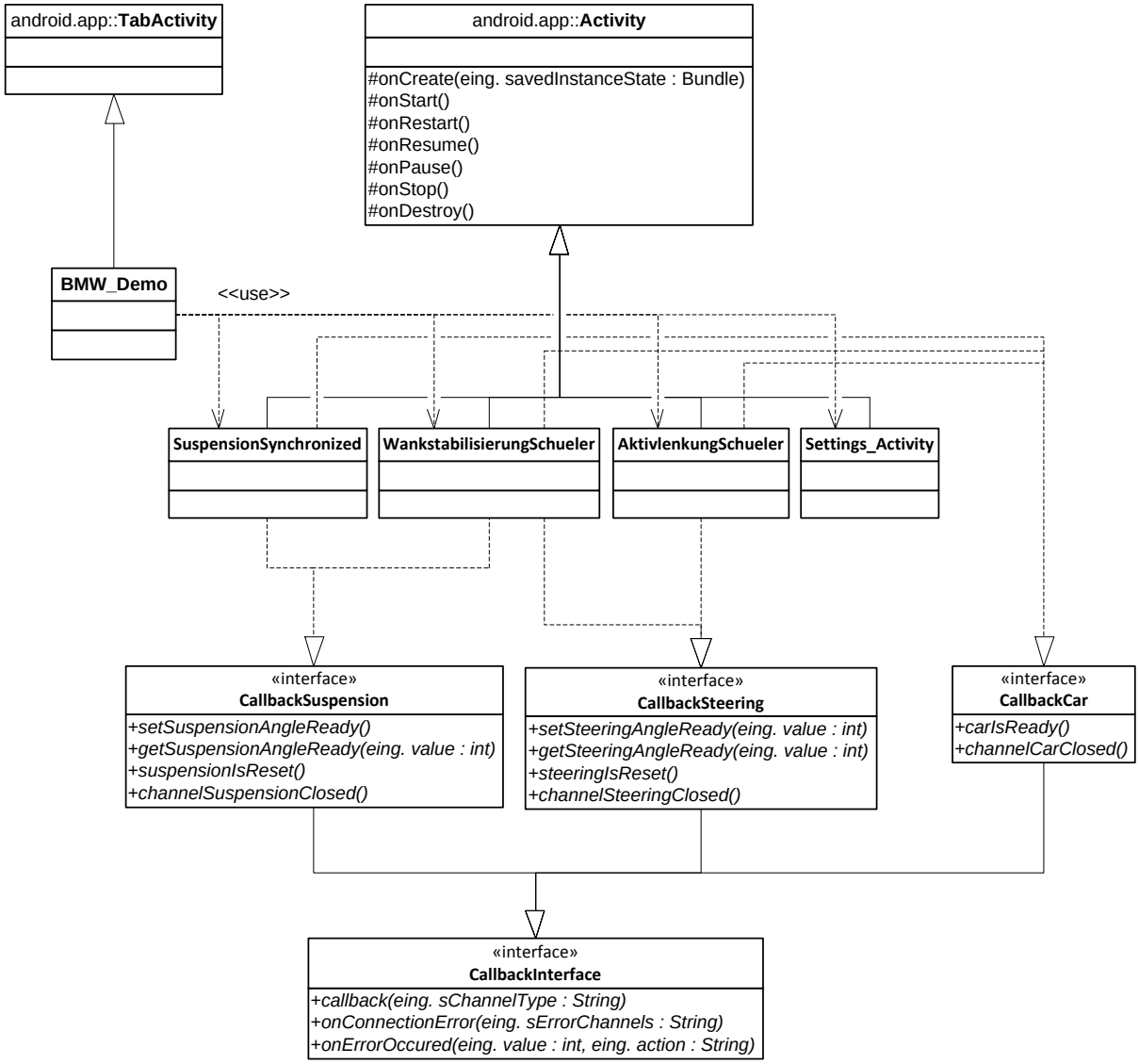


Abbildung 15 UML-Klassendiagramm, Auszug zu Anwendungsfällen und Callbacks

In Abbildung 15 werden in einem ersten Teil des Gesamtdiagramms die Zusammenhänge zwischen den Implementierungen der Anwendungsfälle, bzw. deren Activities, und den *CallbackInterfaces* dargestellt. Alle Reiter der Anwendungsfälle sind zentral als Spezialisierungen der Klasse *Activity* mittig dargestellt. Die *SuspensionSynchronized*- als auch die *WankstabilisierungSchueler*-Klasse verwenden zur Präsentation der Wankstabilisierung das ARS System. Zu diesem Zweck wird von diesen Klassen das *CallbackSuspension*-Interface implementiert. Das *CallbackSuspension*-Interface wie auch alle anderen Callback-Schnittstellen ist eine Spezialisierung des *CallbackInterface*-Interfaces. Von der Schnittstelle *CallbackInterface* wird die Implementierung allgemeiner Funktionalitäten verlangt. Dabei handelt es sich um Methoden, die es erlauben auf Verbindungsfehler oder auf Fehler der Steuergeräte zu reagieren. Mittels eines Aufrufs der *callback()*-Methode kann zudem auf das Ereignis reagiert werden, dass ein Kanal erfolgreich etabliert wurde. Mittels der Methoden in den spezialisierten Schnittstellen, wie *CallbackSuspension*, wird der Nachrichtenversandt, bzw. Nachrichtenempfang, in Methodenaufrufen gekapselt. Die Klasse *BMW_Demo*, die den Rahmen aller Anwendungsfälle repräsentiert, wird mit einer Abhängigkeit zu den Anwendungsfall-Klassen abgebildet. Diese schwache Verbindung wurde aufgrund des Aufrufmechanismus gewählt, der von Android verwendet wird. Hierbei handelt es sich um den Aufruf

der jeweiligen *Activity* mittels eines *Intents*[29]. Auf dieses Konzept wird im Zuge dieser Arbeit aufgrund des Umfangs nicht genauer eingegangen. Bei der zweiten Verbindung der Klasse *BMW_Demo* handelt es sich um eine Generalisierung zur Klasse *TabActivity*. *TabActivity* ist selbst indirekt eine Erweiterung der Klasse *Activity*, was in der Darstellung nicht explizit abgebildet wird. Auf die einzelnen Methoden der Klasse *Activity* der Android API wird hier nicht weiter eingegangen. Allerdings werden diese Methoden von den Spezialisierungen überladen, um entsprechend auf Ereignisse des Android Betriebssystems reagieren zu können.

Zur Kommunikation wird von jeder Anwendungsfall-Implementierung mindestens ein *CommunicationManager* gestartet. Dabei ist ein *CommunicationManager* gleichbedeutend mit der Gruppe aller Kanäle, die ein bestimmtes Automobil betreffen. Für die Klasse *SuspensionSynchronized*, die aufgrund des Anwendungsfalls bis zu drei F11 gleichzeitig verwalten muss, bedeutet dies, dass auch drei *CommunicationManager* gestartet werden müssen. Diese Abhängigkeit wird auf Abbildung 41 im Anhang dargestellt. Methoden, die das Starten und Beenden des *CommunicationManagers* betreffen, werden hier nicht abgebildet und erklärt.

Um die Abhängigkeiten der Klassen mit Bezug zum *CommunicationManager* genauer betrachten zu können, werden diese in Abbildung 16 am Beispiel der Klasse *CommunicatorSUSPENSION* dargestellt.

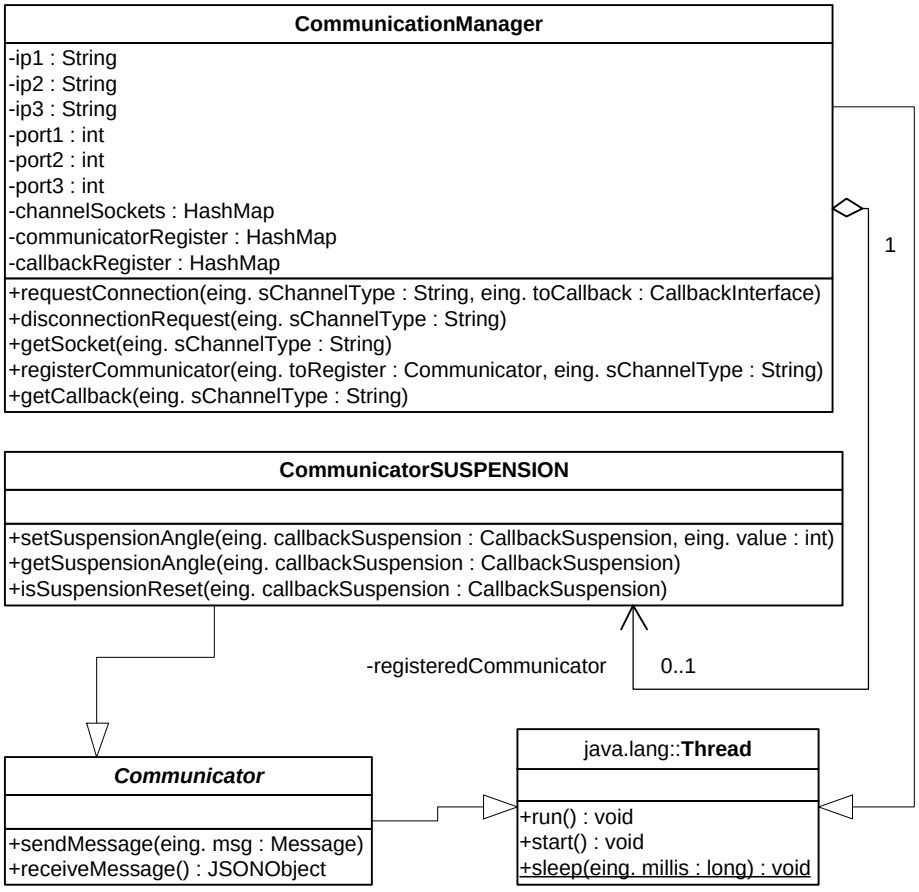


Abbildung 16 UML-Klassendiagramm, Auszug zur Kommunikation

[29] Siehe http://developer.android.com/reference/android/content/Intent.html

Die Klasse *CommunicatorSUSPENSION* wird als Bindeglied zur Kommunikation zwischen Client und Server auf dem SUSPENSION-Kanal eingesetzt. Über diesen Kanal werden Nachrichten übertragen, welche die Wankstabilisierung betreffen. Dabei gilt dieses Diagramm lediglich als Beispiel. In Abbildung 41 im Anhang werden die Abhängigkeiten aller relevanten *Communicator*-Erweiterungen dargestellt. Bei der abstrakten Klasse *Communicator* und der Klasse *CommunicationManger* handelt es sich um Spezialisierungen der Klasse Thread. Die Klasse *Communicator* beinhaltet allgemeine Methoden, welche zum Senden und Empfangen von Nachrichten eingesetzt werden. Die spezialisierte Klasse stellt Methoden zur Verfügung, welche die einzelnen Funktionen zur Manipulation der Aktuatorik in Form von Kommunikationsabläufen kapseln. Dabei übergibt die aufrufende Klasse eine Referenz auf sich selbst in Form des jeweiligen *CallbackInterfaces*. Wurde die Veränderung durchgeführt, wird das referenzierte Objekt zurückgerufen. Um einen Kanal aufzubauen, wird die *requestConnection()*-Methode des *CommunicationManager* aufgerufen. Analog zur Implementierung der *Communicators* wird auch hier eine Referenz als Parameter übergeben, die den Rückruf durch den *CommunicationManager* ermöglicht. Der erfolgreiche Aufbau eines Kanals erzeugt demnach einen Aufruf der *callback()*-Methode des entsprechenden *CallbackInterfaces*. Die Abhängigkeiten zwischen den *Communicator*- und *CallbackInterface*- Erweiterungen wird in Abbildung 16 nicht gezeigt. Erneut wird für einen Überblick zu diesem Zusammenhang auf Abbildung 41 im Anhang verwiesen. Um zwischen den beteiligten Klassen vermitteln zu können, werden vom *CommunicationManager* entsprechende Datenstrukturen, die sogenannten Register, gepflegt. Hat ein *Communicator* seinen Dienst aufgenommen, indem er gestartet wurde, registriert sich dieser über einen Aufruf der Methode *registerCommunicator()* selbst bei seinem *CommunicationManager*. Jeder *Communicator* verfügt über eine *stopChannelRequest()*-Methode, um den Vorgang zum Schließen eines Kanals anzustoßen. Diese Methode wurde, zum Zweck der Übersichtlichkeit, nicht in Abbildung 16 gezeigt.

Im nächsten Abschnitt wird auf das Verhalten und die Interaktion der einzelnen Klassen im Bezug zur Kommunikation eingegangen.

4.2.2.2 Verhalten und Abläufe

Dieses Kapitel beschreibt die Interaktionen zwischen den Klassen bei Abläufen der Client-Server-Kommunikation. Dazu wird der Ablauf der Kommunikation in verschiedene Phasen eingeteilt und jede Phase einzeln betrachtet. In der ersten Phase wird der Verbindungsaufbau beschrieben, welcher bei der Umsetzung jedes Anwendungsfalls zu Beginn durchgeführt werden muss um eine Übertragung zwischen Client und Server zu gewährleisten.

4.2.2.2.1 Verbindungsaufbau

Abbildung 17 zeigt in einem UML-Sequenzdiagramm den Ablauf bezüglich des Verbindungsaufbaus in einer allgemeinen Anwendungsfall-Implementierung. Diese allgemeine Klasse wird hier der Einfachheit halber mit *Activity* bezeichnet. Der dargestellte Ablauf wird in allen Anwendungsfall - Implementierungen eingesetzt. Weitere Kanäle werden spezifisch für den jeweiligen Anwendungsfall zusätzlich aufgebaut. In Abbildung 17 wird auch der Registrierungsvorgang am Beispiel des *CommunicatorCAR*-Objekts abgebildet. An dieser Stelle wird darauf hingewiesen, dass alle drei dargestellten Klassen in jeweils einem eigenen Thread ausgeführt werden.

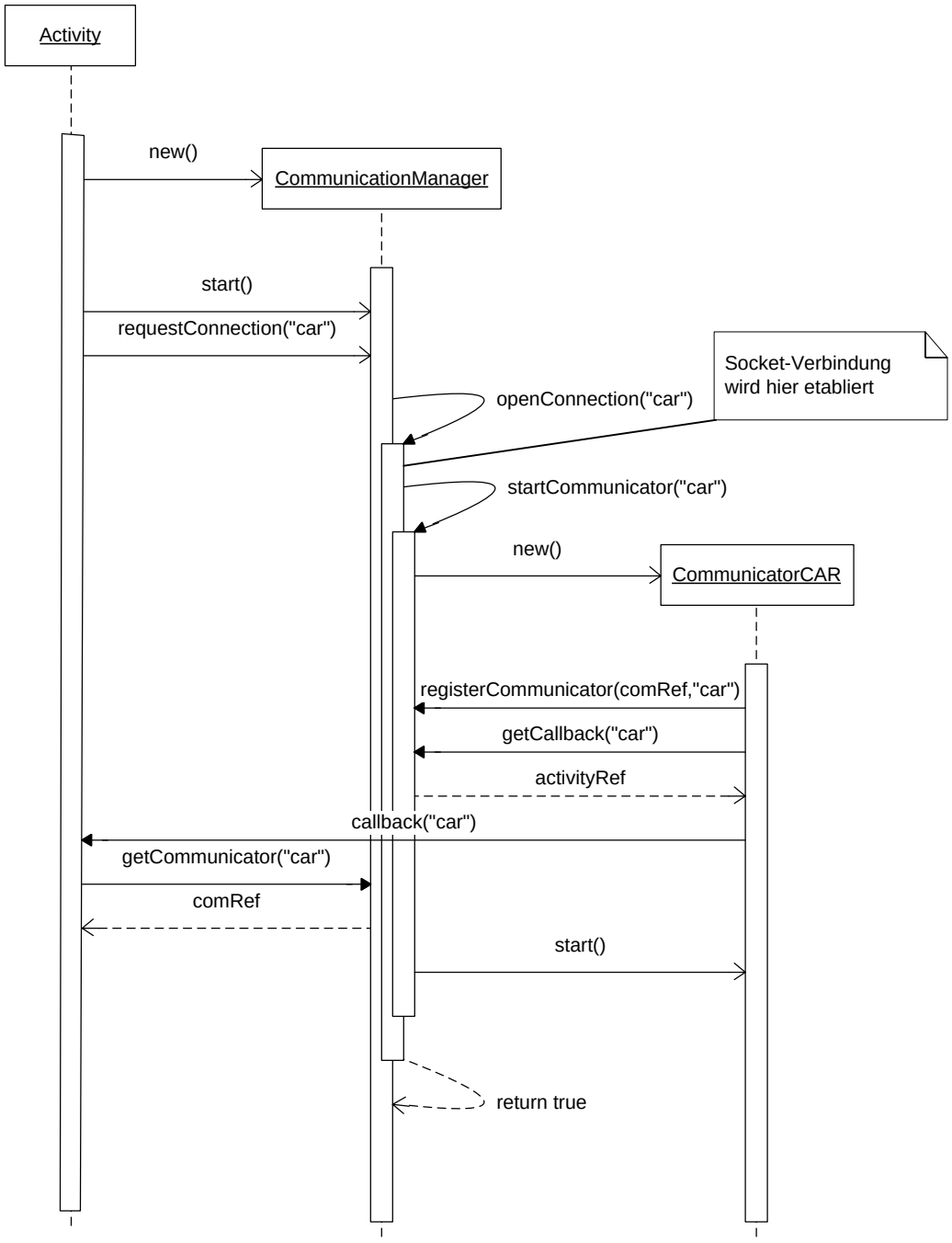


Abbildung 17 UML-Sequenzdiagramm, Verbindungsaufbau im Demonstrator

Zu Beginn wird vom Server auf einem vorher festgelegten Port auf eine eingehende Verbindung gewartet. Verbindet sich ein Client mit dem Wissen der Server-IP und des Ports mit dem Server, wird in jedem Anwendungsfall immer als erstes der CAR-Kanal aufgebaut. Dieser Kanal ist im Allgemeinen eine Ausnahme von der Regel, dass ein Kanal immer an genau ein Steuergerät gekoppelt ist. Mit diesem Kanal werden Funktionen, die nicht klar zugeordnet werden können, zusammengefasst. Der Verbindungsaufbau des CAR-Kanals wird als Grundlage für ein sicheres Systemverhalten vorausgesetzt. Anhand dieses Kanals wird durch eine entsprechende Anfrage das gesamte System zurückgesetzt, bzw. alle Aktuatoren in ihren Initialzustand zurückgesetzt.

Port und IP für das jeweilige Fahrzeug werden durch die Einstellungen des Demonstrators festgelegt und in der jeweiligen *CommunicationManager*-Instanz hinterlegt. Die Einstellungen können im Demonstrator durch die Klasse *Settings_Activity* vorgenommen werden, für die ein eigener Reiter in der GUI vorgesehen wurde.

Bei dem Start einer Anwendungsfall-Implementierung, wie beispielsweise der Klasse *WankstabilisierungSchueler,* wird zunächst ein *CommunicationManager*-Thread gestartet. Durch den Aufruf der *requestConnection()*-Methode wird hier die Verbindung anhand eines CAR-Kanals angefragt. Dabei handelt es sich nicht um einen blockierenden Aufruf. Sollte eine Verbindung über den CAR-Kanal erfolgreich etabliert worden sein, wird die *Activity* mittels des Aufrufs der Methode *callback()* vom Thread der *Communicator*-Erweiterung, hier *CommunicatorCAR*, zurückgerufen. Damit die *callback()*-Methode durch den *Communicator* aufgerufen werden kann, muss diesem zuvor eine entsprechende Referenz übergeben werden. Im Konstruktor der *CommunicatorCAR*-Klasse wird vom *CommunicationManager* das korrespondierende *CallbackInterface* erfragt, welches bei bestimmten Ereignissen aufgerufen werden soll. Dabei wird das Register *callbackRegister* über die *getCallback()*-Methode nach dem konkreten Kanal abgefragt. In Abbildung 17 wird die Referenz auf die *Activity* als Rückgabewert *activityRef* dargestellt.

Zusätzlich registriert sich der *CommunicatorCAR*-Thread zu Beginn seiner Lebenslinie selbst bei dem *CommunicationManager*. Somit kann vom *CommunicationManager* der *Communicator* zu einem bestimmten Kanal von außen, hier durch die *Activity*, erfragt werden. Der *Communicator* wird dabei anhand der *getCommunicator()*-Methode angefragt. Hier wird darauf hingewiesen, dass diese Anfrage innerhalb des Methodenrumpfs der *callback()*-Methode durchgeführt wird. Damit ist der *Activity,* unmittelbar nach dem Verbindungsaufbau des Kanals, der korrespondierende *Communicator* bekannt und es können Anfragen an diesen gerichtet werden. In der Darstellung wird die Referenz auf das *CommunicatorCAR*-Objekt mit *comRef* bezeichnet. Mittels der *start()*-Aufrufe wird lediglich darauf hingewiesen, zu welchen Zeitpunkten die Threads jeweils ihren Betrieb, im Sinne des Ablaufs der *run()*-Methode, aufnehmen.

4.2.2.2.2 Anfrage zur Manipulation

Um die Manipulation der Steuergeräte zu beschreiben, wird in diesem Kapitel das Beispiel aus dem vorherigen Kapitel fortgesetzt. Bei Abbildung 18 wurde der Vorgang der Kommunikation stark eingeschränkt. Für eine Beschreibung der Abläufe innerhalb der Klasse *Communicator* und dessen Erweiterungen wird auf deren Quellcode verwiesen. Hier wird lediglich grundsätzlich auf die Funktionsweise der *Communicator*-Implementierungen eingegangen.

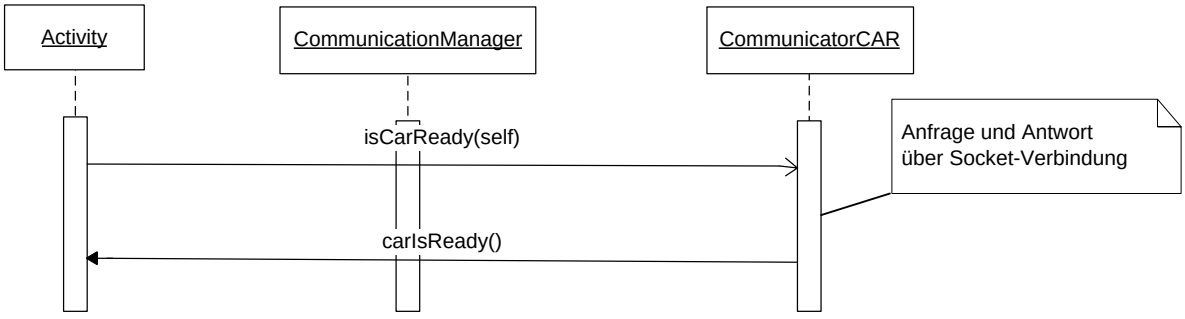


Abbildung 18 UML-Sequenzdiagramm, Manipulationsanfrage am Demonstrator

In Abbildung 18 wird an den Ablauf von Abbildung 17 angeknüpft und durch einen Aufruf der *isCarReady()*-Methode am Communicator der Nachrichtenversandt zum Server ausgelöst. Dieser Vorgang erfolgt dabei asynchron, da lediglich die nächste zu versendende Nachricht des Communicators in einem Eintrag bis zum eigentlichen Versand vorgemerkt wird. Die Abarbeitung dieser Anfragen wird anschließend vom *CommunicatorCAR*-Thread eigenständig vorgenommen und beim Empfang einer entsprechenden Antwortnachricht der Auslöser zurückgerufen. Der Rückruf erfolgt durch einen Aufruf der *carIsReady()*-Methode der Anwendungsfall-Implementierung durch den *CommunicatorCAR*-Thread. Hierbei handelt es sich dann um einen blockierenden Aufruf für den *CommunicatorCAR*-Thread. In der Darstellung wurden die Vorgänge innerhalb der *Communicator* Klasse mittels Abstraktion verdeckt. Dabei wird durch die *run()*-Methode dieser Klassen das Verhalten des Threads definiert. Zyklisch werden in dieser Methode Versand- und Empfangsphasen ausgeführt. Erst wenn der Thread erneut in die Versandphase eintritt wird auch die vorgemerkte Nachricht verschickt. Dieser Mechanismus wird analog bei allen anderen *Communicator*-Erweiterungen eingesetzt.

Im folgenden Kapitel werden, auf diesem Kapitel aufbauend, die Fehlerquellen bezüglich der Client-Server-Kommunikation beschrieben und erläutert wie diese behandelt werden.

4.2.2.2.3 Verbindungsfehlerfälle

Grundsätzlich können zwei Arten von Fehlerfällen unterschieden werden: Bei der ersten Fehlerklasse handelt es sich um Fehler, die bei einer bestehenden Verbindung zum Server auftreten und einen Fehler der Steuergerätefunktionalität implizieren. Diese können beispielsweise durch fehlerhafte Bedienung durch den Benutzer auftreten. Einer dieser Fehler tritt im Zusammenhang mit der Personensicherheit im Kontext des Aktivlenkungsanwendungsfalls auf. Dabei darf sich das Lenkrad nicht automatisch verdrehen, wenn eine Person in das Lenkrad greift. Zu diesem Zweck wechselt die Serverseite bei dieser Situation in eine Rückfallebene, in der sich das Lenkrad frei bewegen lässt und in der die Aktuatorik keinen Einfluss nimmt. Aus Sicherheitsgründen soll erst nachdem dieser Fehler manuell bestätigt wurde, eine erneute automatische Manipulation ermöglicht werden. Um entsprechende Fehlerfälle quittieren zu können, werden dementsprechend Fehlercodes vom Server an den Client versandt. Anschließend wird der Benutzer über die GUI auf den Fehlerfall hingewiesen und kann die Kenntnisnahme per Knopfdruck manuell bestätigen. Durch Nutzung der Klasse *ErrorHandler*, die in Kapitel 4.2.2.1 bereits beschrieben wurde, kann zum jeweiligen Fehlercode eine geeignete Ausgabe erzeugt werden, die den Nutzer auf die Fehlerquelle hinweist. Zur spezifischen Fehlerbehandlung wird die *onErrorOccured()*-Methode des *CallbackInterfaces* beim Anfragensteller aufgerufen. In diesem Aufruf wird dann auch die Klasse *ErrorHandler* eingesetzt, um die GUI bezüglich des Fehlerhinweises anzupassen. Ein Beispiel zu diesem Verhalten wird in Kapitel 4.2.3.2 beschrieben, welches sich mit dem Anwendungsfall der Aktivlenkung befasst.

Bei der zweiten Fehlerklasse handelt es sich um Fehler, welche den Verbindungstatus an sich betreffen. Anders als bei den Fehlercodes werden diese Fehler nicht vom Server zum Client kommuniziert, sondern direkt auf Clientseite festgestellt. Dabei können wiederum zwei Arten von Fehlern unterschieden werden. Wird während der Kommunikation die Verbindung auf einem Kanal unterbrochen, wird die Callback-Methode *onConnectionError()* des *CallbackInterfaces* aufgerufen. Der gleiche Fehler tritt auf wenn der Server die Socket-Verbindung nicht akzeptiert, bzw. nicht unter den angegebenen Einstellungen zu erreichen ist. Beim Fehler bezüglich der Erreichbarkeit wird nach einem Timeout ein Aufruf ausgelöst.

Im nächsten Kapitel wird auf den Verbindungsabbau eingegangen, der durchgeführt wird, um die einzelnen Kanäle der Client-Server-Kommunikation zu schließen.

4.2.2.2.4 Verbindungabbau

Das Schließen der einzelnen Kanäle ist aufgrund der Ressourcenknappheit bezüglich der Akkulaufzeit des Tablets und der Prozessorleistung erforderlich. Somit werden Kanäle unmittelbar geöffnet und geschlossen, falls über den jeweilige Kanal in Kürze kommuniziert werden soll, bzw. in nächster Zeit keine weitere Kommunikation mehr über diesen Kanal erfolgt. Diese Strategie stellt auch im Hinblick auf das Beenden der einzelnen Threads eine Herausforderung beim Schließen der Anwendung, bzw. bei dem Wechsel der Anwendungsfälle, dar. Da für jeden Kanal ein eigener Thread unterhalten wird, muss dieser auch an entsprechender Stelle unmittelbar beendet werden. Auf diese Weise können Ressourcen eingespart werden. Für das Schließen eines Kanals muss eine Nachricht an die Server-Seite versandt werden, welche die Absicht des Verbindungsabbaus impliziert. Zu diesem Zweck wird bei jeder *Communicator*-Erweiterung die abstrakte *stopChannelRequest()*-Methode überschrieben. Diese Methode wurde aus Gründen der Übersichtlichkeit nicht in Abbildung 16 in der abstrakten Klasse *Communicator* dargestellt.

Abbildung 19 zeigt ein UML-Sequenzdiagramm auf, welches den Verbindungsabbau anhand der dabei ausgeführten Methodenaufrufe darstellt.

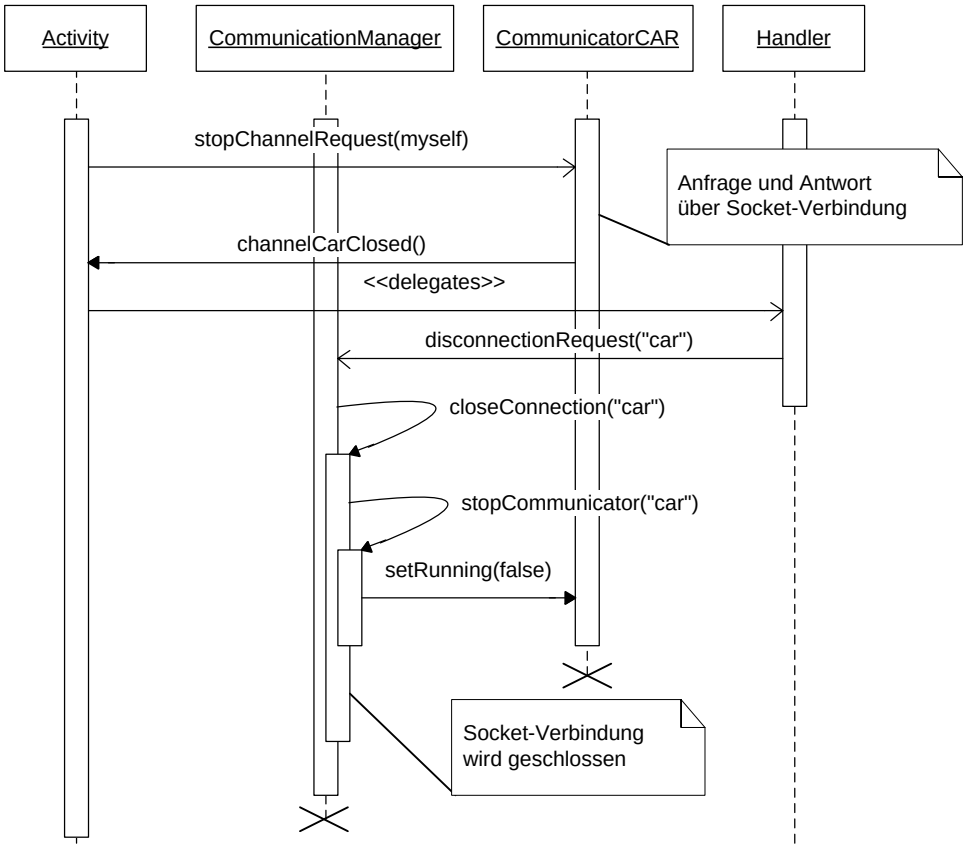


Abbildung 19 UML-Sequenzdiagramm, Verbindungsabbau am Demonstrator

Um einen Kanal zu schließen, wie hier im Beispiel der CAR-Kanal, wird zunächst die Methode *stopChannelRequest()* aufgerufen. Analog zu anderen Anfragen wird dadurch asynchron eine

Nachricht an den Server versandt. Nachdem diese Nachricht vom Server quittiert wurde, ruft der *CommunicatorCAR* die *channelCarClosed()*-Methode auf. Wurde die Methode *channelCarClosed()* aufgerufen wird, damit ausgewiesen, dass die Verbindung bereits serverseitig abgebaut wurde. Gleichzeitig wird damit sichergestellt, dass der Server auf diesem Kanal keine weiteren Nachrichten an den Client überträgt. Aufrufe, die dazu führen, dass Aktivitäten auf einem anderen Thread asynchron ausgeführt werden, sind im Diagramm mit einem ungefüllten Pfeilkopf abgebildet. Grund dafür ist, dass prinzipiell der aufrufende Thread für die Folgeaktivitäten nicht blockiert wird. Diese Aufrufe werden analog zur Ausführung der Methode *isCarReady()* durchgeführt, die in Kapitel 4.2.2.2.2 im Zusammenhang mit der *run()*-Methode im *Communicator*-Thread beschrieben wurde.

Beim Aufruf der Methode *channelCarClosed()* wird ein besonderes Konzept der Android API verwendet, das es erlaubt Quellcode auf einen anderen Thread auszulagern und dort auszuführen. Dabei wird die Klasse *Handler*[30] und deren *post()*-Methode eingesetzt. Dadurch wird allerdings keine Aussage darüber getroffen, zu welchem Zeitpunkt der delegierte Code von dem *Handler*-Thread ausgeführt wird. Um diesen Vorgang zu kennzeichnen, wurde auf die Stereotypen von UML zurückgegriffen, um die Sprache zu erweitern. Das verlagern des Codes wird in der Zeichnung mit einem unausgefüllten Pfeil und der Beschriftung „<<delegates>>“ dargestellt.

Durch den *Handler* wird anschließend die *disconnectionRequest()*-Methode des *CommunicationMangers* ausgeführt. Dieser Vorgang des Abmeldens der verwendeten Ressourcen eines Kanals wird auch hier asynchron abgewickelt. Zusätzlich wird vom *Handler* an den *CommunicationManager* signalisiert, dass sich dieser Beenden soll. Solange eine Anfrage auf das Abmelden eines Kanals besteht, beendet sich der Manager auch nicht selbst. Dies geschieht erst, nachdem sichergestellt wurde, dass momentan keine Anfrage dieser Art mehr vorliegt. Diese Aufrufe, welche das Beenden des Managers auslösen, werden in dem UML-Diagramm von Abbildung 19 nicht dargestellt. Unmittelbar nach diesen Ausführungen ist der delegierte Quellcode abgearbeitet und der *Handler* kann weitere Aufgaben entgegennehmen.

Der *disconnectionRequest()*-Aufruf erzeugt einen Ablauf analog zum Anmeldevorgang der Methode *requestConnection()*. Nach dem Aufruf der Methode *setRunning()* innerhalb der *stopCommunicator()*-Methode wird bei dem nächsten Zyklus des *CommunicatorCAR*-Threads dessen Tätigkeiten eingestellt und auch die Ein- und Ausgabeströme zur Datenübertragung freigegeben. Unterhalb der Aufrufstruktur der *closeConnection()*-Methode werden darüber hinaus die Datenstrukturen des *CommunicationManagers* bereinigt. Nach Abschluss dieses Abmeldevorgangs beendet sich, analog zum *CommunicatorCAR*-Thread, der *CommunicationManager*-Thread selbst.

In allen Anwendungsfällen wird ähnlich zu Beginn des einzelnen Falles verfahren, wie in den vorherigen Kapiteln dargestellt. Erst nachdem der CAR-Kanal auf- und wieder abgebaut wurde, werden weitere Kanäle beim *CommunicationManager* akquiriert. Im Gegensatz zum letzten UML-Sequenzdiagramm, welches das Schließen der Anwendung, bzw. eines Anwendungsfalles, abdeckt, wird beim Start eines Anwendungsfalles der *CommunicationManager* nicht beendet. Bei einem Anwendungsstart wird durch die Methode *channelCarClosed()* nach dem Schließen des CAR-Kanals unmittelbar ein anderer Kanal mittels *requestConnection()*-Methode geöffnet.

Im nächsten Kapitel wird auf die umgesetzten Anwendungsfälle eingegangen und diese anhand ihres Verhaltens in der Demonstrator-Applikation auf Benutzerschnittstellenebene beschrieben.

[30] Siehe http://developer.android.com/reference/android/os/Handler.html

4.2.3 Anwendungsabläufe und Screenflow

Auf den kompletten Screenflow wird in diesem Kapitel aufgrund des Umfanges nicht eingegangen. Für eine vollständige Übersicht zum Ablauf der Anwendung wird auf das entsprechende Diagramm der Abbildung 40 im Anhang verwiesen. In den folgenden Unterkapiteln wird jeweils der Anwendungsfall und die Intention zur Umsetzung erklärt. Begonnen wird mit den Einstellungen der Applikation, die erforderlich sind, um eine Kommunikation zwischen Client und Server zu ermöglichen.

4.2.3.1 Einstellungen

Zur erfolgreichen Übertragung von Nachrichten zwischen Tablet und Fahrzeug, bzw. dessen Server, ist es notwendig, dass die IP-Adresse und der Port des Servers in den Einstellungen eingetragen werden. Wichtig sind in erster Linie die Einstellungen für den ersten BMW. Die Einstellungen für den zweiten und dritten BMW sind lediglich für den Anwendungsfall „Synchrones Wanken“ des ersten Reiters relevant. Da dieser Anwendungsfall sicherheitskritisch bezüglich der geregelten Kommunikation ist, muss sichergestellt werden, dass dieser Anwendungsfall lediglich ausgeführt werden kann, wenn die beiden anderen Tablets nicht aktiv sind. Nicht aktiv bedeutet in diesem Zusammenhang, dass von den Geräten keine Steuerungsanfragen an eines der Fahrzeuge entsendet werden. Um den Anwendungsfall „Synchrones Wanken“ nutzen zu können, musste vom Standpersonal am BMW-Familientag sichergestellt werden, dass lediglich ein Tablet aktiv ist. Die anderen Tablets wurden während dieser Demonstration ausgeschaltet. Die Funktionalität des Anwendungsfalls „Synchrones Wanken“ muss anschließend über den Knopf neben der Beschriftung „Synchrones Wanken App aktiv“ aktiviert werden. Dabei erfolgt eine Passwortabfrage[31], damit Benutzer nicht aus Versehen diese Anwendung aktivieren und starten können. Mittels des zweiten Knopfes neben der Beschriftung „Server Informationen anzeigen“ können die eingestellten IP-Adressen bei dem jeweiligen Anwendungsfall ein-, bzw. ausgeblendet, werden. Abbildung 40 im Anhang zeigt den Reiter zu den Einstellungen im rechten oberen Bereich.

4.2.3.2 Aktivlenkung

Im Anwendungsfall der Aktivlenkung soll dem Benutzer vor Augen geführt werden, dass innerhalb des Fahrzeugs Aktuatorik verbaut wurde, die es erlaub den Fahrer aktiv bei dessen Lenktätigkeiten zu unterstützen. Dabei ist das Maß der Abweichung vom tatsächlich umgesetzten Lenkwinkel durch den Fahrer und der korrespondierenden Veränderung des Radwinkels vom Kontext der Fahrt abhängig. Somit verändert sich dieses Maß zusammen mit der Geschwindigkeit des Automobils.

Da es aus Gründen der Sicherheit nicht erlaubt war das fahrende Fahrzeug über eine Funkverbindung fernzusteuern und die notwendigen Geschwindigkeiten des Fahrzeugs für entsprechende Effekte nicht real wiedergegeben werden konnten, musste diese Einflussgröße aus der Betrachtung ausgenommen werden. Der Benutzer sollte demnach einzelne Effekte mit klaren Unterschieden im Lenkverhalten anhand verschiedener Modi erfahren. Momentan wird von der Demonstrator-Anwendung allerdings nur ein Modus unterstützt. Eine Erweiterung der Anwendung auf mehrere Modi ist möglich, indem erfasste Lenkwinkel des Tablets vor dem Nachrichtenversand um einen, für den Modus spezifischen, Faktor skaliert werden. Dadurch kann anschließend das Verhalten der Aktivlenkung in verschiedenen Situationen an den Benutzer herangetragen werden. Durch die Abbildung eines Lenkrads soll bei diesem Anwendungsfall der Benutzer aus der Intuition heraus in die Lage versetzt werden, mit der Anwendung umzugehen, bzw. diese zu bedienen. Um die

[31] Als Passwort wurde in der aktuellen Version die Zeichenkette „syncit“ vergeben.

Anwendung zu starten, muss zunächst aus Sicherheitsgründen die entsprechende Aktuatorik zurückgesetzt werden. Dies wird über den Knopf „BMW zurücksetzten“ ausgelöst. An dieser Stelle wird darauf hingewiesen, dass die Steuerelemente dieses Anwendungsfalls immer an der gleichen Stelle verbleiben. Auch ein Informationstext wird angezeigt, der den Benutzer auf den nächsten durchzuführenden Schritt hinweist. In Abbildung 20 wird oben im Bild der Knopf als auch der Informationstext dargestellt.

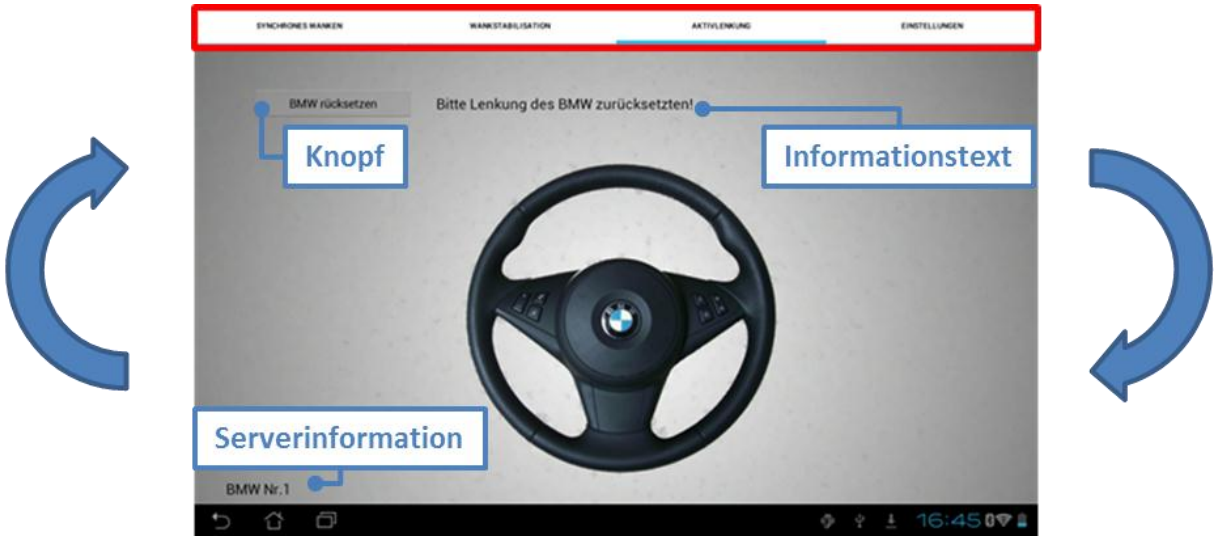


Abbildung 20 Demonstrator Aktivlenkung, Reiterleiste in roter Farbe

Aufgrund der Tatsache, dass zum Aufnahmezeitpunkt die Einstellung „Server Informationen anzeigen“ deaktiviert war, wird bei der Abbildung in der linken unteren Ecke lediglich „BMW Nr.1“ statt der IP-Adresse des BMWs angezeigt. Wurde die Aktuatorik zurückgesetzt, werden Lagedaten des Tablets bezüglich der Gierachse an den Server über den STEERING-Kanal übertragen. In Abbildung 20 wird mit blauen Pfeilen die Drehbewegung dargestellt, welche das Tablet erfahren muss, damit eine Lenkbewegung nach rechts am realen Lenkrad des Fahrzeugs resultiert. Der Benutzer hält das Tablet also im Querformat vor sich als hätte dieser das dargestellte Lenkrad in den Händen. Durch die Vorarbeit von Lukas Obkircher und dessen Prototypen konnte die Verwendung der inertialen Sensorik des Tablets relativ schnell für diesen Anwendungsfall implementiert werden. Zur Lagebestimmung wurden Daten des Beschleunigungs-, wie auch des Drehratensensors erfasst, anhand eines Filter-Algorithmus gefiltert und mittels einer passenden Gewichtung miteinander kombiniert. An dieser Stelle wird für eine exakte Beschreibung auf den Quellcode der Methode *Winkelbestimmen()* der Klasse *AktivlenkungSchueler* und dessen Kommentierung verwiesen. Aufgrund der Vorgaben an der Aktuatorik musste auch ein Filter verwendet werden, um zu hohe Winkelgeschwindigkeiten zu verhindern. Bei zu schnellen Veränderungen der Winkelwerte wurde während des Entwicklungszeitraums häufig der Fehlercode 300, bzw. EPS Failsafe, ausgelöst, der auch beim Hineingreifen in das Lenkrad durch den Benutzer auftritt. Als Fehlerhinweis wird dabei der Informationstext „EPS FAILSAFE EPS ist im Notlauf (>10 Nm Lenkmoment, >400°/s)“ angezeigt. Grund für diesen Fehler war, dass bei einer zügigen Lenkbewegung von einem Extremwert zum anderen ein hohes Trägheitsmoment am Lenkrad auftritt. Ein weiterer Grund für die Reglementierung der Lenkwinkelanfragen war die Spezifikation der Aktivlenkung von BMW. In der Spezifikation wurde die maximale Winkelgeschwindigkeit auf 400 Grad pro Sekunde festgelegt. Um diese Fehler zu vermeiden, wird der Ausgabewert nur in gefilterter Form versandt. Dazu wird die Steigung der einzelnen Winkelanfragen auf ein Maximum beschränkt.

Während des Entwicklungszeitraums wurden zu Testzwecken unterschiedliche Filter zur Glättung der Winkelwerte eingesetzt. Darunter wurden IIR Filter mit n-ter Ordnung und ein Butterworth-Filter der zweiten Ordnung als Tiefpassfilter verwendet, deren Quellcode innerhalb der Methode *Winkelbestimmen()* auskommentiert wurde. Diese Filter werden im Zuge dieser Arbeit nicht genauer erläutert. Stattdessen wird auf einschlägige Fachliteratur verwiesen. Im Zentrum zu diesen Implementierungen steht die Methode *discreteLinearFilterValue()*, welche den nächsten gefilterten Wert eines diskreten linearen Filters berechnet. Ein IIR Filter wurde gegenüber FIR Filtern bevorzugt eingesetzt, da diese mit wesentlich weniger Parametern und einem geringeren Rechenaufwand den gleichen Effekt erzielen. [DATAMI]

Testreihen zu den einzelnen Filtern mit unterschiedlicher Parametrierung befinden sich zusammen mit der jeweiligen Beschriftung auf der CD im Anhang im Verzeichnis „IIR-Filter_Tests“ in mehreren Microsoft Excel Dateien. Ein Auszug daraus wird in Abbildung 21 dargestellt. In dieser Abbildung wird die Begrenzung der Steigung anhand eines Testfalls gezeigt. Dieser Filtereffekt wird auch bei der aktuellen Version der Implementierung eingesetzt und beschränkt die Winkelbeschleunigung auf 390 Grad pro Sekunde. Diese Filterung optimiert die Störanfälligkeit bei starken Lastwechseln durch den Benutzer, ohne dabei eine Phasenverschiebung der Ausgabewinkel zu bewirken. Somit wirkt die Lenkbewegung auf den Benutzer weitestgehend direkt ohne erhebliche Latenzzeiten aufzuweisen.

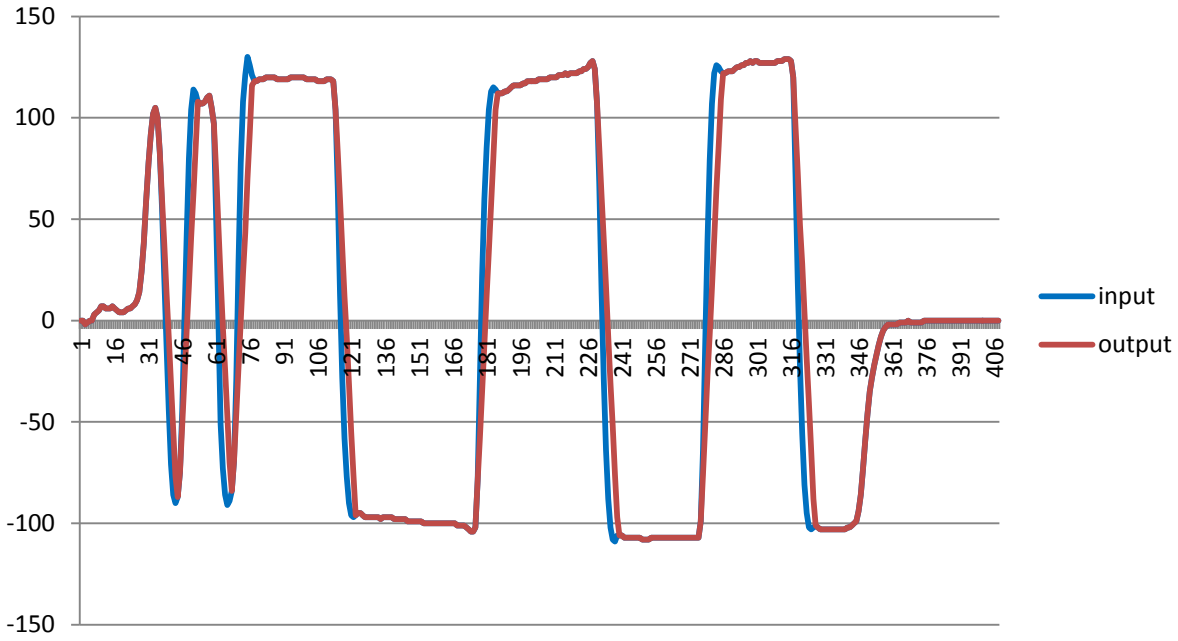


Abbildung 21 Testreihe, gefilterte (rot) und ungefilterte (blau) Winkelwerte

Bei dieser Darstellung wird in blauer Farbe der ungefilterte Wert und mit roter Farbe der gefilterte Wert abgebildet, welcher auch an den Server versandt wird.

Eine weitere Einschränkung bei diesem Anwendungsfall betrifft die benötigte Verarbeitungszeit durch den Server. Hier wurde als Richtlinie eine minimale Zykluszeit von 50 Millisekunden für den Nachrichtenversandt des Clients festgelegt. Auf diese Weise soll ein Überlauf in der Empfangswarteschlange des Servers verhindert werden. Diese Vorgabe gilt auch für alle anderen Anwendungsfall-Implementierungen des Demonstrators.

Treten dennoch Fehler während der Benutzung der Anwendung auf, wie beispielsweise durch einen Fehlercode des Steuergeräts oder aber auch durch einen Verbindungsabriss, so können diese per Knopfdruck bestätigt werden und nach dem Zurücksetzten des Fahrzeugs kann die Benutzung fortgesetzt werden.

Durch die aktuelle Implementierung konnte die besondere Herausforderung einer relativ direkten Übertragung zwischen der Lenkbewegung am Tablet und der Lenkbewegung am Fahrzeug via TCP/IP erfolgreich gelöst werden. Auf weitere Einzelheiten bezüglich des Anwendungsfalls der Wankstabilisierung wird in dem folgenden Kapitel eingegangen.

4.2.3.3 Wankstabilisierung und synchrones Wanken

Um die Funktionsweise der Wankstabilisierung zu erläutern, wurde eine Art Computerspiel in Form eines Rennspiels entwickelt. Dabei soll der Benutzer die Funktion des ARS Systems übernehmen und die Lage des Fahrzeugs in den Kurven ausgleichen. Zu Beginn des Anwendungsfalls kann der Benutzer entscheiden, ob der Benutzer vor dem Spiel eine Einweisung, bzw. Tutorium, wünscht oder sofort mit der Fahrt starten will.

Aufgrund der Funktionsweise des ARS Systems Wankbewegungen auszugleichen spielt das Trägheitsmoment in diesem Anwendungsfall eine entscheidende Rolle. Zur Darstellung des Trägheitsmoments des Fahrzeugs in der Kurve, und dafür ein Maß für die Lage des KFZs, wurden zwei verschiedene Elemente eingesetzt. Hierbei handelt es sich zum einen um eine virtuelle, zum anderen um eine reale Anzeige. Während die virtuelle Anzeige auf dem Display des Tablets abgebildet wird, kann das Trägheitsmoment zusätzlich direkt als reale Anzeige am Fahrzeug, durch dessen Aktuatorik der Wankstabilisierung, dargestellt werden.

Um die virtuellen Bestandteile der Anwendung zu beschreiben, wird in Abbildung 22 die GUI während des Spielverlaufs dargestellt. Für eine detaillierte Erläuterung der Abläufe während der Benutzung wird auf das Screenflow-Diagramm im Anhang verwiesen.

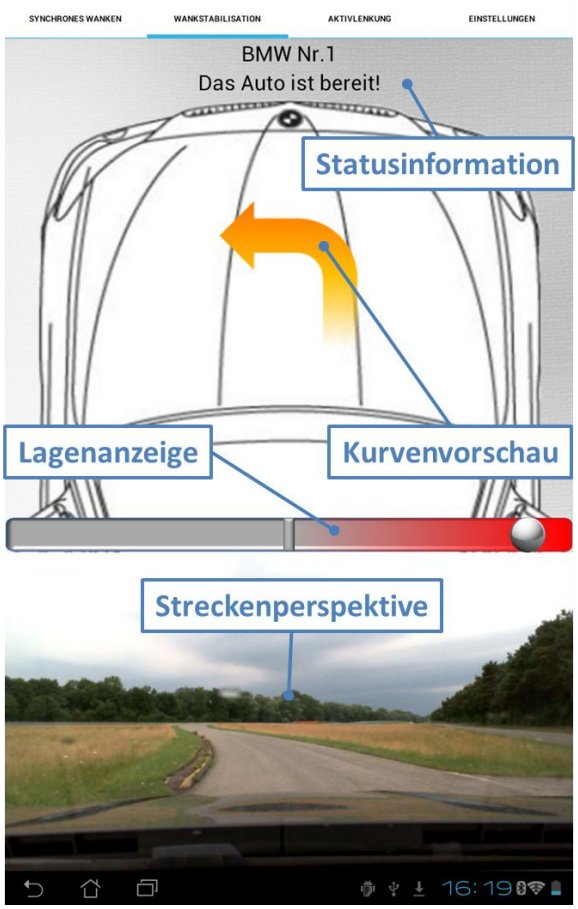


Abbildung 22 GUI-Elemente der Wankstabilisierung

Als virtuelles Element wird die Lage anhand einer beweglichen Kugel dargestellt, die in einer horizontalen Rinne oberhalb der Abbildung zur Perspektive auf die Rennstrecke läuft. Bewegt sich die Kugel aufgrund einer starken Kurve in einen der beiden Randbereiche der Rinne, wird dies dem Benutzer mittels eines Verlaufs in roter Farbe in diesem Bereich dargestellt. Damit wird der Benutzer darauf hingewiesen, dass der Benutzer durch weitere Interaktion diesen negativen Effekt beheben muss. Zusätzlich zur optischen Unterstützung des Benutzers, wird beim Eintritt der Kugel in den negativen Bereich dieser akustisch auf seine schlechte Regelleistung durch einen Brummton hingewiesen. Um die Bewegungen der Kugel beeinflussen zu können, muss der Benutzer das Tablets um dessen Rollachse rotieren. Dabei werden lediglich 60 Grad Drehung des Tablets benötigt, um zwischen den möglichen Extremwerten zu wechseln. Somit ergibt sich die Lage der Kugel als Resultat aus der Summe der Regelung des Benutzers und der Manipulation durch die Kurvenlage des virtuellen Fahrzeugs im Spiel während der Streckenfahrt. Die Abweichung zwischen der Ideallinie beim Ausgleichen des Wankens und der tatsächlich durch den Benutzer erzeugten Linie wird in einen Punktewert umgerechnet und am Ziel dem Benutzer in Form einer Highscore-Liste angezeigt. Die Liste der Highscores wird als Textdatei auf dem Tablet hinterlegt.

Zur Demonstration des Resultats beim Ausgleichsversuch der Wankbewegung durch den Benutzer wird die Aktuatorik des Fahrzeugs entsprechend angesteuert. Der resultierende Wankwinkel wird demnach gemäß der minimalen Zykluszeit, die bereits im vorherigen Kapitel erläutert wurde, an den Server versandt. Der Effekt dieses Vorgehens ist, dass sich das Fahrzeug bei einer schlechten Regelleistung des Benutzers aus der Ruhelage heraus bewegt, bzw. eine Wankbewegung der

Karosserie erkennbar ist. Um dem Benutzer ein Gefühl des Fahrens zu vermitteln, werden zudem anhand des STEERING-Kanals Lenkwinkel entsprechend der Kurvenfahrt im Spiel erzeugt.

Für diesen Anwendungsfall wurde extra das wiederverwendbare Steuerelement *RollAngleBarView* als eigene Klasse des Pakets *Ui* implementiert. Hierbei handelt es sich um eine Erweiterung der View Klasse der Android API, die unabhängig von den eingehenden inertialen Sensordaten vom UI-Thread neu gerendert wird. Die Klasse *RollAngleBarView* kapselt die Funktionalität der Rollwinkelerfassung als auch der Darstellung dieser grafischen Komponente. Um auf Ereignisse wie das Erreichen oder Verlassen der Randbereiche der Leiste reagieren zu können, handelt es sich bei der Klasse zudem um eine Implementierung des *Observer* Entwurfsmusters. Durch die Methode *registerBarHolder()* kann sich ein Objekt als *Observer* für die genannten Ereignisse registrieren. Beim Eintritt eines Ereignisses wird der *Observer* per Callback-Methode zurückgerufen. Aus diesem Grund muss ein *Observer* dieser Klasse auch das Interface *CallbackBarHolder* implementieren. Die Erweiterung der Klasse zur Unterstützung mehrerer observierender Objekte ist durch die Implementierung einer einfachen Schleife über die registrierten *Observer* nachträglich auf einfache Weise realisierbar. [GAMMADP]

Für die Streckenperspektive und die Trägheitsmomente, bzw. Querbeschleunigungen, die im Spiel verwendet werden, wurden die Bussignale und ein Video während einer Testfahrt auf der BMW Teststrecke in Aschheim aufgezeichnet. Die Busdaten wurden anschließend über ein BMW Tool für die weitere Verarbeitung in Textform exportiert. Die exportierten Daten als auch das Video müssen auf dem Tablet hinterlegt werden. Die App greift auf diese Daten während des Spiels zurück.

Um die exportierten Busdaten verwenden zu können, wurde die Klasse *FileReader* des Pakets *FileHandling* implementiert, die es erlaubt eine CSV-Datei einzulesen, die einzelnen Werte zu skalieren und in geeigneter Form in einer Datenstruktur für den späteren Gebrauch abzulegen. In der *onCreate()*-Methode der Klasse *WankstabilisierungSchueler* wird dieser Vorgang in einem eigenen Thread bearbeitet. Vorgänge, bei welchen mit einer erhöhten Laufzeit zu rechnen ist, wurden, wie hier das Einlesen von Daten aus einer Textdatei, in einen eigenen Thread ausgelagert. Somit wird durch langwierige Bearbeitungen nicht die Reaktivität der Benutzerschnittstelle verlangsamt, bzw. gestört. Zur Ermittlung der Kurvenvorhersage und der Wertung einer Kurvenfahrt wird die Methode *updateSensorValue()* der gleichen Klasse von der Klasse *RollAngleBarView* bei neuen Sensorwerten zurückgerufen. Dieses Vorgehen wurde gewählt, da die Sensorwerte von der Android API in unregelmäßigen Abständen ermittelt werden.

Mit Hilfe dieser Spieleanwendung können somit Benutzer direkt den Effekt des ARS am realen Fahrzeug betrachten, bzw. durch Sitzen im Innenraum erfahren. Um das Interesse der Besucher am BMW-Familientag zu gewinnen, wurde darüber hinaus eine zweite Variante zur Erklärung des ARS Systems mit größerer Publikumswirkung erstellt. Der Anwendungsfall auf dem Reiter „Synchrones Wanken“ wurde zu diesem Zweck entwickelt. Nach der Aktivierung dieses Anwendungsfalles über die Einstellungen führen mehrere Fahrzeuge synchron entsprechende Wankbewegungen durch. Prinzipiell werden dabei die gleichen Mechanismen wie im Reiter „Wankstabilisation“ eingesetzt. Die synchronen Wankbewegungen der gekoppelten Fahrzeuge werden allerdings nicht im Zuge eines Rennspiels, sondern anhand von zwei verschiedenen Modi durchgeführt.

Befindet sich der Anwendungsfall im ersten Modus, oder auch Manuell-Betrieb, kann der Benutzer, wie bereits aus dem gerade beschriebenen Anwendungsfall bekannt, anhand der Rollbewegung des Tablets die Fahrzeuge manipulieren. Dabei handelt es sich um eine unverfälschte Manipulation, die sich exklusiv an der Lage des Tablets orientiert. Manipulationen durch eine virtuelle Kurvenfahrt

kommen bei diesem Anwendungsfall nicht zum Tragen. Auf diese Weise kann der Benutzer unmittelbar extreme Effekte bezüglich der Lage des Fahrzeugs herbeiführen. Hierbei wird ersichtlich welche Kräfte von der Aktuatorik ausgehen, wenn man versucht das Auto von Hand in diese Lage zu versetzten, ohne dabei das ARS System als Hilfsmittel zu nutzen.

In dem zweiten Modus, dem Automatik-Betrieb, bewegen sich die Fahrzeuge automatisch und synchron zu einer vorher festgelegten Choreographie. Bei diesem Betriebsmodus reagiert die Aktuatorik komplett unabhängig von der Rollbewegung des Tablet. Die Choreographie wurde im Zusammenhang mit einem Musikstück vorher anhand einer eigenen App[32] aufgezeichnet und in einer eigenen Datei abgelegt. Beim Starten des Automatik-Betriebs, wird zeitgleich das Musikstück auf dem Tablet gestartet. Durch eine Kopplung des Tablet via Bluetooth mit dem Fahrzeug, konnte die Musik über die Boxen des Auto-Radios, als akustische Verstärker, wiedergegeben werden.

Völlig unabhängig vom Betriebsmodus lassen sich einzeln bis zu drei Fahrzeuge unabhängig zu- und abschalten. Grundsätzlich gilt hier, wie für alle Anwendungsfälle, dass Fehlermeldungen in einer entsprechenden Statuszeile dargestellt werden. Zu diesem Zweck wurde die Klasse *ScrollTextView* erstellt, die eine Laufschrift mit dynamischer Laufgeschwindigkeit darstellt. Die Laufgeschwindigkeit des dargestellten Textes ist dabei von dessen Länge abhängig. Längere Texte laufen dabei schneller durch das Bild als kürzere Texte, damit die Durchlaufzeit gleich bleibt.

Im folgenden Kapitel wird der BMW-Familientag bezüglich der Herausforderungen und der Akzeptanz beim Publikum zusammengefasst. Darüber hinaus wird auf die Hindernisse beim Einsatz der „Synchronen Wanken“-Anwendung am Familientag eingegangen.

[32] Die SuspensionRecorder-App schreibt zyklisch inertiale Sensorwerte in eine Textdatei während ein Musikstück abgespielt wird.

4.3 Fazit BMW-Familientag

Durch den interaktiven Charakter der Präsentation konnten Besucher aller Altersklassen für den Demonstrator begeistert werden. Dabei konnten sich besonders die jüngeren Benutzer den Umgang weitestgehend selbst erschießend. Darüber hinaus wurde allerdings eine kurze Einweisung der Benutzer vom Standpersonal durchgeführt. Vereinzelt wurde nach der genauen Funktionsweise des ARS Systems gefragt und ob dieses System bereits in Serienmodellen verbaut wird. Damit konnte in einem ersten Schritt bereits ermittelt werden, dass ein Demonstrator dieser Form definitiv Interesse bezüglich der technischen Hintergründe schafft. Besonders das Sitzen in dem Fahrzeug während der Anwendungsfälle zur Wankstabilisierung lässt den Benutzer die Funktionalität des ARS Systems erfassen. Der Anwendungsfall „Synchrones Wanken“ konnte auf dem BMW-Familientag lediglich jeweils mit einem Fahrzeug durchgeführt werden. Somit wurde jedes Tablet exklusiv mit dem dafür vorgesehenen Fahrzeug verbunden. Die „Synchrone Wanken“-Anwendung wurde deshalb lediglich zwischenzeitlich eingesetzt, solange keine Benutzer die Tablet-Anwendung direkt verwenden wollten. Auf diese Weise konnten schnell neue Benutzer gefunden werden und die Zeiten, in welchen das Tablet nicht unmittelbar verwendet wurde, konnten minimiert werden.

Gerade aufgrund des hohen Energieverbrauchs durch die WLAN-Verbindung und den anfallenden Datenverkehr mussten die Tablets zwischenzeitlich geladen werden. Somit wäre eine effizientere Datenübertragung bezüglich des Energieverbrauchs, wie unter Umständen durch die Bluetooth-Technologie, für längere Präsentation geeigneter.

Durch den Demonstrator konnte Benutzern die Funktion des ARS Systems verdeutlicht werden, indem keine Störgrößen für den Benutzer spürbar aufgetreten sind. Als Störgröße wird in diesem Zusammenhang die Trägheit während der Kurvenfahrt bezeichnet, welche der Fahrzeuginsasse erfährt. Durch diesen Einflussfaktor während einer Fahrt auf der Teststrecke fällt dem naiven Probanden unter Umständen nicht das Eingreifen eines regulierenden Systems auf. Die Effekte der Aktuatorik sind an einem stehenden Fahrzeug anhand des Demonstrators deutlich spürbar, wenn auch hier ähnlich wie bei einer Testfahrt die Benutzer in das Fahrzeug einsteigen. Damit verfügt der Proband nicht nur über einen virtuellen optischen Eindruck für den Kontext der Fahrt, wie Geschwindigkeit und Kurvenverlauf, sondern auch über eine haptische Rückkopplung.

Um den Demonstrator verwenden zu können, musste ein WLAN am BMW-Familientag für den Präsentationsstand eingerichtet werden. Durch den Einsatz von weiteren Funk-Netzwerken in der unmittelbaren Umgebung, wie beispielsweise auch an benachbarten Ständen, wurde vermutlich die Datenübertragung gerade im Zusammenhang mit der „Synchrones Wanken“-Anwendung stark beeinflusst. Während des Testlaufs zu Beginn des Familientages mit mehreren Fahrzeugen waren die Wankbewegungen der verbundenen drei Fahrzeuge asynchron. Aufgrund der Verfügbarkeit von Testfahrzeugen konnte vor dem BMW-Familientag kein Testlauf mit mehreren Fahrzeugen unter realen Bedingungen durchgeführt werden. Die Ermittlung der Fehlerquelle bei der „Synchronen Wanken“-Anwendung mit mehreren Fahrzeugen wurde im Zuge dieser Masterarbeit nicht mehr durchgeführt und bleibt für Folgeprojekte offen.

In der Gesamtheit wurde der Einsatz des Demonstrators am BMW-Familientag als Erfolg verbucht. Insbesondere konnten in Einzelvorführungen nach dem BMW-Familientag einige Führungskräfte für das Projekt begeistert werden.

Kapitel 4.4 geht auf den Einsatz von AR im Zusammenhang mit der Weiterentwicklung des Demonstrators ein.

4.4 Verwendung von AR bei der Weiterentwicklung des Demonstrator-Paketes

Bisher wurde der Demonstrator, dessen Funktionen insbesondere der Kommunikation zwischen Client und Server, beschrieben. In diesem Kapitel soll geklärt werden, in welchen Maßen sich Augmented Reality zur Erklärung von Steuergerätefunktionen in einer Weiterentwicklung des Demonstrators eignet, bzw. welche Möglichkeiten einer Anwendung sich anbieten.

Ein Vorteil beim Einsatz der verwendeten ASUS Tablets ist, dass bereits Kameras im Gerät integriert sind und deshalb keine weiteren Kosten für den Kauf einer Kamera anfallen. Durch eine Kamera auf der Rückseite der Tablet-Geräte ist es möglich optische Bilddaten zu erfassen. Somit ist AR unter Verwendung der Kamera und des Displays innerhalb einer Tablet-Anwendung einsetzbar. Zusätzliche Informationen werden dann virtuell in die aufgenommenen Bilddaten eingeblendet und zur Laufzeit auf dem Display dargestellt. Hierbei gilt die Darstellung des Kamerabildes zusammen mit den zusätzlichen Informationen in Echtzeit als besondere Herausforderung.

Darüber hinaus muss ein geeignetes Tracking-Verfahren eingesetzt werden, um optisch einen Zusammenhang zwischen den virtuellen Komponenten und dem realen Anteil der Anwendung zu erzeugen. Erst durch das Tracking-Verfahren wird es möglich virtuelle Bildbestandteile perspektivisch korrekt im dreidimensionalen Raum darzustellen. Aufgrund der Sensorik, ohne die Verwendung von kostenintensiven zusätzlichen Hilfsmitteln, bietet sich der Einsatz von optischen Verfahren für die Erklärung von Steuergerätefunktionen an. Des Weiteren ist auch ein Hybrid-Verfahren aufgrund der intertialen Sensorik beim Tracking denkbar. Dabei könnten die inertialen Sensoren eingesetzt werden, um die optische Posen-Bestimmung zu stabilisieren, bzw. robuster zu gestalten. [FRIEMT]

Aufgrund dieser Tatsachen und aufgrund der zur Verfügung gestellten Sensorik kann das Tablet als eine Art video-see-through HMD angesehen werden. Der Unterschied dabei zum klassischen video-see-through HMD, das in Kapitel 1.1.2 beschrieben wird, ist, dass das Tablet nicht am Kopf des Benutzers angebracht wird, sondern dieser das Gerät vor sich hält. Die Veränderung der Perspektive ist deshalb nicht von der Bewegung des Kopfes abhängig, sondern von der Bewegung der Kamera, bzw. des Tablets und dessen Pose. Durch dieses Vorgehen befindet sich der Benutzer weiterhin in einer dreidimensionalen Umgebung, die für diesen ein natürliches optisches Verhalten bereitstellt. Die Darstellung von zusätzlichen Informationen ist abhängig von der aktuellen Pose der Kamera und der daraus resultierenden Perspektive. Da die Pose der Kamera indirekt von der Pose des Kopfes und der Arme des Benutzers abhängig gemacht wird, entsteht der Eindruck einer erweiterten Realität.

Durch diese Form der Darstellung, bzw. Wahrnehmung, ist die Barriere zur Benutzung des Präsentationsmaterials sehr gering. Eine intuitive Benutzung erlaubt es Personen jeden Alters die Funktion der Steuergeräte anhand zusätzlich eingeblendeter Informationen zu erfahren.

Um die Funktionsweise der Steuergeräte direkt am realen Fahrzeug betrachten zu können, ist es notwendig, dass die Aktuatorik des Fahrzeugs tatsächlich angesteuert und manipuliert werden kann. Die Voraussetzungen für die Verwendung der Aktuatorik in Verbindung mit einer AR Anwendung wurden durch die Entwicklung des Demonstrators geschaffen. Zur Wiederverwendung dieser Software-Komponenten des Demonstrators auf Clientseite wurden die entsprechenden Klassen in einem JAVA-Archiv zusammengefasst. In Kapitel 5.2.2 wird der Einsatz dieses Archivs innerhalb des entwickelten Prototyps beschrieben.

Im nächsten Kapitel wird der Einsatz von Augmented Reality bezüglich des Aufbaus des entwickelten Prototyps, der verwendeten Tools und Ideen für Anwendungen in zukünftige Projekte behandelt.

5 Einsatz von Augmented Reality in einem Prototyp

Um die Funktionsweise der Niveauregulierung, die bereits in Kapitel 2.2.1.1 angesprochen wurde, im Zuge einer kurzen Präsentation erklären zu können, wurde ein Prototyp in Form einer weiteren Android-App entwickelt.

Die Niveauregulierung erlaubt es den Höhenstand des Hecks eines Fahrzeugs mittels spezieller Aktuatoren zu regulieren. Somit kann das Automobil unabhängig von der Masse der Beladung am Heck auf einem bestimmten Niveau gehalten werden. Um die Funktionsweise erklären zu können, ist es auf der Seite der Aktuatorik notwendig, dass mindestens zwei Zustände unterschieden werden. Im ersten Zustand befindet sich das Fahrzeug, welches mit einer Masse im Kofferraum beladen wurde, bei deaktivierter Niveauregulierung. Damit befindet sich das Heck des Fahrzeugs bei genügend starker Belastung in einem abgesenkten Bereich. Den zweiten Zustand erreicht das Fahrzeug bei Aktivierung der Niveauregulierung. Weitere Zustände könnten hinzugefügt werden, um später mehrere Beladungsstufen im Anwendungsfall unterstützen zu können. Da durch die aktuelle Version des Servers das Ansteuern der Niveauregulierung noch nicht unterstützt wird, wurden Tests zur Entwicklung des Prototyps an einem simulierten Server unter Verwendung eines anderen bereits unterstützten Kanals durchgeführt.

Als Anwendungsfall wurde ein Beladungsszenario gewählt, das interaktiv durch den Benutzer beeinflusst werden kann. Der Benutzer fungiert dabei lediglich als steuernder Beobachter, da dieser keine realen Gewichte in das Fahrzeug einlädt. Der Beladungsvorgang wird von einer virtuellen Person erledigt. Der Benutzer erteilt dieser Person per Knopfdruck die Anweisung virtuelle Gewichte in den Kofferraum zu legen. Dadurch wird das reale Fahrzeug abgesenkt. Anschließend kann durch einen weiteren Knopfdruck das System zur Niveauregulierung aktiviert werden und das reale Fahrzeug wird wieder auf den ursprünglichen Höhenstand zurückversetzt. Die virtuellen Bestandteile der Anwendung werden dabei zusätzlich zu der realen Umgebung, die über die Kamera aufgenommen wird, mittels AR in dem Display des Tablets dargestellt. Auf diese Weise hat der Benutzer der Anwendung die Möglichkeit das Fahrzeug während der Beladung aus verschiedenen Perspektiven zu betrachten und kann sich damit auf Bereiche konzentrieren, die für den Benutzer besonders interessant sind. Würde der Benutzer das Fahrzeug tatsächlich mit realen Gewichten beladen, um den Effekt der Niveauregulierung zu betrachten, hat dieser nicht die Möglichkeit den gesamten Ablauf von außen zu verfolgen. Zudem hat der Benutzer die Möglichkeit den Ablauf der Präsentation selbst zu steuern, bzw. zu beeinflussen. Erst durch Interaktion mit dem Benutzer werden Effekte der Aktuatorik angestoßen. Damit sind die Unterschiede bei aktiviertem, bzw. deaktiviertem System für den Benutzer unmittelbar ersichtlich. Aufgrund der Tatsache, dass ein reales Automobil den Effekt vorführt, entsteht für den Benutzer zudem eine Erfahrung am Automobil, die über einen haptischen Aspekt verfügt. Dementgegen stehen beispielsweise Videofilme zur Erklärung dieser Vorgänge. Videofilme sind rein visueller Natur und bergen demnach eine gewisse Distanz zur Realität zwischen dem Benutzer und dem Fahrzeug. Damit bietet sich aufgrund der Vermischung von Realität und Virtualität der Einsatz von AR in diesem Zusammenhang an. Zusätzlich können bei dieser Form der Darstellung auch verdeckte Elemente der Fahrzeugmechanik am realen Fahrzeug dargestellt werden, die sonst durch die Karosserie verdeckt sind. Auf diese Einsatzmöglichkeit wird in Kapitel 7 genauer eingegangen.

In Abbildung 23 wird der AR-Prototyp auf einem Samsung Galaxy SIII GT-I9300 dargestellt. Durch eine rote Ellipse wird der Bereich gekennzeichnet, in welchem die virtuellen Elemente dargestellt werden.

Dabei handelt es sich um eine weibliche Person im Zentrum und einen Tisch mit einem Goldbarren am Heck des Fahrzeugs.

Abbildung 23 AR-Prototyp mit Markerless 3D Tracking, augmentierter Bereich (rot)

Aufgrund der umfangreichen Arbeiten zur Erstellung von grafischen dreidimensionalen Modellen wurde in der Arbeit auf die Darstellung von mechanischen Elementen in Form einer Überlagerung des Videobildes verzichtet. Des Weiteren werden in der Arbeit Eigenentwicklungen auf Ebene der Algorithmik zum Thema optisches Tracking, bzw. Darstellung von dreidimensionalen Modellen keine eingesetzt. Zur Implementierung einer entsprechenden App, welche über die genannten Eigenschaften verfügt, werden verschiedene Werkzeuge und APIs beschrieben, die in diesem Zusammenhang verwendet wurden.

Eine besondere Herausforderung bei der Verwendung von Augmented Reality ist die Ermittlung von Bildmaterial anhand der Kamera und die unmittelbare Darstellung der Kameradaten erweitert um die zusätzlichen Informationen. Dabei müssen diese Daten in Echtzeit aus einem dreidimensionalen Modell ermittelt und verarbeitet werden. Insbesondere mobile Endgeräte, welche eine geringere Rechenleistung im Vergleich zu modernen stationären Rechnern bereitstellen, sind dabei nur begrenzt einsetzbar. Der Ansatz, dass die Verarbeitung auf einen Server übertragen wird, der die Berechnungen vornimmt und entsprechende Ergebnisse an das Tablet zurücksendet, wie in [FRIEMT] beschrieben, wurde hier nicht verfolgt. Die Ergebnisse des Prototyps zeigen, dass die verwendeten Tablets über ausreichend Rechenleistung verfügen, um die Berechnungen des Anwendungsfalls durchzuführen. Auf diese Weise wird die Kommunikation zwischen Client und Server nicht mit einem noch höheren Datenaufkommen belastet. Auch die Verwendung durch mehrere Benutzer, welche in einer Art Beobachtungsmodus mit dem Server verbunden sind, und demnach selbst nicht die Aktuatorik fernsteuern, könnten so die Szene der AR-Anwendung unabhängig voneinander betrachten. Würden die Berechnungen der Anwendung, wie in [FRIEMT] beschrieben, ausgelagert, steigt damit der Rechenaufwand am Server zusammen mit der Anzahl der verbundenen Benutzer.

Im folgenden Kapitel wird auf die einzelnen Werkzeuge eingegangen, die im Zuge dieser Arbeit zur Erstellung eines Prototyps unter Verwendung von AR eingesetzt wurden. Darüber hinaus wird in

Kapitel 5.2 auf den Aufbau des Prototyps auf Implementierungsebene eingegangen und der Arbeitsablauf in Zusammenhang mit den einzelnen Werkzeugen erläutert.

5.1 Augmented Reality Frameworks und Tools

Während der Entwicklung des Prototyps wurden verschiedene Frameworks und Tools eingesetzt, um die komplexe Aufgabenstellung einer AR Anwendung auf einem mobilen Endgerät zu ermöglichen.

Für den Einsatz von Augmented Reality auf mobilen Endgeräten gibt es bereits einige Rahmenwerke, die in diesem Zusammenhang verwendet werden können. Aufgrund des Umfangs der Masterarbeit wurde hier allerdings nach einer Lösung gesucht, die nicht noch erhebliche Implementierungsaufwand erfordert, auf einem mobilen Gerät einsetzbar und performant ist. Aufgrund der Anforderungen von Seiten der BMW sollte das verwendete Verfahren weitestgehend ohne weitere Hilfsmittel einsetzbar sein. Besonders der Einsatz von Markern in einem Marker-Tracking System wurde dabei vorerst von der Betrachtung ausgeschlossen. Verfahren, wie das Feature Tracking, bzw. Markerless Tracking, sind für den Einsatz im genannten Kontext zur Präsentation aufgrund einer homogenen Umgebung ohne Adaption von Markern dem Marker-Tracking vorzuziehen. Entwicklungen, wie das ARToolKitPlus[33], welches eine Erweiterung der ARToolKit-Bibliothek für mobile Endgeräte darstellt, konnten demnach nicht für den Prototyp eingesetzt werden. Auch das von der Firma Qualcomm zur Verfügung gestellte Vuforia Augmented Reality SDK[34] bietet direkt keine Funktionen für den Bereich des Markerless Tracking an. Die FastCV-Bibliothek[35] von Qualcomm stellt Funktionen für eine eigene Implementierung auf einem niedrigen technischen Level zur Verfügung. Darüber hinaus wurde die FastCV-Bibliothek in erster Linie für mobile Geräte, insbesondere für Android-Geräte, entwickelt. Wegen des Umfangs einer eigenen Implementierung wurde allerdings von einem Einsatz der FastCV-Bibliothek abgesehen. Für den Einsatz eines Markerless Tracking Verfahrens wurde das Metaio Mobile SDK eingesetzt, dass unter anderem eine Art modellbasierten Ansatz des Trackings unterstützt. Ein weiteres Tool, welches Markerless Tracking einsetzt, ist die kommerzielle AR-Lösung D'Fusion Studio Suite[36] der Firma Total Immersion, die allerdings im Zuge dieser Arbeit nicht genauer betrachtet wurde.

Im nächsten Kapitel wird das Metaio Mobile SDK beschrieben und welche Möglichkeiten dieses Tool für eine AR-Anwendung bereitstellt. Auf andere bereits erwähnte Frameworks wird aufgrund des Umfangs der Arbeit hier nicht genauer eingegangen.

5.1.1 Metaio Mobile SDK

Das Metaio Mobile SDK wurde von der metaio GmbH als Lösung für AR-Anwendungen auf mobilen Endgeräten entwickelt. Insbesondere wird die Entwicklung von AR-Software für iOS- und Android-Geräte unterstützt. Aus diesem Grund lassen sich Inhalte der Masterarbeit auch teilweise für Weiterentwicklungen auf iOS-Geräten wiederverwenden. Ein weiterer Vorteil ist, dass das Metaio Mobile SDK über ein Plugin für die Unity Spiele-Entwicklungsumgebung verfügt, die in Kapitel 5.1.3 beschrieben wird. Zudem können Inhalte legal unter Verwendung des Metaio Mobile SDK veröffentlicht werden, ohne das Produkt käuflich erworben zu haben. Lediglich ein Wasserzeichen während des Anwendungsbetriebs und die Anzeige des Metaio Firmenlogos zu Beginn der Anwendung werden bei der kostenlosen Version als Branding Features vorausgesetzt. Das

[33] Siehe http://studierstube.icg.tugraz.at/handheld_ar/artoolkitplus.php
[34] Siehe https://developer.qualcomm.com/mobile-development/mobile-technologies/augmented-reality
[35] Siehe https://developer.qualcomm.com/docs/fastcv/api/index.html
[36] Siehe http://www.t-immersion.com/products/dfusion-suite

Wasserzeichen wird am Beispiel des AR-Prototyps in Abbildung 23 in der unteren linken Ecke des Displays dargestellt. Die Begriffe Unity und Unity3D werden im weiteren Verlauf der Arbeit synonym verwendet und bezeichnen die Spiele-Entwicklungsumgebung der Firma Unity Technologies ApS. [METAIO]

Das Metaio Mobile SDK in Bezug auf die Android-Entwicklung beinhaltet die Hyperlinks zur Dokumentation der API im Internet, Ausdrucke für verschiedene Tracking-Verfahren, ein Beispiel-Projekt und das JAR-Archiv der Bibliothek zur Nutzung in einem Android-Projekt als auch eine *unitypackage*-Datei, die es erlaubt, das Metaio Mobile SDK in Unity einzusetzen.

In der Dokumentation wird unter anderem beschrieben, wie man das Metaio Mobile SDK in ein Android-Projekt der Eclipse Entwicklungsumgebung oder in ein Unity Projekt einbinden kann. Gerade in Bezug auf die Integration des Metaio Mobile SDK in ein Unity Projekt konnte die Dokumentation für die Entwicklung des AR-Prototyps eingesetzt werden. [METUNI]

Kernstück des Metaio Mobile SDKs ist das metaio Unifeye SDK mobile, welches in einer modularen Weise implementiert wurde. Durch das Unifeye SDK mobile wird AR Funktionalität für mobile Geräte durch drei verschiedene Komponenten zur Verfügung stellt. Bei den einzelnen Komponenten handelt es sich um eine Tracking, eine Rendering und eine Capture Komponente. Während das Tracking und die Aufnahme der Bilddaten, bzw. das Capturing, im AR-Prototyp grundsätzlich durch das Metaio Mobile SDK wahrgenommen werden, wird das Rendering, bzw. die Darstellung der Inhalte auf dem Display, durch Unity vorgenommen. [UNIFEY]

Durch das Metaio Mobile SDK werden prinzipiell zwei Kategorien verschiedener Tracking Technologien zu Verfügung gestellt. Dabei handelt es sich einerseits um die optischen Tracking-Verfahren und andererseits um Verfahren, die keine optischen Daten nutzen. In der zweiten Kategorie wird das Tracking über inertiale Sensorik und GPS zusammengefasst. Im weiteren Verlauf wird lediglich auf zwei Formen der unterstützten optischen Tracking-Verfahren eingegangen. Welches Verfahren in der Anwendung verwendet wird, kann unter anderem über eine XML-Datei festgelegt werden. Unter anderem werden in diesen Dateien auch Schwellwerte festgelegt, die entscheidend für die Detektion der Marker, bzw. Ziele, sind. Zu niedrige Schwellwerte führen dabei zu dem Problem, dass dem Marker ähnliche Bildbereiche fälschlicherweise als der Marker selbst erkannt werden.

In Kapitel 5.2.4 wird auf den Zusammenhang zwischen der XML-Konfiguration und den dargestellten Objekten bezüglich der Anwendung im AR-Prototyp eingegangen, während die Kapitel 5.1.1.1 und 5.1.1.2 die Konfigurationsmöglichkeiten für das jeweilige Tracking-Verfahren beschreiben. [TRACON]

5.1.1.1 ID-Marker Tracking und Picture Tracking

Bereits in Kapitel 1.1.2.3 wurde darauf hingewiesen, welcher besonderen Form ein ID Marker entspricht, bzw. welche spezifischen Merkmale dieser aufweist. Durch das Metaio Mobile SDK ist es möglich bis zu 512 verschiedene ID-Marker in einer AR-Anwendung einzusetzen. Dabei ist es allerdings im Falle der Implementierung durch Metaio hinsichtlich aller Kriterien irrelevant, ob es sich tatsächlich um einen ID-Marker oder um ein Bild in dem weiß/schwarz/weiß Rahmen handelt. In einer mobilen AR-Anwendung auf einem modernen Smartphone können laut [TRACON] zwischen 10 bis 30 dieser Marker eingesetzt werden.

Im Unterschied zum Einsatz von ID-Markern als Hilfsmittel können für das optische Tracking-Verfahren auch eigene Marker erstellt und verwendet werden. Dabei muss es sich nicht zwangsläufig um eine schwarz/weiß-Kodierung im Inneren des Markers handeln. Beliebige Bilder können im Inneren des weiß/schwarz/weiß Rahmens des Markers abgebildet werden. Diese Form der Marker wird von Metaio als Picture Marker bezeichnet.

Dieses Tracking-Verfahren unter Verwendung von ID-Markern wurde aufgrund der hohen Robustheit und des geringe Aufwands im Umgang mit dem ID-Marker für Testzwecke bei der Entwicklung des Prototyps häufig eingesetzt. In Tabelle 2 wird eine Übersicht zu den unterschiedlichen Tracking-Verfahren hinsichtlich einiger Kriterien dargestellt. In den Spalten werden die unterschiedlichen Tracking-Verfahren und in den Zeilen die einzelnen Kriterien angetragen. Dabei wird mit der Anzahl der Plus-Zeichen und Minus-Zeichen die Güte des Verfahrens bezüglich eines Kriteriums bezeichnet. Das Plus-Zeichen steht dabei für eine bessere Leistung gegenüber einer Variante, welche mit einem Minus-Zeichen gekennzeichnet wurde. Mit „Anzahl der Ziele“ wird bezeichnet, wie viele verschiedene Marker, bzw. Ziele maximal in einer Anwendung eingesetzt werden können. Die Tabelle ist gültig für die Generation von mobilen Endgeräten, welche über einen 1GHz Prozessor verfügen.

	ID Marker	Picture Marker	Markerless (FAST)	Markerless (ROBUST)	Inertial Sensors
Robustheit	+	+	-	-	++
Genauigkeit	+	+	+	+	-
Geschwindigkeit	+	+	+	-	++
Anzahl der Ziele	Bis zu 512	Bis zu 30	Bis zu 10	Bis zu 3	1

Tabelle 2 Vergleichstabelle, Metaio Tracking-Verfahren [TRACON]

Zur Konfiguration des Tracking-Verfahrens wurde eine XML-Konfigurationsdatei eingesetzt. In dieser Datei können verschiedene Parameter hinterlegt werden, welche das Tracking-Verfahren genauer spezifizieren. Welche Möglichkeiten der Parametrierung vorhanden sind, wurde von Metaio in der Dokumentation allerdings nicht beschrieben. Beispiele zu den unterschiedlichen Verfahren wurden in [TRACON] hinterlegt. Durch die XML-Konfigurationsdatei wird beispielsweise die Art des Tracking-Verfahrens, der Modus bezüglich der Tracking Qualität, ein Schwellwert zur Marker-Erkennung, die Anzahl der Suchläufe im Bild und für den Marker spezifische Informationen festgelegt. Zu den Informationen bezüglich der Marker wird beispielsweise dessen Größe in Millimeter angegeben.

Die XML-Konfigurationsdatei des ID-Marker Tracking-Verfahrens, welches im AR-Prototyp zu Testzwecken eingesetzt wurde, befindet sich auf der CD-ROM im Anhang unter „Metaio_Tracking_Config_Files/trackingID80.xml“. Durch die Konfigurationsdateien kann somit auch das eingesetzte Tracking-Verfahren am AR-Prototyp ohne erheblichen Aufwand verändert werden. Lediglich die Anpassung der Parameter für eine verbesserte Nutzung des Tracking-Verfahrens im gegebenen Kontext musste empirisch vorgenommen werden.

Im folgenden Kapitel wird auf die verschiedenen Varianten des Markerless Tracking anhand des Metaio Mobile SDK eingegangen. Dabei wird auf die unterschiedlichen Betriebsmodi des Markerless Tracking eingegangen und die Merkmale und Konfigurationen der jeweiligen Methode beschrieben.

5.1.1.2 Markerless und Markerless 3D Tracking

Das Markerless Tracking, wie in Kapitel 1.1.2.4 beschrieben, weicht von der im Metaio Mobile SDK verwendeten Definition ab. In der Definition von Metaio wird das Tracking von Bildern ohne einen

weiß/schwarz/weiß Rahmen bereits als ein Markerless Tracking-Verfahren bezeichnet. Somit wird durch die Definition von Metaio ein Marker durch einen Rahmen gekennzeichnet, der diesen von der Umgebung optisch abgrenzt.

Das von Metaio zur Verfügung gestellte Markerless Tracking Verfahren kann in zwei weitere Kategorien unterteilt werden. Das erste dieser Tracking-Verfahren ermöglicht es Bilder, die entsprechende Texturierungseigenschaften aufweisen, zu detektieren. Dieses Tracking-Verfahren wird von Metaio als Markerless Tracking bezeichnet, welches genau wie die vorher beschriebenen Verfahren über zwei unterschiedliche Betriebsmodi verfügt. Der FAST-Betriebsmodus sollte bei gewöhnlich texturierten Bildern eingesetzt werden, während der ROBUST-Betriebsmodus besser bei hochgradig texturierten Bildern verwendet werden sollte. Eine besondere Eigenschaft des ROBUST-Betriebsmodus ist, dass sich das Tracking-Ergebnis dieser Methode während der Laufzeit verbessert. Bei der Wahl des Bildes sollten einige Kriterien beachtet werden, die in [MLESCO] beschrieben werden. Unter anderem wird der optimale Betrachtungswinkel und indirekt der optimale Betrachtungsabstand bezüglich der Größe des Bildes beschrieben. Entscheidend für eine stabile Tracking-Anwendung ist außerdem, dass die Parameter des Referenzbildes korrekt in der XML-Konfigurationsdatei eingetragen werden. Dabei werden die Abmessungen in Millimeter und der Dateiname des Referenzbildes angegeben. Auch der Einsatz mehrerer Marker in einer einzigen Anwendung kann mithilfe der Konfigurationsdatei definiert werden. [TRACON]

Aufgrund der Benennung des Wertes im *type*-Attribut der Sensorbeschreibung in der XML-Konfigurationsdatei des Markerless Tracking Beispiels aus [TRACON] kann man davon ausgehen, dass Metaio für das Markerless Tracking eine Art Feature Tracking einsetzt, das in Kapitel 1.1.2.2 beschrieben wurde. Der Wert des Sensortyps wird dabei mit „FeatureBasedSensorSource“ bezeichnet. In Tabelle 2 wird deutlich, dass dieser Ansatz bezüglich der Robustheit im Gegensatz zu anderen Tracking-Verfahren, wie dem ID-Marker Tracking, einen Nachteil bietet. Zusätzlich können aufgrund der Komplexität der Bilder bezüglich der Texturierung Einbußen bei der Geschwindigkeit der Anwendung entstehen. Auch die Anzahl der in einer Anwendung verwendbaren Bilder, bzw. Ziele, ist im Gegensatz zu einer ID-Marker oder Picture Marker Anwendung stark verringert. Ein Grund dafür könnte sein, dass für eine Detektion die Feature in dem Referenzbild gefunden und mit dem Kamerabild verglichen werden müssen, während bei dem Marker Tracking lediglich der Rahmeninhalt in ein grobes Raster zerlegt und verglichen wird. Somit könnten dabei Diskrepanzen zwischen den beiden Verfahren hinsichtlich des Rechenaufwands der Methode als auch der zu speichernden Referenzmodelle auftreten. Da keine Informationen über die internen Abläufe des Metaio Mobile SDK bezüglich der Umsetzung einzelner Tracking-Verfahren bekannt sind, kann an dieser Stelle nicht weiter auf die Unterschiede dieser Verfahren eingegangen werden. Auch ohne eine ausführliche Beschreibung der XML-Elemente und Attribute der Konfigurationsdatei müssen die korrekten Parameter einer Tracking-Anwendung in Tests empirisch ermittelt werden. Laut Metaio soll die Dokumentation allerdings in Zukunft noch um diese Informationen erweitert werden.

Als zweite Form des Markerless Tracking wird von Metaio das Markerless 3D Tracking unterstützt, welches es erlaubt ein Objekt der realen Welt als Referenzmodell für das Tracking einzusetzen. An dieser Stelle wird auf das Kapitel 1.1.2.4 verwiesen, welches den modellbasierten Tracking-Ansatz erläutert. Für das Markerless 3D Tracking wird das Referenzmodell im vornherein mittels dem Metaio Creator Mobile erzeugt. In Kapitel 5.1.2 wird genauer auf den Umgang mit dem Metaio Creator Mobile und dessen Funktion eingegangen. Durch dieses Werkzeug können auf komfortable Weise AR-Anwendungen auf mobilen Endgeräten, wie Android- oder iOS-Smartphones, realisiert

werden. Dabei ist die Qualität des Referenzmodells für die Tracking-Anwendung kritisch bezüglich einer robusten, bzw. genauen, Posen-Bestimmung. Die XML-Konfigurationsdatei, die zur Beschreibung des Tracking-Verfahrens im Metaio Mobile SDK eingesetzt werden kann, wird direkt zusammen mit dem Referenzmodell von dem Metaio Creator Mobile geliefert.

Das Markerless 3D Tracking wird seit der Version 3.1 des Metaio Mobile SDK unterstützt und ist damit die neueste Tracking-Technologie des SDKs. Es bietet zudem die Möglichkeit eine AR-Anwendung zu erstellen, die ohne weitere optische Hilfsmittel die Posen-Bestimmung der Kamera durchführen kann. Durch dieses Tracking-Verfahren wird das optische Erlebnis des Benutzers nicht durch unnatürliche Eingriffe in die Umgebung, in Form von künstlichen Markern, gestört. [TRACON]

Betrachtet man eine XML-Konfigurationsdatei, die von dem Metaio Creator Mobile für das Markerless 3D Tracking erstellt wurde, so kann man anhand der Benennung der einzelnen XML-Elemente und Parameter darauf schließen, dass auch Informationen der inertialen Sensorik in die Erstellung des Referenzmodells mit einbezogen werden. Dabei werden zum Beispiel die XML-Elemente *GravityVector* und *GravityToleranceAngle* festgelegt. Das *GravityToleranceAngle*-Element könnte dazu verwendet werden, um Ausreißer der Eingabedaten herauszufiltern, damit weiterhin eine stabile Pose angezeigt werden kann. Anhand des *Map*-Elements der Konfigurationsdatei wird der Name der Datei die das Referenzmodell enthält festgelegt. Mittels des *subtype*-Attributs des *Sensor*-Elements wird durch den Wert „ML3D“ spezifiziert, dass es sich bei dem verwendeten Tracking-Verfahren um das Markerless 3D Tracking handelt. Des Weiteren kann in der Konfigurationsdatei durch entsprechende XML-Elemente, wie *minTrackingMatches*, vermutlich die Mindestanzahl der zu erkennenden Feature festgelegt werden, damit das aktuelle Bild für eine Posen-Bestimmung qualifiziert ist. Wegen der Beschaffenheit der XML-Konfigurationsdatei wird davon ausgegangen, dass Metaio für das Markerless 3D Tracking eine Art Hybrid-Verfahren einsetzt. Dabei werden vermutlich optische Daten zusammen mit den inertialen Daten zur Posen-Bestimmung verwendet. Aufgrund des Berechnungsaufwands zur Detektion von korrespondierenden Bildpunkten zwischen Referenzmodell und den Bilddaten wird davon ausgegangen, dass sich die Geschwindigkeit für dieses Verfahren mit steigender Feature Anzahl erhöht. Diese Annahme stützt sich auf die Angabe der „Anzahl der Ziele“ in Tabelle 2 und deren Zusammenhang mit den verschiedenen Betriebsmodi des Markerless Tracking-Verfahrens.

Die XML-Konfigurationsdatei die für den AR-Prototyp unter Verwendung des Markerless 3D Tracking-Verfahrens eingesetzt wurde, befindet sich auf der im Anhang enthaltenen CD-ROM unter dem Pfad „Metaio_Tracking_Config_Files\bmwmodel\Tracking3D_bmwmodel.xml“.

Im nächsten Kapitel wird auf das Tool eingegangen, dass es erlaubt Markerless Tracking mit Hilfe des Metaio Mobile SDK durchzuführen. Dabei wird auf die Funktionsweise, den Umgang mit der Ausgabe und auf die Anwendbarkeit für verschiedene mobile Geräte eingegangen.

5.1.2 Metaio Creator Mobile

Bei dem Metaio Creator Mobile handelt es sich um eine App, die für Android-Geräte im Google Play Store unter dem Namen „Creator Mobile“ kostenlos heruntergeladen und installiert werden kann. Entwickler der App ist die Metaio GmbH selbst und stellt diese Software zum Erstellen von Referenzmodellen für das Markerless 3D Tracking-Verfahren zu Verfügung. Die Software kann zusammen mit einem ausgedruckten Marker, ähnlich dem Marker aus Abbildung 5, eingesetzt werden. Der Marker dient dabei als Ursprung für das Referenzmodell. Bei diesem Modell handelt es sich um eine Menge von Punkten im dreidimensionalen Raum, die durch das Werkzeug ermittelt und

aufgezeichnet werden. Lediglich Anwendungen, die mittels des Metaio Mobile SDKs entwickelt wurden, können das erzeugte Referenzmodell in Form einer speziellen Datei verwenden.

Nach dem Start des Metaio Creator Mobile wird in einer Trainingsphase das dreidimensionale Referenzmodell anhand der zweidimensionalen Bilddaten und der verschiedenen Perspektiven der aufnehmenden Kamera erzeugt. Zu diesem Zweck muss der Benutzer das Objekt von verschiedenen Perspektiven durch die Kamera betrachten. Wichtig dabei ist, dass das Modell durch langsame kontinuierliche Bewegungen um das Objekt sukzessive verbessert, bzw. zu Beginn erstellt und optimiert, wird. In der gesamten Trainingsphase ist es notwendig, den Referenzmarker der Trainingsphase komplett im aufgenommenen Bildbereich zu behalten. Nur anhand dieses ID-Markers ist es der Anwendung möglich, geeignete Feature auf dem Objekt relativ zu einander zu ermitteln. Auch die Dauer der Trainingsphase ist für die Qualität der Tracking-Anwendung entscheidend. Eine längere Trainingsphase erhöht die Anzahl der gefundenen dreidimensionalen Referenzpunkte. Die Menge der Referenzpunkte bildet am Ende dieser Lernphase das Referenzmodell des betrachteten Objektes. Dieses Referenzmodell kann anschließend ohne den ID-Marker in der gleichen App getestet werden. Ist das Referenzmodell bezüglich der Qualität noch nicht akzeptabel, kann der Benutzer die Trainingsphase wieder aufnehmen und damit das Modell weiter verbessern. Somit ist es durch dieses Tool möglich, in einer eigenen Anwendung Markerless 3D Tracking zu verwenden. Grundlage für die Qualität dieses Modells ist dabei das optische Verfahren, um Referenzpunkte im Bild zu ermitteln, der Erfahrungswert des Benutzers im Umgang mit der App während der Trainingsphase als auch relativ konstante Lichtverhältnisse. Diese sollten nach Möglichkeit bei der Aufnahme des Referenzmodells nicht von den Verhältnissen zur Laufzeit der AR-Anwendung abweichen. Zusätzlich sind Reflektionen auf dem zu trackenden Objekt, bzw. Ziel, für das Erstellen des Modells als auch für den Betrieb der AR-Anwendung kritisch und sollten vermieden werden. [CREAFAQ]

Das durch das Tool erzeugte Referenzmodell wird in Form einer *3dmap*-Datei abgelegt. Diese Datei muss für die weitere Verwendung in ein ZIP-Archiv umbenannt und entpackt werden. In dem Archiv sind unter anderem die Dateien *TrackingData _ML3D.xml* und *map0.raw* enthalten. Dabei handelt es sich um die XML-Konfigurationsdatei, welche die Parameter des Tracking-Verfahrens definiert, und das Referenzmodell in Form der Referenzpunkte-Liste selbst. Laut Metaio sollen spätere Versionen des Metaio Mobile SDK das direkte Verwenden einer *3dmap*-Datei unterstützen. [TRACON]

Die Qualität eines eingelesenen Objekts kann in der Datei *map.xml* des Archivs ermittelt werden. In dieser Datei wird durch den Zahlenwert des XML-Elements *Stars* eine Wertung der Qualität vergeben. Bisher wurde allerdings bei variabler Trainingsdauer immer maximal eine Qualität von drei Sternen erreicht.

Bei der Datei *map0.xyz* handelt es sich vermutlich um eine durch einen Texteditor lesbare Version des Referenzmodells. In der Datei wird eine Liste von Sechstupel angegeben, wobei die einzelnen Werte eines Sechstupels mit einem Leerzeichen voneinander getrennt werden. Die ersten drei Werte könnten dabei jeweils die Position des Punktes im dreidimensionalen Raum bestimmen. Unklar ist, ob die letzten drei Werte eines Sechstupels die Freiheitsgrade bezüglich der Rotation, Werte der inertialen Sensorik oder aber Farbwerte des referenzierten Bildpunktes beschreiben.

Zusätzlich zu den bereits erwähnten Dateien wird eine PNG-Datei des Zielobjekts hinterlegt, wobei auch die gefundenen Referenzpunkte eingeblendet werden. In Abbildung 24 wird diese Datei mit Namen „thumbnail.png“ dargestellt. Das Archiv, bzw. die *3dmap*-Datei, wurde mit einem Samsung

Galaxy SIII GT-I9300 aufgezeichnet und zeigt das Modell eines BMW 5er Touring, bzw. F11, mit geöffneter Heckklappe.

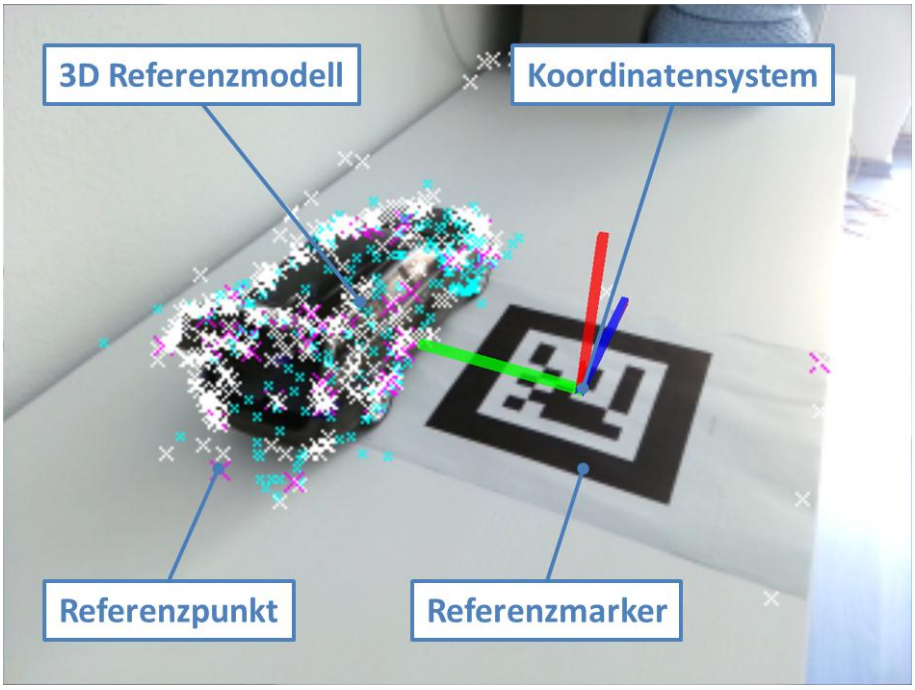


Abbildung 24 Grafische Darstellung des Referenzmodells zum BMW Modell

Empirisch konnte ermittelt werden, dass die Farbe der Kreuze ausschlaggebend für deren Wertigkeit ist und die Größe des dargestellten Koordinatensystems kritisch für eine spätere Anwendung ist. Weiße Kreuze stehen dabei für Merkmale, die noch relativ instabil sind und noch nicht aus verschiedenen Perspektiven erkannt wurden. Blaue Kreuze wurden schon mehrfach erkannt, während magentafarbene Kreuze am häufigsten detektiert wurden. Ein weiteres Indiz für ein funktionierendes Referenzmodell ist die Größe des Koordinatensystems in der Mitte des Referenzmarkers. Nach längerem Training sollte das Koordinatensystem größer dargestellt werden als zu Beginn der Trainingsphase. Erst beim Einsatz der Tablet-Geräte zur Erstellung einer *3dmap*-Datei wurde festgestellt, dass hier dieser Vergrößerungseffekt unabhängig von der Trainingsdauer nicht eintritt. Bei allen getesteten Tablets konnte die Creator Mobile App nicht verwendet werden. Zudem konnte Markerless 3D Tracking auf keinem dieser Geräte unter Verwendung eines funktionsfähigen Referenzmodells eingesetzt werden. Lediglich die Samsung-Geräte konnten für die Erstellung einer funktionsfähigen *3dmap*-Datei genutzt werden.

Aufgrund der Benennung des *type*-Attributs im *Sensor*-Element der XML-Konfigurationsdatei des Archivs mittels „FeatureBasedSensorSource“, könnte man auf den Einsatz von Feature Tracking schließen, welches zur Erstellung des Referenzmodells eingesetzt wird. Wegen der Natur des Verfahrens, das durch die Bewegungen der Kamera dreidimensionale Punkte im Raum als Merkmale für ein Referenzmodell erschließt, wird davon ausgegangen, dass Metaio eine SfM-Methode, wie bereits in Kapitel 1.1.2.4 erwähnt, einsetzt. Nachdem das Referenzmodell erfolgreich erstellt wurde, könnte anschließend in der AR-Anwendung ein modellbasierter Ansatz des Trackings mithilfe dieses Modells verfolgt werden. Aufgrund der geringen Menge an Informationen in der öffentlich zugänglichen Dokumentation von Metaio, bleiben einige Fragen bezüglich der verwendeten Technologien ungeklärt.

Das folgende Kapitel befasst sich mit der Spiele-Entwicklungsumgebung Unity, die im AR-Prototyp eingesetzt wurde. In diesem Kapitel wird vor allem auf die einzelnen Vorteile der Entwicklung mittels Unity für den AR-Prototyp und die Eigenschaften von Unity eingegangen.

5.1.3 Unity3D

Die Spiele-Entwicklungsumgebung Unity wird von der Firma Unity Technologies entwickelt. Durch das Programm wird es Entwicklern ermöglicht komplexe Spieleideen anhand von bereits bestehenden Komponenten umzusetzen. Die einzelnen Komponenten bilden ein Rahmenwerk der notwendigen Softwarebestandteile für ein Computerspiel. Unity verfügt über entsprechende Tools innerhalb der Anwendung, die den Ablauf der Spieleentwicklung stark vereinfachen können. Es werden durch die Unity Engine die folgenden Bereiche abgedeckt:

- **Rendering**, zur Darstellung von Szenarien
- **Lighting**, für die Erstellung eines Beleuchtungsmodells für Szenarien
- **Terrains**, ermöglicht die Oberflächengestaltung der Spielewelt
- **Substances**, erlauben eine dynamische Texturierung der Szenenobjekte
- **Physics**, bzw. eine komplette Physik-Engine in Form der NVIDIA PhysX Engine
- **Pathfinding**, zum Auffinden von Bewegungspfaden für Spielobjekte zur Laufzeit
- **Audio**, zur Erstellung einer komplexen akustischen Umgebung im Spiel
- **Programming**, unterstützt die Erstellung der Spielelogik anhand von verschiedenen Skript-Sprachen
- **Network**, für die Anbindung der Inhalte an ein Netzwerk für Mehrspieler-Anwendungen

Auf die einzelnen Bereiche wird hier nicht explizit eingegangen. Lediglich die verwendeten Bereiche von Unity bezüglich des AR-Prototyps werden im weiteren Verlauf beschrieben. [UNITYE]

Ein besonderer Vorteil bei dem Einsatz von Unity für den AR-Prototyp ist, dass der Arbeitsaufwand bezüglich der Umsetzung von Animationen und Darstellung dreidimensionaler Objekte relativ gering ist. Die Animation von Charakteren wurde allerdings nicht mit Unity selbst erstellt, sondern mit Autodesk MotionBuilder und Autodesk Maya durchgeführt. Ein Charakter beschreibt dabei ein virtuelles menschenähnliches Modell. Der Umgang mit diesen Programmen wird im Rahmen dieser Arbeit nicht behandelt. In Kapitel 5.2.1 wird auf Charakter-Animationen eingegangen. Dabei wird deren Integration in das Unity-Projekt des AR-Prototyps und deren Verwendung in Verbindung mit nativen Unity-Animationen beschrieben.

Anhand der Unity-Software ist es möglich Inhalte plattformübergreifend zu entwickeln. Unterstützt werden je nach Lizenz unterschiedliche Systeme. Außer auf modernen Spielekonsolen wie Microsoft XBox 360, Sony Playstation 3 und der Nintendo Wii können Inhalte auch auf Mac, PC und den Betriebssysteme für mobile Anwendungen, wie iOS und Android, ausgeführt werden. Von Unity Technologies werden ständig weitere Plattformen in das Portfolio der unterstützen Systeme aufgenommen. Ein Beispiel dafür ist Unity Version 4, das sich im Beta-Stadium befindet, und die Veröffentlichung von Inhalten auf Adobe Flash Player als auch auf Linux unterstützt. [UNITY4]

Diverse Inhalte können in Unity in Form von Paketen für die weitere Verwendung importiert werden. Auch das Metaio Mobile SDK stellt eine *unitypackage*-Datei zur Verfügung die in ein Unity-Projekt integriert werden kann. Damit können Funktionalitäten des SDKs im Zusammenhang mit den Eigenschaften der Unity-Entwicklungsumgebung verwendet werden.

In Abbildung 25 wird die Unity-Benutzerschnittstelle des AR-Prototyp-Projekts dargestellt. Auf der linken Seite wird in der Szenenansicht der weibliche Charakter dargestellt, der bereits in Abbildung 23 gezeigt wurde. Zusätzlich wird in Abbildung 25 ein grauer Quader abgebildet, der das reale Fahrzeug der AR-Anwendung ersetzt. Für den späteren Einsatz im Prototyp wurde der Quader vor der Verteilung auf das mobile Gerät ausgeblendet. Manipulationen dieser Art können in der Szene durch Unity relativ schnell und komfortabel umgesetzt werden.

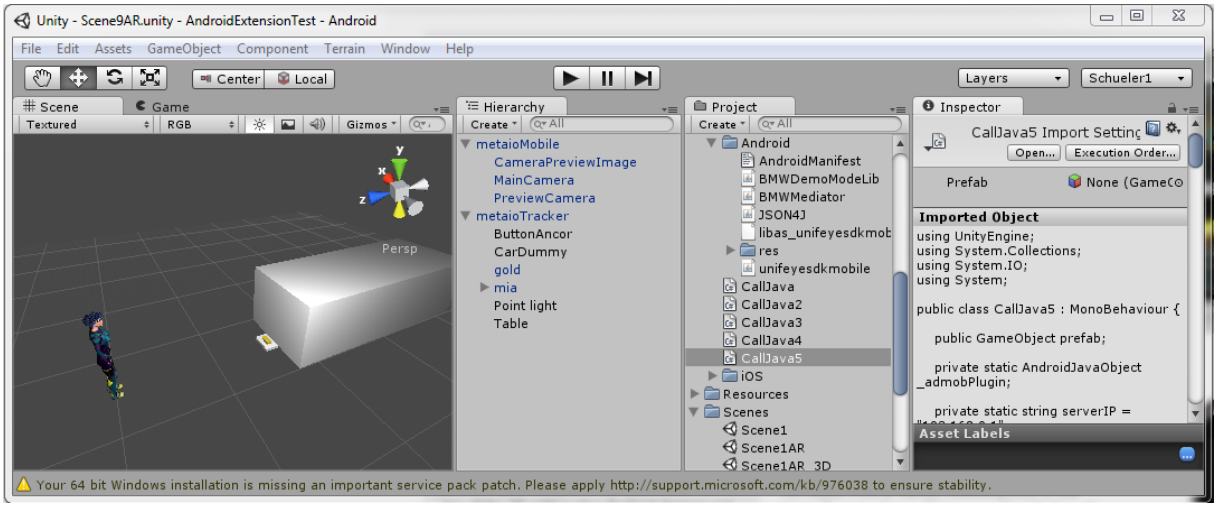


Abbildung 25 Unity-Benutzerschnittstelle des AR-Prototyp-Projekts

Unity dient demnach als Adapter, um die Funktionalität des Metaio Mobile SDKs auf mobilen Endgeräten einsetzen und weitere Inhalte komfortabel hinzufügen zu können. Dabei kann auf vorhandene Tools zur Animation des Charakters in der Szene zurückgegriffen werden. Die Darstellung der Szene lässt sich ohne weiteren Implementierungsaufwand durchführen. Lediglich für die Anbindung der Schnittstelle zur Programmlogik, die bereits durch die Entwicklung des Demonstrators vorhanden ist, musste weiterer Quellcode in der Programmiersprache C# geschrieben werden.

Im weiteren Verlauf wird auf besondere Techniken bei der Verwendung von Unity eingegangen. Dabei handelt es sich um Techniken, welche für die Realisierung des AR-Prototyps im Bereich der Animation eingesetzt wurden. Durch einen Animations-Assistenten ist es in Unity möglich, alle Freiheitsgrade eines Objektes entlang eines zeitlichen Verlaufs an definierten Stellen festzulegen. Werte in den Zwischenbereichen werden dabei von dem Assistenten nach gewünschten Vorgaben interpoliert. Dadurch entsteht eine flüssige Bewegung beim Abspielen der Animation.

Für den Übergang zwischen den verschiedenen Animationen des Charakters kann die Technik des *Animation Blending* eingesetzt werden. Dabei werden die Grenzen bei der Aneinanderreihung von Animationen des Charakters ohne Sprünge der Bewegungskurve ineinander überführt. Dadurch entsteht beispielsweise zwischen der Animation eines stehenden Charakters und der Animation eines laufenden Charakters bei Aneinanderreihung der Eindruck, dass dieser auf realistische Weise kontinuierlich in eine Laufbewegung übergeht.

Eine weitere im Prototyp eingesetzte Technik wird als *Animation Layering* bezeichnet. *Animation Layer* erlauben es mehrere Animationen in verschiedenen Schichten gleichzeitig ablaufen zu lassen. Verwendet man demnach zwei dieser Schichten, ist es möglich die Bewegung beim Winken eines Charakters mit der Bewegung des Laufens zu verschmelzen auch wenn die Winken-Animation für

einen stehenden Charakter erzeugt wurde. Auf diese Weise ist es möglich die Bewegungen der Charaktergelenke und die Bewegung des gesamten Charakters in der Szene zu trennen und unabhängig voneinander einzusetzen. Somit kann die Animation des Charakters als Summe verschiedener Animationen betrachtet werden. Die Translationsbewegung des Charakters in der Szene und die Bewegungen der einzelnen Gelenke des Charakters ergeben dabei in Summe erst die gesamte Animation die zum Beispiel eine Laufbewegung des Charakters darstellt. Dadurch können Animationen des Charakters sehr flexibel eingesetzt werden. Zusammen mit dem *Animation Blending*, welches für weiche Übergänge zwischen den Animationen sorgt, entsteht so der Eindruck einer realistischen Bewegung des Charakters in der Szene.

Um auf verschiedene Situation in der Szene bezüglich der Animationen reagieren zu können, existieren in Unity sogenannte *Animation-Events*. Dabei handelt es sich um Ereignisse, die zum Aufruf einer Methode, bzw. Funktion, von Unity-Quellcode führen. Im Falle des AR-Prototyps wird dabei C#-Quellcode aufgerufen. *Animation-Events* können durch den Animations-Assistenten an bestimmten Stellen des zeitlichen Verlaufs einer Animation gesetzt werden. Erreicht die Animation diese Stelle, wird der zugewiesene Quellcode ausgeführt. Dabei ist es notwendig, dass die Quellcode-Datei an die *GameObject*-Instanz gebunden ist, welche die Animation ausführt. Bei der Klasse *GameObject*[37] handelt es sich um die Basisklasse aller Objekte innerhalb einer Szene eines Unity-Projekts. Grundsätzlich ist jeder dreidimensionale Körper in Unity als *GameObject*-Instanz zu verstehen. Aus dieser Einschränkung beim Aufruf von Quellcode ergibt sich, dass Aufrufe von Methoden, welche nicht in dieser Datei enthalten sind, weitergeleitet werden müssen. Dieser Forward-Mechanismus und dessen Vorteile werden bezüglich der Anwendung im AR-Prototyp in Kapitel 5.2.3 genauer erläutert. In Abbildung 26 wird der Animations-Assistent zusammen mit der Animation des Goldbarrens im AR-Prototyp angezeigt. Dabei wurden genau vier Schlüssel-Frames gesetzt, die alle Freiheitsgrade definieren. Die Verläufe zwischen diesen Frames wurden vom Assistenten automatisch erzeugt. Am Ende der Animation wird ein *Animation-Event* ausgelöst, das die Methode *forward()* aufruft.

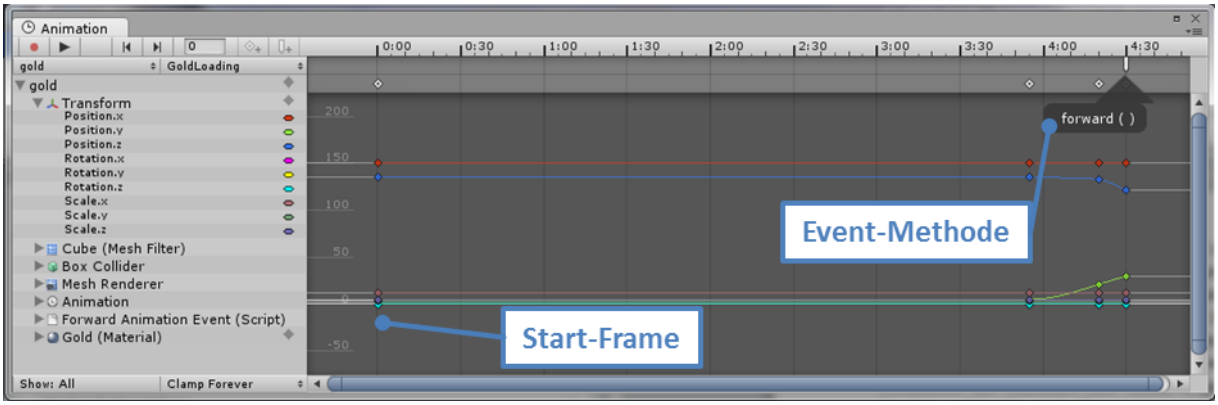


Abbildung 26 Animations-Assistent in Unity, Darstellung der Animation des Goldbarrens und eines Animation-Events

Im folgenden Kapitel wird auf die logische Struktur des Prototyps und dessen Verhalten eingegangen. Des Weiteren wird der Arbeitsablauf, bzw. Workflow, bei der Entwicklung des AR-Prototyps in Verbindung mit der Verteilung auf ein Android Endgerät beschrieben.

[37] Siehe http://docs.unity3d.com/Documentation/ScriptReference/GameObject.html

5.2 Aufbau und Entwicklung des Prototyps

Anhand der in den vorherigen Kapiteln behandelten Werkzeuge, wurde der Anwendungsfall, welcher bereits in Kapitel 5 beschrieben wurde, realisiert. Die Entwicklung des AR-Prototyps lässt sich in verschiedene Aufgabengebiete unterteilen, die im weiteren Verlauf einzeln betrachtet werden. Darunter fällt unter anderem auch das Erstellen der dreidimensionalen Objekte für die Szene des Anwendungsfalls. Für den Anwendungsfall der Niveauregulierung sollte eine virtuelle Person Gewichte in Form von Goldbarren in das reale Fahrzeug einladen, welches sich durch diese Tätigkeit am Heck absenkt. Da keine entsprechenden Objekte in der Medien-Datenbank von BMW vorhanden waren mussten diese selbst entwickelt werden. Auch die Animation dieser Objekte, insbesondere eines dreidimensionalen Charakter-Modells, musste im Zuge der Masterarbeit realisiert werden.

Im nächsten Unterkapitel wird auf die für den Prototyp verwendeten dreidimensionalen Objekte und deren Animation eingegangen. Dabei werden Techniken zur Animation von Charakteren angesprochen. Aus Gründen der Übersicht wird auf eine detaillierte Beschreibung zum Vorgehen bei der Charakter-Animation verzichtet.

5.2.1 Dreidimensionale Objekte und Animation

In Abbildung 25 wurden bereits die virtuellen Bestandteile der AR-Anwendung dargestellt. Dabei handelt es sich um den Akteur in Form einer weiblichen Person, einen Tisch auf dem die Goldbarren liegen und die Goldbarren selbst. Goldbarren als auch Tisch konnten wegen der primitiven Form direkt in Unity erstellt werden. Bei dem Akteur handelt es sich um eine Person, die über entsprechende Animationen zum Beladen des Fahrzeugs und zum Laufen verfügen muss. Im ersten Anlauf wurde der Versuch unternommen mittels Autodesk Maya an ein bereits existierendes Modell einer männlichen Person ein Animationsskelett, bzw. Character-Rig, hinzuzufügen. Dieser Vorgang wird auch als Character Rigging bezeichnet. Das Skelett ist dabei notwendig, damit bei einer Bewegung der Figur anhand des Skeletts festgelegt werden kann, welche Bereiche der Oberfläche des Modells davon betroffen sind und sich demnach mitbewegen müssen. Die Haltung des Charakter-Modells wich dabei stark von der optimalen T-Pose ab, die für das Erstellen eines Rigs eingesetzt werden sollte. Durch Überschneidungen der Gliedmaßen, bzw. Oberflächen, führte dies zu unerwünschten Effekten. Ein Beispiel dafür ist, dass sich bei einer Bewegung des Oberarms die Oberfläche der Hüftgegend unnatürlich verzerrt hat. Wegen der minderwertigen Qualität des Ergebnisses wurde dieses Modell verworfen. Stattdessen wurde ein bereits bestehendes Modell, welches über ein Skelett verfügt, verwendet. Das Programm Autodesk MotionBuilder verfügt über einige Beispiele von Charakteren. Hier konnte das Mia-Modell verwendet werden, um die notwendigen Animationen zu erstellen, da dieses Modell bereits über ein Character-Rig verfügt. Für den Anwendungsfall der Niveauregulierung mussten anschließend unterschiedliche Charakter-Animationen erstellt werden.

Eine *idle*-Animation wurde erstellt, um dem Charakter, selbst ohne den Auftrag etwas einzuladen, lebendig erscheinen zu lassen. Bei dieser Animation steht der Charakter ruhig auf einer Stelle und bewegt einzelne Gelenke minimal. Dies entspricht einer Wartehaltung des Characters.

Als zweite Animation, bzw. *walk*-Animation, wurde ein sogenannter Walk-Cycle[38] erstellt. Dabei vollführt der Charakter zwei Schritte und endet in der Haltung mit der die Animation begonnen hatte. Durch Aneinanderreihung dieser Animation entsteht der Eindruck als würde der Charakter laufen. Wichtig bei dieser Animation ist, dass die Person in MotionBuilder auf der Stelle läuft. Die

[38] Ein Walk-Cycle stellt den zyklischen Ablauf der Laufbewegungen dar.

zurückgelegte Strecke wird gesondert mittels einer Animation in Unity durchgeführt. Dabei handelt es sich um eine Translationsbewegung des Charakters unabhängig von dessen Bewegungen durch das Skelett. Die hier eingesetzte Technik wird im Unity-Kontext als *Animation Layering* bezeichnet und wurde bereits im Kapitel 5.1.3 erläutert.

Bei der letzten Animation, der *lift*-Animation, handelt es sich um die Bewegung des Charakters um einen Goldbarren anzuheben und diesen in das Heck des Fahrzeuges einzuladen. In Abbildung 27 wird das Mia-Modell in der Haltung am Ende der *walk*-Animation dargestellt.

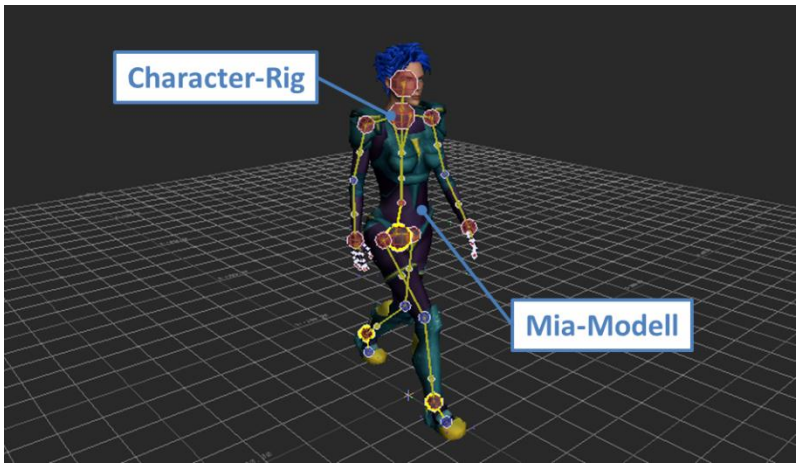


Abbildung 27 Animation des Mia-Modells in MotionBuilder

Um das animierte Modell in Unity einsetzen zu können musste dieses an das Programm Autodesk Maya weitergleitet und anschließend durch Maya als MB-Datei gespeichert werden. Diese Datei kann schließlich in Unity importiert und verwendet werden.

Für das Ansteuern der Niveauregulierung im Anwendungsfall ist es notwendig, dass die Funktionalität bezüglich der Client-Server-Kommunikation in das Unity-Projekt integriert wird. Im nächsten Kapitel wird dieser Arbeitsschritt detailliert zusammen mit dem strukturellen Aufbau der Software beschrieben.

5.2.2 Struktur zur Kommunikation zwischen AR-Anwendung und Aktuatorik

Zur indirekten Verbindung der Aktuatorik des Fahrzeugs mit dem AR-Prototyp ist es notwendig, dass einige Komponenten, die bereits aus der Beschreibung des Demonstrators bekannt sind, im Prototyp wiederverwendet werden. Zu diesem Zweck wurden die Klassen, welche die Funktionalität für eine Client-Server-Kommunikation zur Verfügung stellen, in das JAVA-Archiv *BMWDemoModeLib* gepackt.

Als Bindeglied zwischen den Klassen des Demonstrators und der Logik des AR-Prototyps musste eine weitere vermittelnde Komponente entwickelt werden. Grund dafür ist, dass die Programmiersprache JAVA in Unity nicht direkt unterstützt wird. Um JAVA Quellcode als Plugin in Unity nutzen zu können werden JNI Klassen[39] und JNI Helferklassen[40] verwendet. Darunter fallen unter anderem auch die Klassen *AndroidJavaClass* und *AndroidJavaObject* die für Aufrufe der JAVA Methoden eingesetzt werden können.

[39] Siehe http://docs.unity3d.com/Documentation/ScriptReference/AndroidJNI.html
[40] Siehe http://docs.unity3d.com/Documentation/ScriptReference/AndroidJNIHelper.html

Bei der vermittelnden Komponente handelt es sich um die Klasse *CarMediator,* die in JAVA implementiert wurde und sich in dem Archiv *BMWMediator* befindet. Anhand dieser Klasse ist es möglich Methodenaufrufe des nativen Unity Quellcodes in C# an Methodenaufrufe des JAVA-Quellcodes der Demonstrator-Funktionalität weiterzuleiten. Um Ergebnisse, bzw. Antworten in Form von Callback-Methodenaufrufen, ausgehend vom JAVA Quellcode an den C# Quellcode aus Untiy weiterleiten zu können, wird die statische Methode *UnitySendMessage()* der Klasse *com.unity3d.player.UnityPlayer* verwendet. [UNIJNI]

In Abbildung 28 wird die Aufrufstruktur zwischen C# Quellcode und JAVA Quellcode stark abstrahiert dargestellt.

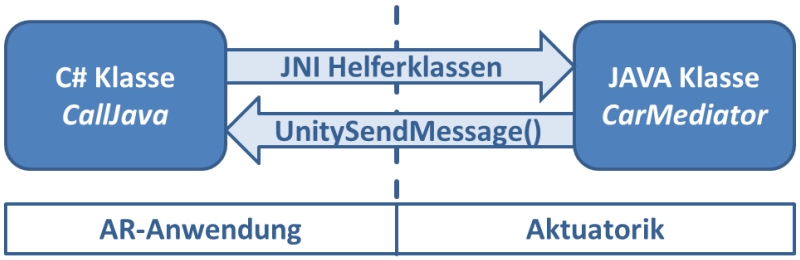


Abbildung 28 Aufrufstruktur zwischen der AR-Anwendungs- (links) und Aktuatorikbereich (rechts)

Die Klasse *CarMediator* kapselt die Funktionalität, um die Niveauregulierung anzusteuern. Da in der aktuellen Version des Servers das Ansteuern der Niveauregulierung nicht unterstützt wird, wurde hier die Schnittstelle der Aktivlenkung als Platzhalter verwendet. Sollte der Server erweitert werden, kann der AR-Prototyp simpel angepasst werden, da hier lediglich Umbenennungen im Quellcode vorgenommen werden müssen. Das Verhalten der Kommunikationskomponenten und Schnittstellen unterscheidet sich nur unerheblich voneinander. Besonders wird dies an den Methoden *lowerCar()* und *elevateCar()* deutlich, die das Absenken, bzw. Anheben, des Fahrzeughecks bewirken sollen. Diese Methoden kapseln lediglich den Methodenaufruf einer *Communicator*-Erweiterung. Bei einer veränderten Version müsste an dieser Stelle lediglich analog ein Aufruf des Niveauregulierungs-Communicators, statt des hier verwendeten *STEERINGCommunicators*, mit entsprechender Methode durchgeführt werden. In Tabelle 4 wird der Quellcode der Methoden *lowerCar()* angezeigt. Analog dazu ist die *elevateCar()*-Methode aufgebaut.

Die Struktur des AR-Prototyps bezüglich der einzelnen Klassen wird zur Übersicht in Abbildung 29 in einem UML-Klassendiagramm dargestellt. Dabei wird sowohl auf die Klassen der JAVA Umfänge als auch auf Klassen in C# eingegangen. Zur Übersichtlichkeit wurde in dem Diagramm auf Attribute verzichtet. Zudem werden nur für das Verständnis relevante Klassen angezeigt. Die Klasse *CommunicationManager* als auch die beiden Schnittstellen *CallbackSteering* und *CallbackCar* wurden bereits in Kapitel 4.2.2 ausgiebig behandelt.

In Abbildung 29 wird die Klasse *CallJava5* und einigen Methoden dieser Klasse dargestellt. Diese Klasse aus dem C#-Bereich ist eine Erweiterung der Klasse *MonoBehaviour*[41]. Jede C#-Klasse in einem Unity-Skript muss diese Klasse erweitern, um die Funktionsfähigkeit im Unity-Kontext zu gewährleisten. Die *CallJava5* Klasse ermöglicht den Übergang zu Aufrufen der Android API und wird selbst von der Klasse *CarMediator* aufgerufen. Die Methoden der *CallJava5*-Klasse lösen entsprechende Aufrufe der Methoden der *CarMediator*-Klasse aus. Beispielsweise wird durch einen

[41] Siehe http://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.html

Aufruf der C#-Methode *ConnectToServer()* die JAVA-Methode *startCommunication()* ausgeführt. Um Zugriff auf Systemdienste zu erhalten, muss eine Referenz der aktuellen *Activity*, bzw. *com.unity3d.player.UnityPlayerNativeActivity*, von der Klasse *CallJava5* ermittelt und an die *startCommunication()*-Methode übergeben werden. Diese Referenz kann über die Klasse *com.unity3d.player.UnityPlayer* abgefragt werden. Dadurch ist die Klasse *CarMediator* die zentrale Komponente, an der alle Aufrufe zusammenlaufen. Das Archiv *BMWMediator* besteht lediglich aus dieser einen Klasse, während das Archiv *BMWDemoModeLib* alle Klassen und Pakete enthält, die für den Demonstrator und zur Kommunikation mit dem Server implementiert wurden. Lediglich die spezifische Anwendungslogik des Pakets *com.android.BMW.DemoMode* ist nicht in diesem Archiv enthalten. Die Anwendungslogik des AR-Prototyps wird in der C#-Klasse *CallJava5* gekapselt. Durch die beiden Archive *BMWMediator* und *BMWDemoModeLib* ist es möglich aus nativem Unity Quellcode heraus Effekte der Aktuatorik hervorzurufen.

Im folgenden Abschnitt wird detailliert auf das Verhalten des AR-Prototyps anhand eines Beispiels eingegangen. Darüber hinaus werden einige Mechanismen im Umgang mit Unity beschrieben.

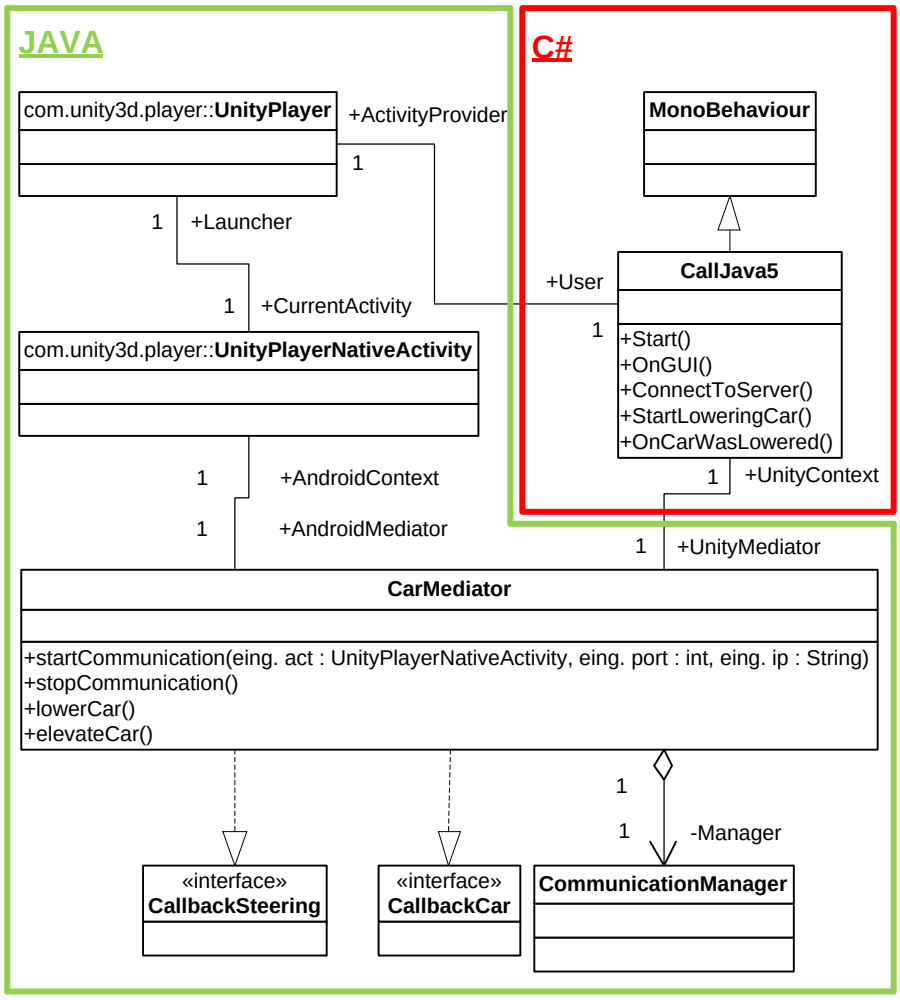


Abbildung 29 UML-Klassendiagramm, Auszug des AR-Prototyps der JAVA- (grün) und C#-Klassen (rot)

5.2.3 Verhalten der Komponenten des AR-Protoyps

Um die Aufrufstruktur zwischen C# und JAVA Quellcode anhand der Verwendung im AR-Prototyp zu erläutern, wird als Beispiel der Vorgang des Absenkens des Fahrzeugs im weiteren Verlauf detailliert beschrieben.

Wie bereits in Abbildung 29 zu erkennen ist, verfügt die Klasse *CallJava5* unter anderem über die Methode *Start()*, die zu Beginn der AR-Anwendung einmalig aufgerufen wird. In dieser Methode werden JNI Aufrufe durchgeführt, um eine Referenz auf die aktuelle *UnityPlayerNativeActivity* zu erhalten. Diese Referenz ist notwendig, um bestimmte Methoden und Klassen der Android API verwenden zu können. In Tabelle 3 wird der Ausschnitt der *Start()*-Methode dargestellt, welcher durch JNI Aufrufe entsprechende Referenzen für den späteren Gebrauch in der C#-Klasse hinterlegt.

```
1  _unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
2
3  AndroidJavaObject currentActivity =
   _unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
4
5  using( var pluginClass = new
   AndroidJavaClass("com.schueler.home.transfer.CarMediator"))
6
7  _admobPlugin = pluginClass.CallStatic<AndroidJavaObject>( "instance" );
```

Tabelle 3 Ausschnitt des C# Quellcodes der Start()-Methode

In der dritten Zeile wird die Klassenvariable *currentActivity* der Klasse *UnityPlayer* abgefragt und entsprechend als Referenz in einer *AndroidJavaObject*-Instanz hinterlegt. Des Weiteren wird in Zeile fünf eine Referenz auf die Klasse *CarMediator* in Form einer *AndroidJavaClass*-Instanz erzeugt. Diese Referenz wird in der siebten Zeile verwendet um die statische Methode *instance()* der Klasse *CarMediator* aufzurufen. Es wird darauf hingewiesen, dass in Zeile fünf keine Instanz der Klasse *CarMediator* entsteht. Eine entsprechende Instanz wird durch einen Aufruf der *instance()*-Methode erzeugt, bzw. eine bereits bestehende Instanz zurückgegeben. Durch die Klasse *CarMediator* wird demnach das *Singleton*-Entwurfsmuster[42] umgesetzt. Durch die Implementierung als *Singleton* ist ein minimaler Speicherbedarf zur Laufzeit notwendig, da es zu jeder Zeit nur eine Instanz der Klasse geben kann. [GAMMADP]

Durch die Methode *Start()* sind alle für die Anwendungslogik notwendigen Instanzen bekannt, um anschließend die Verbindung zur Aktuatorik des Fahrzeugs zu etablieren. Zu diesem Zweck wird bei Anwendungsstart die Methode *ConnectToServer()* eingesetzt, welche einen Aufruf der *startCommunication()*-Methode der Klasse *CarMediator* auslöst. Die *startCommunication()*-Methode leitet dementsprechend den Verbindungsaufbau mit dem *CommunicationManager* und den Kanälen ein. Durch den Aufruf der Callback-Methoden wird analog zum Demonstrator der erfolgreiche Verbindungsaufbau quittiert. Dabei wird über die Methode *UnitySendMessage()* die C#-Methode *Callback()* der *CallJava5*-Klasse aufgerufen, welche indirekt die Darstellung eines Knopfes in der GUI bewirkt. Durch drücken diese Knopfes kann der Benutzer den Einlade-Vorgang, bzw. die Animation des Charakters, starten. Um zu ermitteln, zu welchem Zeitpunkt sich der virtuelle Goldbarren im Heck des Fahrzeugs befindet und sich dieses Absenken muss, wurde die Unity Technik der *Animation-Events* eingesetzt. Somit wird, sobald der Goldbarren eine bestimmte Position erreicht, die Methode *forward()* des mit dem Goldbarren verbundenen C#-Skripts *ForwardAnimationEvent* aufgerufen. Die Verbindung zwischen Objekten in Unity und C#-Skripts wird später im Zusammenhang mit dem

[42] Beschreibt eine Klasse, die sicherstellt, dass zu jeder Zeit nur eine Instanz dieser Klasse existiert. [GAMMADP]

CallJava5-Skript detailliert erläutert. Durch die Methode *forward()* wird der Aufruf an die Klasse *CallJava5* weitergeleitet. Dieser zusätzliche Aufruf ist notwendig, da *Animation-Events* einigen Einschränkungen unterliegen, welche bereits in Kapitel 5.1.3 erläutert wurden. Durch diese Weiterleitung wird die Methode *StartLoweringCar()* der Klasse *CallJava5* ausgeführt. Grundsätzlich wäre es auch möglich gewesen, den entsprechenden Quellcode direkt in der Klasse *ForwardAnimationEvent* zu implementieren. Um die Komplexität der Struktur nicht unnötig zu steigern, wurde auf Kosten eines zusätzlichen C#-Methodenaufrufs der Ansatz zur Weiterleitung gewählt. Ein weiterer Vorteil dieses Ansatzes ist, dass die Klasse *CallJava5* die Funktionalitäten, welche einer starken Bindung unterliegen, als zentrale Komponente vereint.

Die Methode *StartLoweringCar()* gilt als Anfang der Aufrufstruktur, um das Fahrzeug abzusenken. Durch diese Methode wird mittels JNI-Aufruf die Methode *lowerCar()* der Klasse *CarMediator* aufgerufen. In Tabelle 4 wird der Quellcode der *lowerCar()*-Methode dargestellt.

```
1  /**
2  * This method is used to call the car-interface to lower the rear of the car.
3  */
4  public void lowerCar() {
5        comSteer.setSteeringAngle((CallbackSteering) this, iValueLowerCar);
6  }
```

Tabelle 4 JAVA Quellcode der lowerCar()-Methode

In Zeile fünf der Tabelle 4 wird der *SteeringCommunicator comSteer* verwendet, um den Lenkwinkel auf *iValueLowerCar* zu setzten. Alle Antworten des Servers werden an die Klasse *CarMediator* übermittelt, die selbst das *CallbackSteering*-Interface implementiert. Zu diesem Zweck wird eine Referenz auf die *CarMediator*-Instanz als erster Parameter an die *setSteeringAngle()*-Methode übergeben. Bei einer Antwort durch den Server werden Methoden der Interface-Implementierung per Callback zurückgerufen. In diesen Callback-Methoden können Informationen über Aufrufe der *UnitySendMessage()*-Methode an die AR-Anwendung, bzw. eine C#-Klasse, weitergeleitet werden. Der *UnitySendMessage()*-Methode werden der Name der *GameObject*-Instanz und der Name der aufzurufenden C#-Methode als Parameter übergeben. Damit die *GameObject*-Instanz über den entsprechenden Quellcode verfügt, muss dieser in Form der Skript-Datei an das Objekt der Szene gebunden werden. In Abbildung 30 wird die *GameObject*-Instanz *ButtonAncor* und das damit verknüpfte C#-Skript *CallJava5* angezeigt. In dem *Inspector*-Reiter auf der rechten Seite wird in dem roten Kasten die Referenz auf das C#-Skript dargestellt.

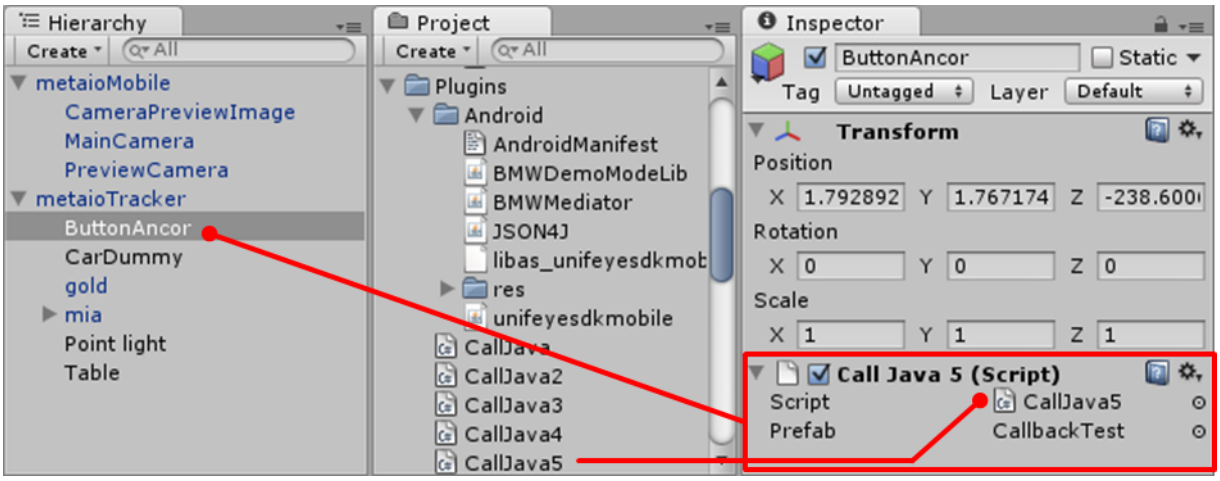


Abbildung 30 GUI Ausschnitt von Unity, Zusammenhang zwischen GameObject-Instanz und C#-Skript (rot)

Ein Beispiel für den Aufruf einer Methode des C#-Skripts *CallJava5* wird in Tabelle 5 am Aufruf der Methode *setSteeringAngleReady()* gezeigt. Dabei wird allerdings aus Gründen der Übersicht nur ein Ausschnitt des Methoden-Quelltextes abgebildet, der auf den Aufruf der Methode *lowerCar()* folgt. Die Methode *setSteeringAngleReady()* wird vom *CommunicatorSTEERING*-Thread aufgerufen, sobald vom Server die Nachricht erhalten wurde, dass die Aktuatorik erfolgreich manipuliert wurde. Auch wenn bei der aktuellen Version des AR-Prototyps die Manipulation der Lenkungsaktuatorik erfolgt, wird darauf hingewiesen, dass zu diesem Zeitpunkt der logische Schritt des Absenkens des Fahrzeugs stattgefunden hat. Somit befindet sich das Heck des Automobils fortan bis zur Aktivierung der Niveauregulierung auf einem niedrigeren Niveau als ohne virtuelle Last.

In der sechsten Zeile von Tabelle 5 wird der Aufruf der statischen Methode *UnitySendMessage()* gezeigt. Dieser Aufruf führt zur Ausführung der *OnCarWasLowered()*-Methode des *CallJava5*-Skripts, welches mit der *GameObject*-Instanz *ButtonAncor* verbunden ist. Auf diese Weise wird signalisiert, dass das Ereignis des Absenkens des Hecks am Fahrzeug eingetreten ist.

```
1  public void setSteeringAngleReady(int newValue) {
2    Log.i(TAG, "setSteeringAngleReady() called!");
3
4    if (newValue == iValueLowerCar) {
5      Log.i(TAG, "Car was lowered to the defined value of " + iValueLowerCar);
6      UnityPlayer.UnitySendMessage("ButtonAncor", "OnCarWasLowered", null);
```

Tabelle 5 Ausschnitt des JAVA Quellcodes der setSteeringAngleReady()-Methode

Anhand der Methode *OnCarWasLowered()* wird dann Einfluss auf die Darstellung der AR-Inhalte durch Unity genommen. Durch diesen Ansatz wird ein ereignisgetriebenes Verhalten der Software bezweckt. Durch dieses Verhalten kann durch die Software unmittelbar auf Ereignisse, die das System betreffen, reagiert werden. Besonders im Zusammenhang mit der GUI, die abhängig von den Ereignissen verändert werden soll, ist dies von Vorteil.

Durch den Aufruf der *OnCarWasLowered()*-Methode werden zu diesem Zweck lediglich zwei Variablen neue Werte zugewiesen. Dabei handelt es sich um die Variable *bCarIsLow* und *bSelfLevelingIsActive*. Durch die Zuweisung neuer Werte wird die GUI unter Umständen im nächsten Darstellungszyklus verändert. Für jeden dieser Zyklen wird einmalig die Methode *OnGUI()* aufgerufen, welche die genannten Variablen verwendet, um zu ermitteln, welche GUI-Elemente gezeichnet werden sollen. Auch Fehlerfälle werden berücksichtigt und durch verschiedene Variablen

ausgedrückt. Somit ist die Darstellung im einzelnen Fall von allen diesen Zustandsvariablen gemeinsam abhängig. In Tabelle 6 wird ein Ausschnitt aus der Methode *OnGUI()* des *CallJava5*-Skripts dargestellt, der dieses Design am Beispiel des Knopfes zur Aktivierung der Niveauregulierung erläutert.

```
1 // The car is in lowered pose because there is gold in the trunk
2 if (!bSelfLevelingIsActive) {
3       if (GUI.Button(new Rect(Screen.width/2 + 205 , Screen.height - 80, 200,
        75), "Niveauregulierung aktivieren")) {
4             bSelfLevelingIsActive = true;
5             StartElevatingCar();
6       }
7 }
```

Tabelle 6 Ausschnitt des C# Quellcodes der OnGUI()-Methode

Dieser Quellcode wird nach dem Aufruf der *OnCarWasLowered()*-Methode im nächsten Darstellungszyklus innerhalb der *OnGUI()*-Methode ausgeführt. Dabei wird in der zweiten Zeile überprüft, ob die Niveauregulierung bereits vorher aktiviert wurde. Ist diese noch nicht aktiviert, wird durch den Code der dritten Zeile ein Knopf gezeichnet, der mit dem Text „Niveauregulierung aktivieren“ beschriftet wird. Drückt der Benutzer diesen Knopf, wird durch das Setzen der Variablen *bSelfLevelingIsActive* verhindert, dass im nächsten Zyklus der Knopf erneut erscheint. Damit ist der Knopf nach dem Drücken unmittelbar verschwunden. Zusätzlich wird durch den Methodenaufruf in Zeile fünf indirekt die Aktuatorik des Fahrzeugs angesprochen, um das Fahrzeug wieder bezüglich des Niveaus am Heck zu regulieren. Dieser Ablauf wird vom Standpunkt der Implementierung aus analog abgewickelt.

Zusätzlich zur bereits erläuterten Funktionalität wurden noch Mechanismen integriert, die den Umgang mit der AR-Anwendung komfortabler gestalten sollen. Damit beispielsweise bei der Verwendung der AR-Anwendung nach einem Neustart Verbindungsinformationen zum Server nicht verloren gehen, können diese durch Methoden der *CarMediator* Klasse gesichert werden. Somit sind diese Informationen vor Verlust geschützt und müssen bei einem Neustart nicht erneut eingegeben werden. Zusätzlich kann über einen Menü-Dialog die Verbindung zu einem anderen Server aufgebaut werden. Im AR-Prototyp werden diese Mechanismen bereits eingesetzt.

Damit das animierte Modell, die verschiedenen Skripte und Archive im Kontext einer AR-Anwendung eingesetzt werden können, musste ein entsprechendes Projekt in Unity angelegt werden, welches die einzelnen Komponenten miteinander verbindet. In dem folgenden Kapitel werden dazu der Aufbau des Projektes in Unity und die Logische Struktur darin detailliert erläutert. Darüber hinaus wird auf die Arbeitsweise zur Entwicklung eines JAVA-Plugins, bzw. Android-Plugins eingegangen.

5.2.4 Projekt-Aufbau in Unity und Entwicklungsschritte für Android-Projekte

Die vom Metaio Mobile SDK zur Verfügung gestellte *unitypackage*-Datei beinhaltet alle notwendigen Dateien, um eine AR-Anwendung innerhalb eines Unity-Projektes zu realisieren. Die Datei „metaioMobile.unitypackage“ der Metaio Mobile SDK Installation muss dazu in das Unity-Projekt importiert werden. Durch diesen Import-Vorgang werden alle Dateien in das Projektverzeichnis kopiert und können anschließend verwendet werden. Für die Entwicklung des AR-Prototyps konnte eine Beispiel-Szene des Paketes mit Dateinamen „ARScene.unity“ als Grundlage verwendet werden. Die Szene des AR-Prototyps verfügt auf der obersten Hierarchiestufe, wie auch die Beispiels-Szene,

über zwei verschiedene *GameObjects*. In Abbildung 30 wird die Hierarchie dieser Szene unter dem Reiter *Hierarchy* auf der linken Seite dargestellt.

Das *metaioMobile*-Object kapselt verschiedene *GameObjects*, die für die Darstellung der Inhalte eingesetzt werden. Von Besonderer Bedeutung ist das C#-Skript *metaioMobile*, welches an genau dieses *GameObject* gekoppelt wurde. Dort werden zu Beginn die Parameter bezüglich der Auflösung der Gerätekamera angegeben. Als Trade-off zwischen Performanz und möglichst hoher Bildauflösung wurden hier als Kamerabreite 720 Pixel und für die Kamerahöhe 576 Pixel festgelegt. Dieses Skript startet die Sensorik des mobilen Endgeräts in Form der Kamera mit entsprechender Auflösung als auch der intertialen Sensoren. Dateien, die sich im Verzeichnis „Resources/metaio“ befinden, werden beim Start der Anwendung in eine Datenstruktur eingelesen, um später schnellere Zugriffszeiten während der Laufzeit zu erhalten. Entscheidend ist, dass die globalen Variablen bei dem Skript dieses *GameObjects* gesetzt werden. Dabei handelt es sich um die Variablen *Application Signature* und *Tracking Data*.

Die *Application Signature* ist eine anwendungsspezifische Zeichenkette, welche nur als registrierter Benutzer des *Metaio Mobile SDK Developer Portals*[43] ermittelt werden kann. Für eine valide Signatur ist dabei der Paketname der Anwendung, welcher auch im *AndroidManifest* verwendet wurde, entscheidend. Dieser Paketname wird von Metaio auch als *Application Identifier* und von Unity als *Bundle Identifier* bezeichnet. Wird bei der Entwicklung dieser Paketname verändert, muss somit auch eine neue Signatur angefordert werden. Die einzelnen Schritte für dieses Vorgehen werden in [APPSIG] beschrieben.

Bei der Variablen *Tracking Data* handelt es sich um eine Zeichenkette die den Namen der XML-Konfigurationsdatei zum Tracking der Anwendung enthält. In dieser Zeichenkette wird der Name der XML-Datei zusammen mit der Dateiendung „.xml“ angegeben. Die korrespondierende Datei im Projekt muss allerdings mit der Dateiendung „.xml.bytes“ bezeichnet werden. Nicht nur bei der XML-Konfigurationsdatei, sondern bei allen Dateien des *3dmap*-Archivs des Metaio Creator Mobile müssen die Dateiendungen um das Suffix „.bytes“ für die Nutzung in Unity erweitert werden. Aufgrund der Implementierung der *metaioMobile*-Klasse müssen sich diese Dateien im Verzeichnis „Resources/metaio“ des Projekts befinden.

Unterhalb der Hierarchiestufe des *metaioMobile*-GameObjects befinden sich die drei verschiedenen *GameObjects CameraPreviewImage*, *MainCamera* und *PreviewCamera*. Die jeweils mit C#-Skripten verbunden sind. Der Zusammenhang zwischen den Objects und den Klassen ist nicht unmittelbar ersichtlich und wurde auch nicht von Metaio für die Öffentlichkeit dokumentiert. Entscheidend für die Darstellung ist das *MainCamera-GameObject*, welches unter anderem den Sichtbereich definiert. An dieser Stelle wurden die Werte der *Near* und *Far Clipping Plane* angepasst. Diese Werte bezeichnen den Abstand von der Kamera zum Ziel, der minimal und maximal erreicht werden darf, damit die virtuellen Elemente noch auf dem Display dargestellt werden. Des Weiteren wurden hier die Parameter der Kategorie *Normalized View Port Rect*, welche beschreiben an welcher Stelle des Displays das Kamerabild gezeichnet werden soll, an die Auflösung von 720x576 Pixel angepasst. [UNICAM]

Unterhalb des *GameObjects metaioMobile* befindet sich auf gleicher Stufe das *GameObject metaioTracker,* welches mit dem Skript *metaioTracker* versehen wurde und damit über eine globale

[43] Siehe https://mobiledeveloperportal.ar-live.de/

Variable *CosID* verfügt. Anhand dieser Variablen wird der Zusammenhang zwischen dem Tracking und den virtuellen Objekten geschaffen, die bei einer erfolgreichen Detektion des Ziels angezeigt werden sollen. Der *CosID*-Wert wurde auf den numerischen Wert eins gesetzt. Damit werden bei der Detektion des ersten Ziels der XML-Konfiguration, alle Kind-Elemente dieses *metaioTracker-GameObjects* auf dem Zielkoordinatensystem eingeblendet. Anhand des *CosID*-Wertes können somit in einem Szenario auch mehrere verschiedene Ziele getrackt und individuell virtuelle Objekte eingeblendet werden. Alle Kind-Elemente der aktuellen Szene beeinflussen die grafische Darstellung der Szene. Die einzige Ausnahme bildet das *GameObject ButtonAncor*, welches lediglich als Hüll-Objekt[44] für das *CallJava5*-Skript dient.

Im weiteren Verlauf der Masterarbeit wird auf den Arbeitsablauf in Verbindung mit Unity und Eclipse eingegangen und die Schritte beschrieben die notwendig sind, um ein Android-Plugin in Unity verwenden zu können. Der Versuch Inhalte aus einem Unity-Projekt in ein Eclipse Android-Projekt zu importieren schlug fehl. Der Grund für diesen Fehlschlag ist noch unklar, wird aber auf Seiten der Metaio-Komponenten und dessen Signatur-Mechanismus vermutet. Ein Grund für diese Annahme ist, dass ein Unity-Projekt ohne Einsatz des Metaio Mobile SDKs auf ein von Eclipse ausführbares Android-Projekt übertragen werden konnte. Ein Vorteil bei dem Import der Unity-Inhalte in ein Eclipse Android-Projekt wäre die Überwachbarkeit mittels DDMS zur Laufzeit gewesen. Mittels DDMS ist es beispielsweise möglich, die einzelnen Threads der Anwendung zur Laufzeit zu überwachen.

Der Import eines JAR-Archivs nach Unity konnte dagegen erfolgreich eingesetzt werden. Zu Debugging-Zwecken konnte hier ausschließlich die Debug-Ausgabe als Hilfsmittel verwendet werden. Aus diesem Grund werden durch den Quellcode fortlaufend Ausgaben erzeugt, die den Entwickler über den aktuellen Zustand der Anwendung informieren. Diese Ausgaben können nach dem Start des AR-Prototyps unter anderem durch die DDMS Ansicht in Eclipse in Form des LogCat-Systems[45] gefiltert und angezeigt werden. Weitere Informationen, wie zum Beispiel über laufende Threads, können allerdings nicht eingesehen werden. Auch bei entsprechender Deklaration im *AndroidManifest* anhand des Attributs „android:debuggable“[46] bleiben weitere Information durch den Build-Prozess von Unity verdeckt. Um das JAR-Archiv in Unity integrieren zu können, müssen im Unity-Projektverzeichnis einige Ordner angelegt werden, die durch Abbildung 31 in dem roten Kasten dargestellt werden.

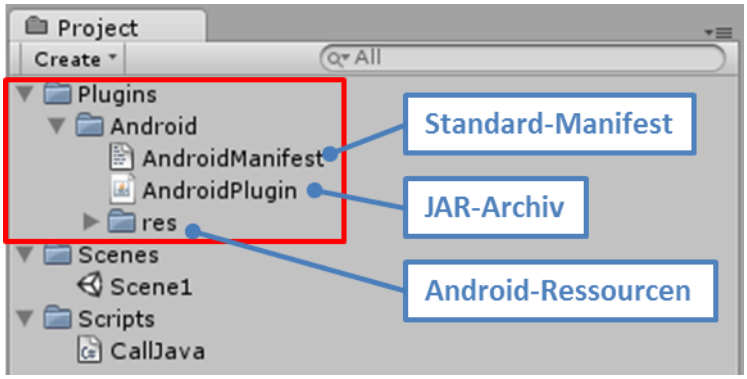


Abbildung 31 Verzeichnissturktur eines Unity-Projekts mit Android-Plugin (rot)

[44] Ein Hüll-Objekt verfügt über keine grafische Repräsentationsform in der Szene.
[45] Siehe http://developer.android.com/tools/help/logcat.html
[46] Siehe http://developer.android.com/guide/topics/manifest/application-element.html

Dabei handelt es sich um ein Beispiel einer Verzeichnisstruktur und nicht um die Struktur des AR-Prototyps. Zur Übersichtlichkeit wird ein Beispiel statt des AR-Prototyps zur Erläuterung dieser Struktur eingesetzt. Der AR-Prototyp ist allerdings analog zu diesem Beispiel aufgebaut. Einzig der Ordner „res“ kann aus dem Android-Projekt kopiert werden, welches erstellt wurde, um das Plugin in JAVA zu implementieren. Das Android-Projekt muss ein Paket enthalten, dessen Name in Unity anschließend als *Bundle Identifier* in den Build-Settings eingetragen wird. Die Funktionalität des Plugins muss in Klassen innerhalb dieses Paketes gekapselt werden. Um auf Methoden der Unity API im Kontext der Implementierung von JAVA Quellcode zugreifen zu können, muss ein JAR-Archiv aus Unity in das Android-Projekt eingebunden werden. Dabei handelt es sich um die Bibliothek *classes.jar*, die im Installationsverzeichnis von Unity zur Verfügung gestellt wird. Nachdem der Quellcode verfasst wurde, wird dieser in ein JAR-Archiv exportiert und in das Verzeichnis „Android“ des Unity-Projektverzeichnisses kopiert. Bei der *AndroidManifest*-Datei handelt es sich um eine Standard-Datei, die für verschiedene Unity-Projekte dieser Art eingesetzt werden kann. Lediglich die *Permissions*, bzw. Privilegien, der Anwendung müssen spezifisch festgelegt werden. In Kapitel 1.1.1 wurde diese Technik der Android-Plattform bereits erläutert. Die Standard-Manifest-Datei befindet sich auf der CD-ROM im Anhang im Verzeichnis „Unity_With_Plugin“. [UNIPLU]

Die JAR-Archive *JSON4J*, *BMWDemoModeLib* und *BMWMediator* wurden im entsprechenden Verzeichnis für Plugins abgelegt. Diese Archive und die in Unity eingesetzten C#-Skripte realisieren, abgesehen von der Funktionalität des Metaio Mobile SDKs, die Anwendungslogik des AR-Prototyps. In Abbildung 30 werden diese Dateien im Reiter *Projekt* des Unity-Projekts für den AR-Prototyp dargestellt.

Im nächsten Kapitel wird auf die verwendeten Hilfsmittel eingegangen, die im Verlauf der Masterarbeit eingesetzt wurden. Zusätzlich wird auf einige Testläufe im Zuge der Entwicklung und deren Ergebnisse hingewiesen. Dabei werden außerdem einige Missstände im Umgang mit der verwendeten Software beschrieben.

6 Verwendete Mittel zur Umsetzung der Masterarbeit

Im Zuge der Masterarbeit wurden verschiedene Programme und Geräte eingesetzt, um diverse Aufgaben durchführen zu können. Diese werden zusammen mit den relevanten Informationen in den folgenden Kapiteln aufgelistet. Dabei wird insbesondere auf die Tests bezüglich der Hardware in Verbindung mit dem AR-Prototyp eingegangen.

6.1 Software und Tools

Die Programme die für die Entwicklung des Demonstrators als auch des AR-Prototyps verwendet wurden, werden in der folgenden Aufzählung angegeben:

- Android SDK Tools, Revision: 19
- Eclipse Java EE DIE for Web Developers, Version: Indigo Service Release 2
    - Plug-Ins: Android Developer Tools (ADT), Version: 18
- Metaio Mobile SDK, Version 3.1.1
- Metaio Creator Mobile, Version 1.0.0
- Unity, Version 3.5.3f3
- Autodesk Maya 2013, Student Version
- Autodesk MotionBuilder 2013, Student Version

Bei dieser Aufzählung werden weder Software-Produkte zur Erstellung des Textes oder der Abbildungen noch zur Recherche für die Masterthesis abgedeckt. Die Art und Weise der Verwendung der jeweiligen Software wurde bereits an den entsprechenden Stellen der vorherigen Kapitel genannt. Der Vollständigkeit halber wird darauf hingewiesen, dass als Betriebssystem auf dem Entwicklungsrechner zur Masterarbeit MS Windows XP eingesetzt wurde.

6.2 Hardware und Tests

Um den Demonstrator als auch den AR-Prototyp betreiben zu können, war ein geeignetes mobiles Endgerät notwendig. Aufgrund von Schwierigkeiten bei der Darstellung der virtuellen Objekte des Prototyps wurden verschiedene Tests auf unterschiedlichen Geräten durchgeführt. Auch wurden Tests bezüglich des Demonstrators durchgeführt, da keine drei Tablets vom gleichen Typ für den BMW-Familientag zur Verfügung gestellt werden konnten.

In den folgenden Aufzählungen werden die Tablets und Smartphones aufgeführt, die bereits mit den Anwendungen getestet wurden.

Tablets:

- ASUS Eee Pad Transformer Prime TF201, Android 4.0.3
- ASUS Transformer Pad TF300T, Android 4.0.3
- ASUS Transformer Pad Infinity TF700T , Android 4.0.3

Smartphones:

- Samsung Galaxy S GT-I9000, Android 2.3.3
- Samsung Galaxy SII GT-I9100, Android 2.3.3
- Samsung Galaxy SIII GT-I9300 , Android 4.0.4

Dabei wurden nicht alle Geräte für die gleichen Tests eingesetzt. Die Tablets wurden für Tests bezüglich der Kommunikation zwischen Client und Server eingesetzt, um die WLAN Übertragung für den BMW-Familientag abzusichern. Bereits in Kapitel 4.2.1 wurden die Tests bezüglich der Kommunikation mittels WLAN angesprochen. Deshalb wird an dieser Stelle nicht weiter auf diese Tests eingegangen. Lediglich Tests des AR-Prototyps in Bezug auf die unterschiedlichen Geräte, bzw. Hardware, werden in diesem Kapitel beschrieben.

Durch Tests bezüglich der Darstellung von Inhalten des AR-Prototyps konnte eine Diskrepanz zwischen Tablets und Smartphones festgestellt werden. Gleiche Ergebnisse erzielten Geräte der gleichen Kategorie. Dabei wurde festgestellt, dass die Tablets der Firma ASUS die Inhalte des AR-Prototyps nicht korrekt darstellen können. In Abbildung 32 wird dieser Unterschied am Beispiel des ASUS Transformer Pad TF300 und des SAMSUNG Galaxy S GT-I9000 gezeigt.

Abbildung 32 Darstellung des AR-Prototyp, ASUS Tablet (links) SAMSUNG Smartphone (rechts)

Durch rote Kreise werden auf den beiden Bildern die zu vergleichenden Bereiche hervorgehoben. Bei dem linken Bild, welches von dem ASUS-Tablet erstellt wurde, werden die Texturen auf dem Kopf des Charakters als auch auf dem Goldbarren nicht korrekt dargestellt. Im Gegensatz dazu werden im rechten Bild, welches ein Abbild des AR-Prototyps mittels des SAMSUNG Smartphones darstellt, diese Bereiche korrekt dargestellt.

Auch für die Erstellung von *3dmap*-Archiven, bzw. Referenzmodellen für das Markerless 3D Tracking, durch die Creator Mobile App der Firma Metaio konnten die Tablets der Firma ASUS nicht erfolgreich eingesetzt werden, was bereits in Kapitel 5.1.2 angesprochen wurde. Wie aus den Foren der Firma Metaio zu entnehmen ist, wurden bereits ähnliche Schwierigkeiten mit anderen Tablet-Geräten festgestellt. Angeblich wird allerdings an der Unterstützung von Tablet-Geräten gearbeitet. Für den Einsatz der Software auf Tablet-Geräten wurde auf spätere Releases von Metaio-Mitarbeitern verwiesen.

Unabhängig von der Anwendung des Markerless 3D Trackings konnte bei einigen Tests festgestellt werden, dass bei einem ID Marker Tracking mittels des Metaio Mobile SDK ein Sortierungsproblem bei der Darstellung der virtuellen Bestandteile auftritt. In Abbildung 33 wird dies anhand einer stark vereinfachten Form einer Fahrzeughinterachse als dreidimensionales Objekt sichtbar. Für die Ansicht des AR-Prototyps wurde ein ASUS Transformer Pad TF300T und ein Samsung Galaxy SIII GT-I9300 eingesetzt. Auf der linken Seite der Abbildung wird der Sortierungsfehler angezeigt, der sich in einer invertierten Reihenfolge der Objektdarstellung bezüglich der räumlichen Tiefe äußert. Die korrekte Darstellung bei der vorgegebenen Perspektive der Kamera wird auf der rechten Seite abgebildet.

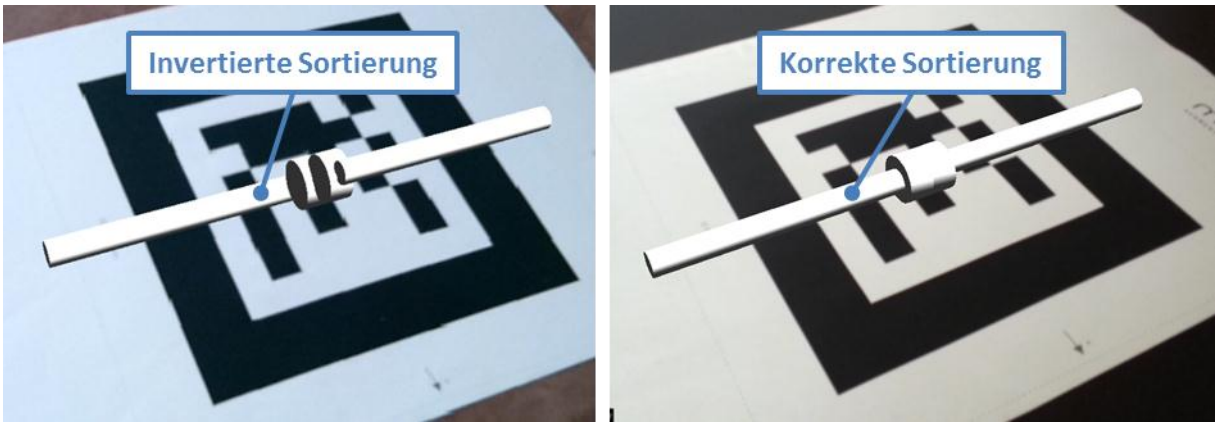


Abbildung 33 Sortierungsproblem, Ansicht ASUS-Tablet (links) und Ansicht SAMSUNG Smartphone (rechts)

Auf die Fehlerquellen bei der Darstellung von Inhalten im Umgang mit einem Tablet-Gerät werden in dieser Masterthesis nicht weiter eingegangen. Unter Umständen basieren die beiden Darstellungsfehler sogar auf der gleichen Ursache.

Im folgenden Kapitel wird das Thema hinsichtlich der einzelnen Arbeitsschritte zusammengefasst und Weiterführungen des Projekts als auch Verbesserungsvorschläge für konsekutive Abschlussarbeiten beschrieben. Des Weiteren wird die Technologie Augmented Reality im Zusammenhang mit zukünftigen Entwicklungen der Hardware- als auch Software-Seite bezüglich der Anwendung auf mobilen Endgeräten diskutiert.

7 Zusammenfassung und Ausblick

Für den BMW-Familientag konnte ein Demonstrator entwickelt werden, der zur Validierung der Kommunikation zwischen Tablet-Anwendung und Fahrzeug-Aktuatorik eingesetzt werden konnte. Die entstandene Software wurde teilweise im Kontext einer AR-Anwendung zur Erklärung von Steuergerätefunktionen eingesetzt. Anhand von unterschiedlichen mobilen Endgeräten wurde auf die Nutzbarkeit von AR-Anwendungen durch eine bestimmte Werkzeugsammlung eingegangen und auf Probleme, bzw. Missstände, hingewiesen. Zusätzlich konnten verschiedene Technologien im Bereich der AR in Bezug auf das Metaio Mobile SDK erläutert und auf dessen Vor- und Nachteile eingegangen werden.

Besonders profitabel beim Selbststudium der Benutzung verschiedener Programme, wie zum Beispiel zur Animation von Charakteren mittels der Autodesk-Produkte, als auch der theoretischen Hintergründe bezüglich Animation und Augmented Reality, waren die Dokumentationen und Videotutorien im Internet. Besonders im Umgang mit Unity konnte bezüglich der Entwicklung von virtuellen Inhalten komfortabel und ressourcensparend gearbeitet werden.

Zur Verbesserung des eingesetzten Trackings in Bezug auf den AR-Prototyp bleibt zu ermitteln, welche Technologie für den Einsatz am Automobil am geeignetsten ist. Dabei sollten vorher die Kriterien festgelegt werden, die für eine entsprechende Anwendung entscheidend sind. Da bei einer solchen Anwendung von AR nicht mit sicherheitskritischen Effekten bezüglich der Genauigkeit des getrackten Ziels zu rechnen ist, sind eher Robustheit und Geschwindigkeit die ausschlaggebenden Kriterien. Welche Kriterien priorisiert werden sollten hängt allerdings vom jeweiligen Anwendungsfall ab. Aufgrund der Größe der einzelnen mechanischen Bestandteile eines Fahrzeugs könnte die Positionsbestimmung der darzustellenden Komponenten im Zentimeterbereich bereits ausreichen. Ein Beispiel dafür wäre die Darstellung der Mechanik als Überblendung auf dem Fahrzeug. Durch Interaktion mit dem Benutzer wird die Aktuatorik manipuliert und gleichzeitig entsprechende Veränderungen anhand der virtuellen Mechanik eingeblendet. Zusätzlich ist das Kriterium der Genauigkeit natürlich vom Abstand, bzw. dem Bereich, des Betrachters zum Ziel-Objekt abhängig. Durch die eingeschränkte Rechenleistung mobiler Endgeräte muss darüber hinaus ermittelt werden, welche Verfahren performant durchgeführt werden können. In [FRIEMT] wurde bereits der Einsatz des PSA Algorithmus in diesem Zusammenhang untersucht, der unter Umständen in ähnlicher Form von Metaio im Markerless 3D Tracking eingesetzt wird und explizit für mobile Endgeräte entwickelt wurde. Auch die Größe des Ziel-Objekts, dass anhand des Referenzmarkers, bzw. Ankers, als Referenzmodell verwendet wird, muss dabei berücksichtig werden. Grundsätzlich ist ein Ansatz mittels PSA, wie in [ISMAR9] beschrieben, eher für kleine bis mittelgroße Objekte geeignet. Somit ist für eine mobile Anwendung dieses Verfahren möglicherweise in Bezug auf ein reales Automobil als Referenzmodell ungeeignet. Aus diesem Grund sollte zusätzlich über die Verwendung von mehreren Markern in einem Modell diskutiert werden. Gerade wegen den reflektierenden Oberflächen eines Automobils sollten eventuell Merkmale aus nicht reflektierendem Material am Fahrzeug oder in der unmittelbaren Fahrzeugumgebung angebracht werden. Aufgrund der Tatsache, dass die Kommunikation via WLAN bereits für die Manipulation der Aktuatorik eingesetzt wird, kann ein verteilter Ansatz mit ausgelagerten Rechenprozessen aus weiteren Betrachtungen ausgeschlossen werden. Allerdings verfügen bereits heutige Tablet-Rechner, wie beispielsweise das ASUS Transformer Pad TF300T, über einen Vierkern-Prozessor und einen Arbeitsspeicher von einem Gigabyte Speicherplatz. Aus diesem Grund können in Zukunft wohl auch rechenintensivere und speicheraufwendigere Verfahren auf mobilen Endgeräten ausgeführt werden. Der in dem ASUS-

Tablet befindliche Prozessor der Firma NIVIDIA verfügt bereits laut [NVIDIA] über eine Leistung von 309 MFLOPS, die anhand des multi-threaded LINPACK Benchmarks gemessen wurde. Als Vergleichswerte werden hier einige Messwerte angegeben. Beispielsweise erreichte der SuperMUC des Leibniz-Rechenzentrums in Garching bei München bei einer Anzahl von 147456 Rechenkernen einen maximalen Punktestand 2897 TFLOPS im LINPACK Benchmark und steht damit auf dem vierten Platz der Supercomputer weltweit bezüglich der Rechenleistung. [TOP500]

Die Rechenleistung dieses Supercomputers dient hier nur als Rahmen und Obergrenze für weitere Angaben. Bei einem Intel Core 2 Quad Q8200 mit einer Taktung von 2,33 GHz wurden durchschnittlich im LINPACK Benchmark der Firma Intel 23 GFLOPS gemessen. Anhand einer Implementierung des LINPACK Benchmarks für Android konnten auch eigene Tests an verschiedenen mobilen Geräten durchgeführt werden. Dabei erreichten das SAMSUNG Galaxy SIII und das ASUS Transformer Pad TF300T jeweils Werte von circa 100 MFLOPS im multi-threaded Benchmark. Dagegen erreicht das SAMSUNG Galaxy SII mit Dual-Core Prozessor im gleichen Test circa 80 MFLOPS und das SAMSUNG Galaxy SI mit dem Singlecore Prozessor circa 15 MFLOPS. Anhand dieser Entwicklung und der Ankündigungen durch NVIDIA auf deren offiziellen Blog kann man davon ausgehen, dass in wenigen Jahren heutige Algorithmen für stationäre, bzw. leistungsstärkere, Rechner auch auf mobilen Endgeräten bearbeitet werden können. In Abbildung 34 werden die Ziele von NVIDIA bezüglich der Performanz ihrer Prozessoren bis zum Jahr 2014 dargestellt. Dabei soll Kal-El, bzw. der 1 GHz Quad-Core-ARM-SoC von NVIDIA[47] bereits die fünffache Rechenleistung gegenüber dem Tegra 2 zur Verfügung stellen. [NVBLOG]

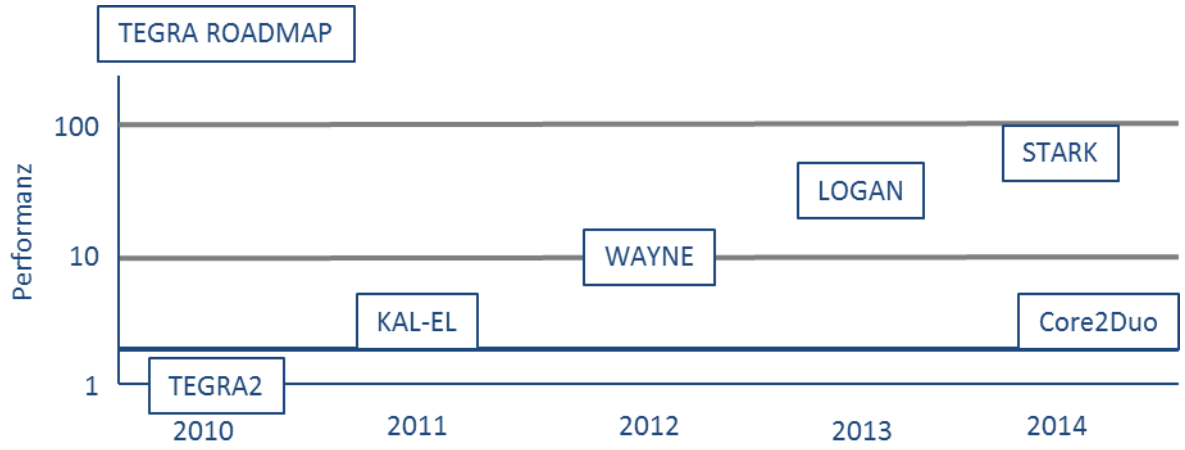


Abbildung 34 Ziele von NVIDIA bezüglich der Performanz ihrer Prozessoren in den nächsten Jahren [NVBLOG]

Bis weitere Verfahren und Algorithmen für mobile Android-Geräte verfügbar sind, kann der AR-Prototyp noch um einige Anwendungsfälle erweitert werden. Eine Variante zur Erweiterung wäre die Benutzung des Prototyps zeitgleich durch mehrere Beobachter. Beispielsweise kann ein passiver Betriebsmodus in den Prototyp integriert werden, der lediglich zur Anzeige dient, jedoch über keine Möglichkeit zur Manipulation der Aktuatorik verfügt. Somit steuert ein einzelner Benutzer den Präsentationsverlauf, dem mehrere Beobachter aus wechselnden Perspektiven folgen können. Zusätzlich sollten relevante Bestandteile der Mechanik als Überblendung auf dem Fahrzeug angezeigt werden. Dabei sollte es dem Benutzer durch Interaktion mit dem Tablet, egal ob im Beobachtungsmodus oder Steuerungsmodus ermöglicht werden, Informationen während der Präsentation zu einzelnen Bereichen zu erfragen. Ein Beispiel dafür wäre das Abfragen technischer

[47] Dieser Chipsatz wird von NVIDIA auch als Tegra 3 bezeichnet.

Informationen von einzelnen Bestandteilen der Überblendung anhand einer Berührung des Touch-Displays. Somit hätte man die Möglichkeit eines virtuellen Exponats, das darüber hinaus den Vorteil hat, keinen Raum für die Lagerung einzunehmen, leicht transportierbar zu sein und vermutlich weniger Kosten verursacht als ein reales Exponat. Als Kosten fallen einmalig die Tablets und die Entwicklung der Software an. Verschiedene Modelle können dann aus den CAD-Daten der Entwicklung generiert werden. Die Technische Universität Graz forscht in verschiedenen AR-Projekten an Einsatzmöglichkeiten am Automobil. Dabei werden unter anderem Technologien eingesetzt, die für eine Weiterentwicklung des AR-Prototyps relevant sein könnten. In Abbildung 35 werden einige dieser Technologien dargestellt, auf die im Zuge dieser Arbeit nicht weiter eingegangen wird. [TUGRAZ]

Abbildung 35 AR-Anwendungen der TU Graz, Cutaways (links) Labeling (mitte) Focus and Context (rechts) [TUGRAZ]

Auch die Verwendung der bisher entwickelten Umfänge sollte für eine Anwendung auf anderen Plattformen möglich sein. Zum Beispiel müsste für eine Adaption der AR-Inhalte des AR-Prototypen an eine iOS App lediglich die Schnittstelle in Unity, bzw. die C#-Skripte, und der Kommunikationsanteil, der im *CarMediator*-Archiv zusammengefasst wurde, für iOS neu implementiert werden. Somit lässt sich unter Verwendung von Unity der AR-Prototyp zumindest teilweise für die Portierung auf andere Plattformen nutzen. Bei entsprechenden Lizenzen und einem Apple-Rechner kann Unity zum Erstellen der iOS-Variante dieser App eingesetzt werden. An dieser Stelle wird für genauere Informationen auf die Dokumentation von Unity verwiesen. [UNIIOS]

In Abbildung 36 wird ein Graph bezüglich des Potentials und der Erwartungen an verschiedene Technologien dargestellt. Durch einen roten Pfeil und einen roten Kasten wurde die Augmented Reality Technologie hervorgehoben. Am Graphen wird entlang der Ordinatenachse die Erwartungshaltung angetragen, während die Abszissenachse den zeitlichen Verlauf kennzeichnet. Die Augmented Reality Technologie benötigt somit laut dieses Graphen noch circa fünf bis zehn Jahre bis es bezüglich der Erwartungen auf einem produktiven, bzw. realistischem Niveau angelangt ist.

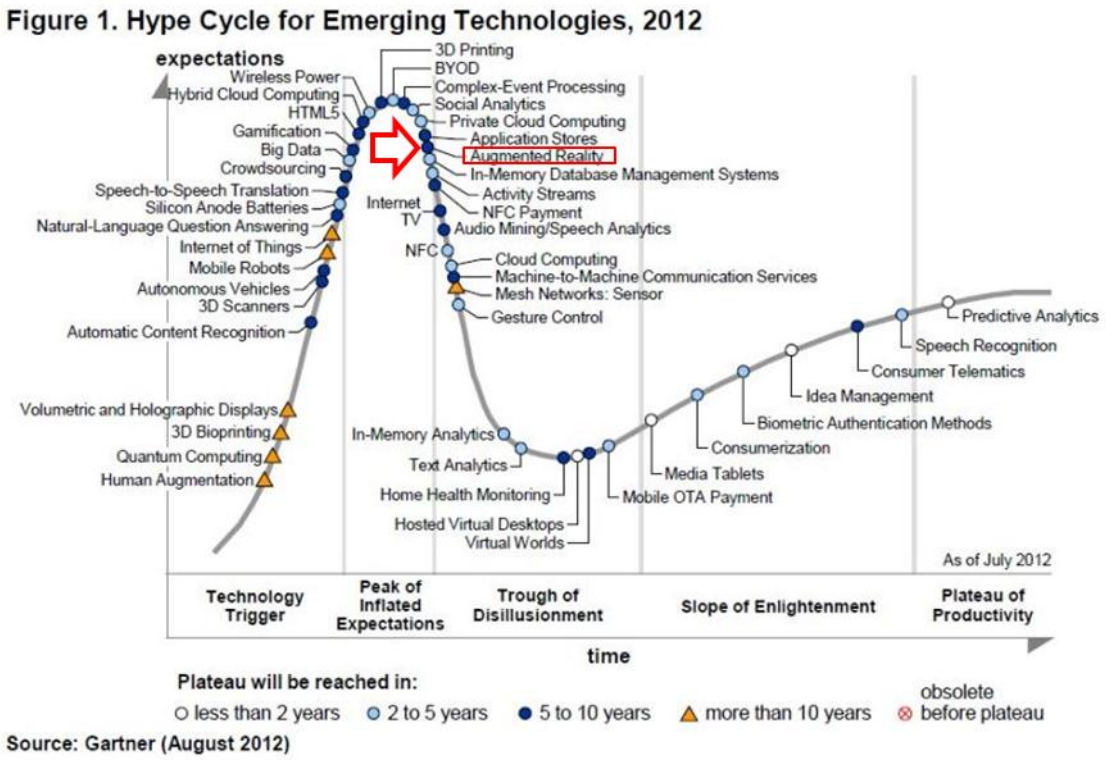


Abbildung 36 Prognose der aufkommenden Technologien der Firma Gartner, Inc. [GARHAR]

Anhand dieser Prognose des Marktforschungs- und Beratungsunternehmens Gartner, Inc. zum Thema Hype um die aufkommenden Technologien vom Jahr 2012 kann man davon ausgehen, dass Augmented Reality und dessen Anwendungen in Zukunft ein anderes Verständnis bezüglich der Umsetzbarkeit, bzw. dem Potential, von AR-Anwendungen auch auf mobilen Endgeräten erfahren. Trotz allem ist Augmented Reality eine Technologie mit hohem Potential und innovativem Charakter, welche mit Sicherheit in zukünftigen Entwicklungen gewinnbringend zur Erklärung von Steuergerätefunktionen im Automobilbereich eingesetzt werden kann.

Abkürzungsverzeichnis

AR	Augmented Reality
ARS	Aktive Rollstabilisierung (Dynamic Drive)
AV	Augmented Virtuality
CAD	Computer-Aided Design
CSV	Comma-seperated values (Dateiformat)
DDMS	Dalvik Debug Monitor Server
F11	Baureihenbezeichnung des BMW 5er Touring seit 2010
FIZ	Forschungs- und Innovationszentrum
FLOPS	Floating Point Operations Per Second
GFLOPS	GigaFLOPS (=10^3 FLOPS)
GPS	Global Positioning System
GUI	Graphical User Interface (Grafische Benutzerschnittstelle)
HMD	Head-mounted display
HUD	Head Up Display
IDC	International Data Corporation
IDE	Integrierte Entwicklungsumgebung
IIR	Infinite impulse response (unendliche Impulsantwort)
IP	Internet Protocol (hier: IPv4, Internet Protocol Version 4)
IR	Infrarot
JNI	Java Native Interface
JSON	JavaScript Object Notation
KLT	Kanade-Lucas-Tomasi
LoC	Lines of Code (Quellcodezeilen)
MB	Maya Binary (Dateiformat)
MFLOPS	MegaFLOPS (= 10^6 FLOPS)
MR	Mixed Reality
POJO	Plain Old Java Object
PSA	Projection Shift Analysis
PTAM	Parallel Tracking and Mapping
RV	Reality-Virtuality
SfM	Structure from Motion
SIFT	Scale-Invariant Feature Transform

SoC	System-on-a-Chip
ST	see-through AR
SURF	Speeded Up Robust Features
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TFLOPS	TeraFLOPS (= 10^{12} FLOPS)
TOS	Type-of-Service
UDP	User Datagram Protocol
UML	Unified Modeling Language
WLAN	Wireless Local Area Network
WoW	Window-on-the-world
XML	Extensible Markup Language

Glossar

Der Glossar dient zur Erklärung einzelner Begriffe die in der Masterthesis verwendet wurden. Dabei werden Begriffe durch diesen Glossar bezüglich der Verwendung in der Masterthesis definiert.

Activity[48]	Bezeichnet im Umgang mit Android eine Klasse, welche vom Benutzer durch ein User-Interface wahrgenommen werden kann und Mittel zur Interaktion mit dem Benutzer bereitstellt.
Aktuatorik	Ein System, das durch aktives Eingreifen in die Umgebung Bewegungen, bzw. Veränderungen in der Umgebung erzeugen kann
Black-Box	Ist ein Konzept welches verdeutlicht, das lediglich Ein- und Ausgabe bei einer Informationsverarbeitung bekannt sind. Der Vorgang um von der Eingabe auf die Ausgabe schließen zu können ist verdeckt und nicht ersichtlich.
Branding Features	Dieser Begriff umfasst Merkmale, die ein Produkt mit einer Firmenzugehörigkeit ausweisen und dabei somit beispielsweise einen Markennamen tragen.
Broadcast	Ist die Nachrichtenübertragung von einem Sender an alle Empfänger in einem Netzwerk
Character-Rig	Ist das Skelett eines dreidimensionalen Modells, welches zur Animation dieses Modells verwendet werden kann.
Charakter	Beschreibt ein dreidimensionales menschenähnliches Modell, das über eine Oberfläche, eine Texturierung und Animationen verfügt.
Feature	optisch relativ eindeutig zu unterscheidendes Merkmal in Form eines Bildpunktes oder eines Bildbereichs
KeepAlive time	Die maximale Zeit, welche paarweise zwischen zwei Nachrichten liegen darf damit sich die Verbindung in einem aktiven Zustand befindet. Hier bedeutet der aktive Zustand, dass beide Verbindungspartner vom jeweils anderen erreichbar sind.
Marker	Ein künstliches optisches Hilfsmittel um die Detektion, bzw. das Wiedererkennen eines Ziels, zu unterstützen (bei optischen Verfahren zur Detektion werden häufig ID-Marker verwendet)
Multicast	Ist die gleichzeitige Nachrichtenübertragung von einem Sender an mehrere Empfänger, bzw. eine Empfängergruppe
Multithreading	Die Verwendung mehrerer leichtgewichtiger Prozesse(Threads) zur nebenläufigen Programmverarbeitung.
Physik-Engine	Ein System, dass es erlaubt virtuelle Objekte mit nahezu physikalisch korrektem Verhalten in der simulierten Umgebung zu versehen.
Piconetz	Der Begriff Piconetz wird im Zusammenhang mit der Bluetooth-Technologie als ad-hoc-Netzwerk von maximal acht Teilnehmern bezeichnet. [ITSECB]

[48] Siehe http://developer.android.com/reference/android/app/Activity.html

POJO	Ein Objekt der JAVA Programmiersprache, welches nur rudimentär über ein eigenes Verhalten verfügt. Grundsätzlich wird ein solches Objekt als Container für Informationen ohne spezifisches Verhalten angesehen.
Pose	Entspricht der Position und Orientierung eines Objektes im dreidimensionalen Raum durch die Angabe aller sechs Freiheitsgrade bezüglich Translation und Rotation
Rechtzeitigkeit	Die Rechtzeitigkeit ist ein Begriff der Echtzeitverarbeitung und wird durch Antwortzeitverhalten, zeitliche Gültigkeit der Informationen und Hochlastfestigkeit gekennzeichnet. [INFOHB]
Sensorik	Ein System, welches die in einer Umgebung vorherrschenden physikalischen Größen erfassen kann
Unicast	Bezeichnet die exklusive Nachrichtenübertragung zwischen zwei festgelegten fixen Parteien
Walk-Cycle	Bezeichnet den zyklischen Bewegungsablauf beim Laufen anhand einer Charakter-Animation.
Wankbewegung	Die Rollbewegung entlang der Längsachse des Fahrzeugs
Wankrantenverteilung	Die Verteilung des Konterwankmoments auf Vorder-, bzw. Hinterachse

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[APPSIG] Internetseite, http://docs.metaio.com/bin/view/Main/ApplicationSignature, vom 04.10.2012

[AR3DML] Lima, Joao Paulo S. do M.; Simoes, Francisco P. M.; Figueiredo, Lucas S.; Teichrieb, Veronica; Kelner, Judith; Santos, Ismael H. F. *Model Based 3D Tracking Techniques for Markerless Augmented Reality*, Paper, Universität von Pernambuco (UFPE), 2009, Seiten 39-43, https://www.gprt.ufpe.br/grvm/Publication/FullPapers/2009/SVR2009_Limaetal.pdf

[ARCLAB] Ludwig, Christine; Reimann, Christian *Augmented Reality: Informationen im Fokus*, Report, Siemens Business Services GmbH & Co. OHG und Universität Paderborn, 2005

[ARGFTT] Shi, Jianbo; Tomasi, Carlo *Good Features to Track*, Paper, IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, Juni 1994, http://movement.nyu.edu/mocap11f/papers/lec03_OpenCV_FeaturesFinding.pdf

[ARINDU] Georgel, Pierre; Schroeder, Pierre; Appel, Mirko; Benhimane, Selim; Navab, Nassir; Hinterstoisser, Stefan *An Industrial Augmented Reality Solution For Discrepancy Check*, Paper, Technische Universität München, 2007, http://ar.in.tum.de/pub/georgel2007ismar/georgel2007ismar.pdf

[ARMTSW] Lima, Joao Paulo S. do M.; Pinheiro, Pablo C.; Teichrieb, Veronica; Kelner, Judith *Markerless Tracking Solutions for Augmented Reality on the Web*, Paper, Universität von Pernambuco (UFPE), 2010, Seiten 2-4, https://www.gprt.ufpe.br/grvm/Publication/FullPapers/2010/SVR2010_Limaetal.pdf

[ARNASA] Reisman, Ronald J.; Brown, David M. *Design of Augmented Reality Tools for Air Traffic Control Towers*, Paper, 6th AIAA Aviation Technology Integration and Operation Conference (ATIO), September 2006, http://www.aviationsystemsdivision.arc.nasa.gov/publications/more/other/reisman_09_06.pdf

[ARPTAM] Klein, Georg; Murray, David *Parallel Tracking and Mapping for Small AR Workspaces*, Paper, Universität von Oxford, 2007

[ARRVC] Milgram, Paul *Augmented Reality: A class of displays on the reality-virtuality continuum*, Paper, SPIE Vol. 2351, Telemanipulator and Telepresence Technologies, 1994

[ARTOOL] Internetseite, http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm, vom 08.10.2012

[ARVIKA] Internetseite, http://www.arvika.de/www/d/topic1/einfuehrung.htm, vom 05.09.2012

[ARVIKAb] Friedrich, Wolfgang *ARVIKA Augmented Reality für Entwicklung, Produktion und Service*, Siemens AG, http://www.arvika.de/www/pdf/IVIP_Forum_d.pdf

[AZUMA] Azuma, Ronald T. *A Survey of Augmented Reality*, Paper, Teleoperators and Virtual Environments 6, August 1997, Seiten 355-385, http://www.cs.unc.edu/~azuma/ARpresence.pdf

[BLUETO] Internetseite, https://www.bluetooth.org/Building/overview.htm, vom 11.09.2012

[BMWARS] Lenthaparambil, Nino; Weber, Ingo; Kecskeméthy, Andrés *Aktive Wankstabilisierung mit einem aktiven Stabilisator*, Vortragsfolien zu VDI Mechatronik 2009 in Wiesloch, 2009

[BMWRTD] Trampler, Robert *Fahrzeugzugang für Demo-Mode-Fahrdynamik*, Diplomarbeit, Hochschule Augsburg (University of Applied Sciences), Juli 2012

[BMWSV1] Internetseite, http://www.bmw.com/com/de/owners/service/augmented_reality_introduction_2.html, vom 08.10.2012

[BMWTL1] Internetseite, http://www.bmw.de/de/de/insights/technology/technology_guide/articles/mm_active_steering.html?source=index&article=mm_active_steering, vom 06.07.2012

[BMWTL2] Internetseite, http://www.bmw.de/de/de/insights/technology/technology_guide/articles/dynamic_drive.html?source=index&article=dynamic_drive, vom 10.07.2012

[BMWTL3] Internetseite, http://www.bmw.de/de/de/insights/technology/technology_guide/articles/adaptive_drive.html?source=index&article=adaptive_drive, vom 10.07.2012

[BMWTL4] Internetseite, http://www.bmw.de/de/de/insights/technology/technology_guide/articles/self_levelling.html, vom 05.09.2012

[BMWUSA] Internetseite, http://www.bmwusa.com/Standard/Content/Owner/dgh.aspx, vom 07.09.2012

[CAMP] Internetseite, http://campar.in.tum.de/Chair/AugmentedReality, vom 06.07.2012

[CREAFAQ] Internetseite, http://docs.metaio.com/bin/view/Main/MetaioCreatorMobileFAQ, vom 30.09.2012

[DALVIK] Internetseite, http://developer.android.com/guide/appendix/glossary.html, vom 08.10.2012

[DATAMI] Runkler, Thomas A. *Data Mining – Methoden und Algorithmen intelligenter Datenanalyse*, 1.Auflage, Vieweg+Teubner, 2010, ISBN 978-3-8348-0858-5, Seiten 26 ff.

[DEVAND] Internetseite, http://developer.android.com/about/dashboards/index.html, vom 08.10.2012

[FLXRAY] Internetseite, http://www.flexray.com/, vom 10.10.2012

[FOWLER] Internetseite, http://www.martinfowler.com/bliki/POJO.html, vom 19.09.2012

[FRIEMT] Pankratz, Frieder *Tracking für Augmented Reality Anwendungen auf Mobiltelefonen*, Masterarbeit, Technische Universität München, 15 Juli 2009, Seiten 18 ff.

[FRMADER] Mader, Franz *Entwurf und Integration eines kamerabasierten Trackingsystems für ein Flugzeugcockpit zur Darstellung fortschrittlicher Flugführungsinformationen in einem Head-Mounted Display*, Diplomarbeit, Technische Universität München, Juli 2004

[FTSURF] Bay, Herbert; Tuytelaars, Tinne; Van Gool, Luc *SURF: Speeded Up Robust Features*, Paper, ETH Zürich, http://www.vision.ee.ethz.ch/~surf/eccv06.pdf

[GAMMADP] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John *Design Patterns, Elements of Reusable Object-Oriented Software*, 38.Auflage, Addison-Wesley professional computing series, 2010, ISBN 978-0-201-63361-0, Seiten 127 ff., 273 ff., 293 ff.

[GARHAR] Internetseite, http://www.gartner.com/it/page.jsp?id=2124315, vom 07.10.2012

[GARTGA] Internetseite, http://www.gartner.com/it/page.jsp?id=1370213, vom 10.09.2012

[GARTTA] Internetseite, http://www.gartner.com/it/page.jsp?id=1570714, vom 10.09.2012

[GARTWS] Internetseite, http://www.gartner.com/it/page.jsp?id=2120015, vom 10.09.2012

[GOBLOG] Internetseite, http://googleblog.blogspot.de/2007/11/wheres-my-gphone.html, vom 08.10.2012

[GUFAR1] Vorlesungsfolien, http://www.gdv.informatik.uni-frankfurt.de/lehre/ss2002/Folien/VR&MR/vrmr_02_anwendungen.pdf, vom 11.07.2012

[HORNET1] Internetseite, http://www.horizont.net/aktuell/marketing/pages/protected/BMW-testet-Augmented-Reality-in-Printanzeigen_104327.html, vom 11.07.2012

[IDCIWS] Internetseite, http://www.idc.com/getdoc.jsp?containerId=prUS23638712, vom 10.09.2012

[IEEESP] Charette, Robert N. *This Car Runs on Code*, Artikel der Zeitschrift IEEE Spektrum, Februar 2009, http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code

[IMUKLT] Internetseite, http://www.ri.cmu.edu/research_project_detail.html?project_id=666&menu_id=261, vom 16.09.2012

[INFOHB] Pomberger, Gustav; Rechenberg, Peter *Informatik-Handbuch*, 4.Auflage, Carl Hanser Verlag GmbH & CO. KG, 2006, ISBN 978-3-446-40185-3, Seiten 740 ff.,

[INJSON] Internetseite, http://www.json.org/, vom 05.09.2012

[ISMAR9] Keitler, Peter; Pankratz, Frieder; Schwerdtfeger, Björn; Rödiger, Wolf; Klinker, Gudrun; Rauch,Christian; Chathoth, Anup; Collomosse, John; Song, Yi-Zhe *Mobile Augmented Reality based 3D Snapshot*, Sechster Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR, Braunschweig, November 2009

[ITSECA] Eckert, Claudia; Folien zur Vorlesung „*Sichere mobile Systeme*“ SS11, Technische Universität München, August 2011, http://www.sec.in.tum.de/sichere-mobile-systeme-ss11/

[ITSECB] Eckert, Claudia *IT-Sicherheit, Konzepte –Verfahren – Protokolle*, 6.Auflage, Oldenbourg Wissenschaftsverlag GmbH, München, 2009, ISBN 978-3-486-58999-3, Seiten 911 ff.

[JAVANE] Internetseite, http://docs.oracle.com/javase/1.5.0/docs/api/java/net/package-summary.html, vom 11.09.2012

[JSON4JDO] Internetseite, http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.json.help%2Fjavadoc%2Findex.html, vom 13.09.2012

[KLTSIFT] Sinha, Sudipta N.; Frahm, Jan-Michael; Pollefeys, Marc; Genc, Yakup *Feature Tracking and Matching in Video Using Programmable Graphics Hardware*, Paper, Universität von North Carolina bei Chapel Hill, http://www.cs.unc.edu/Research/urbanscape/public/Sinha_FeatureTrackingGPU_MVA07.pdf

[METAIO] Internetseite, http://www.metaio.com/fileadmin/upload/documents/pdf/LCNS-metaio_mobile_sdk_3_1_2012-2.pdf, vom 29.09.2012

[METUNI] Internetseite, http://docs.metaio.com/bin/view/Main/MobileGettingStartedUnity, vom 29.09.2012

[MLESCO] Internetseite, http://docs.metaio.com/bin/view/Main/UnifeyeMobileMarkerlessConfiguration, vom 29.09.2012

[NAGLEA] Internetseite, http://tools.ietf.org/pdf/rfc896.pdf, vom 18.09.2012

[NVBLOG] Internetseite, http://blogs.nvidia.com/2011/02/tegra-roadmap-revealed-next-chip-worlds-first-quadcore-mobile-processor/, vom 07.10.2012

[NVIDIA] Internetseite, http://www.nvidia.de/content/PDF/tegra_white_papers/tegra-whitepaper-0911a.pdf, vom 06.10.2012

[OHSALL] Internetseite, http://www.openhandsetalliance.com/index.html, vom 08.10.2012

[OHSFAQ] Internetseite, http://www.openhandsetalliance.com/oha_faq.html, vom 08.10.2012

[TOP500] Internetseite, http://www.top500.org/list/2012/06/100, vom 06.10.2012

[TRACON] Internetseite, http://docs.metaio.com/bin/view/Main/UnifeyeMobileTrackingConfiguration, vom 29.09.2012

[TUGRAZ] Internetseite, http://www.icg.tugraz.at/project/visualar/technology, vom 07.10.2012

[TUMAR1] Internetseite, http://campar.in.tum.de/Chair/ProjectChemistry, vom 11.07.2012

[TUMAR2] Internetseite, http://campar.in.tum.de/Chair/ProjectMedUI, vom 11.07.2012

[TUMSWA] Bergner, Klaus; Broy, Manfred; Sihling, Marc Folien zur Vorlesung „*Softwarearchitektur*“ WS10/11, Technische Universität München, Oktober 2010, http://www4.in.tum.de/lehre/vorlesungen/sw_arch/WS1011/

[TUMSWT] Broy, Manfred; Ehler, Herbert; Schätz, Bernhard Skript zur Vorlesung „*Software Engineering I: Softwaretechnik*“ WS10/11, Technische Universität München, Oktober 2010, http://www4.in.tum.de/lehre/vorlesungen/sw_engineering/ws1011/index.shtml

[UNICAM] Internetseite, http://docs.unity3d.com/Documentation/Components/class-Camera.html, vom 04.10.2012

[UNIFEY] Internetseite, http://docs.metaio.com/bin/view/Main/UnifeyeMobileSDK , vom 29.09.2012

[UNIIOS] Internetseite, http://unity3d.com/unity/publishing/ios, vom 07.10.2012

[UNIJNI] Internetseite, http://docs.unity3d.com/Documentation/Manual/PluginsForAndroid.html, vom 02.10.2012

[UNIPLU] Internetseite, http://docs.unity3d.com/Documentation/Manual/PluginsForAndroid.html, vom 04.10.2012

[UNITY4] Internetseite, http://unity3d.com/unity/4/?unity4, vom 01.10.2012

[UNITYE] Internetseite, http://unity3d.com/unity/engine/, vom 01.10.2012

[VTFAR] Klein, Georg *Visual Tracking for Augmented Reality*, Doktorarbeit, Universität von Cambridge, Januar 2006, Seiten 2 ff., 19f., 130 ff.

Anhang

Auf der beiliegenden CD-ROM befinden sich unterschiedliche Dateien und Verzeichnisse, die bereits im Text der Masterthesis referenziert wurden. Darüber hinaus ist dort der Screenflow aus Abbildung in der Datei *Screenflow.png* zur Detailansicht mit besserer Auflösung hinterlegt. Im Verzeichnis „Literature“ befinden sich zusätzlich einige Dokumente des Literaturverzeichnisses die analog zu den Textmarken des Verzeichnisses gekennzeichnet sind.

Das Eclipse-Projekt des Demonstrators befindet sich einem Archiv auf der CD unter dem Namen „Demonstrator_project.zip“. Des Weiteren wurde dort das Archiv „ARPrototyp_project.rar“ hinterlegt, welches das Unity-Projekt des AR-Prototyps enthält.

Auf den nachfolgenden Seiten befinden sich die in der Masterthesis referenzierten Diagramme. Dazu zählen die UML-Klassendiagramme der JSON-Nachrichten zur Client-Server-Kommunikation, das Screenflow-Diagramm des Demonstrators zur Übersicht und das komplette UML-Klassendiagramm des Demonstrators mit einigen Einschränkungen.

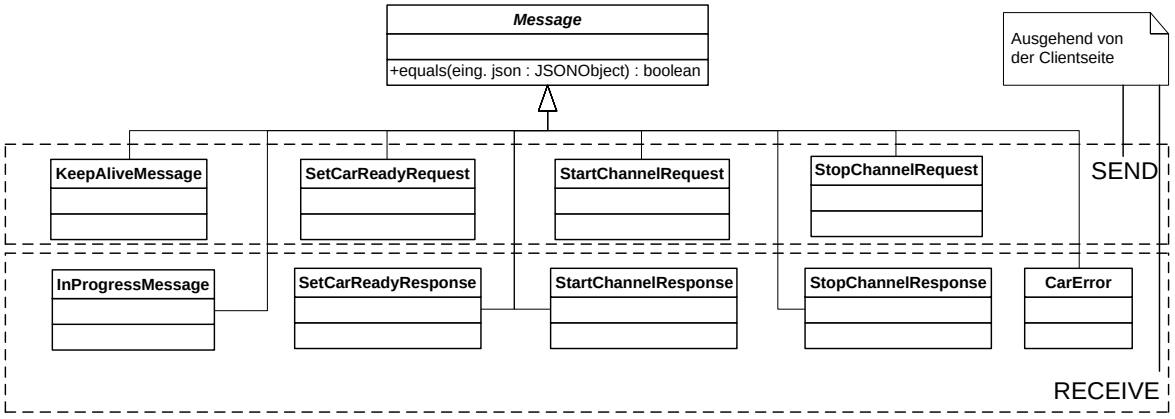


Abbildung 37 JSON-Nachrichten des Pakets messages

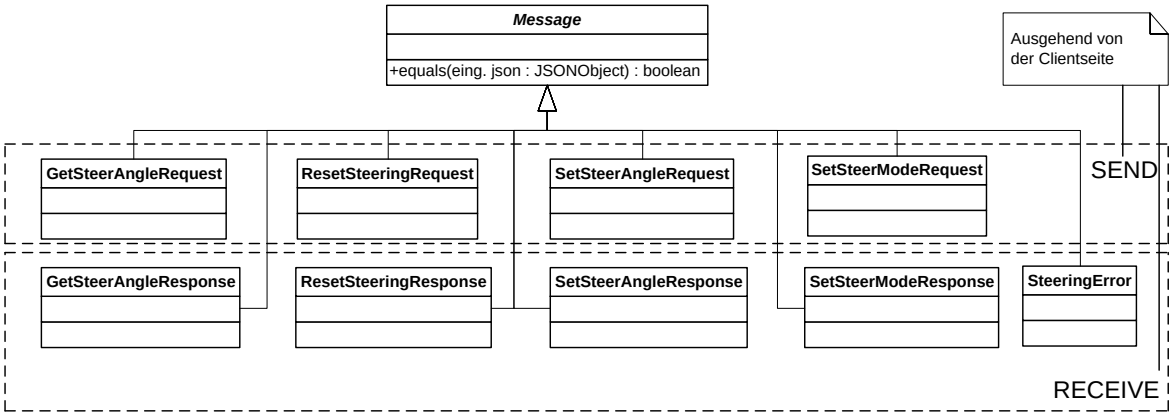


Abbildung 38 JSON-Nachrichten des Pakets steering

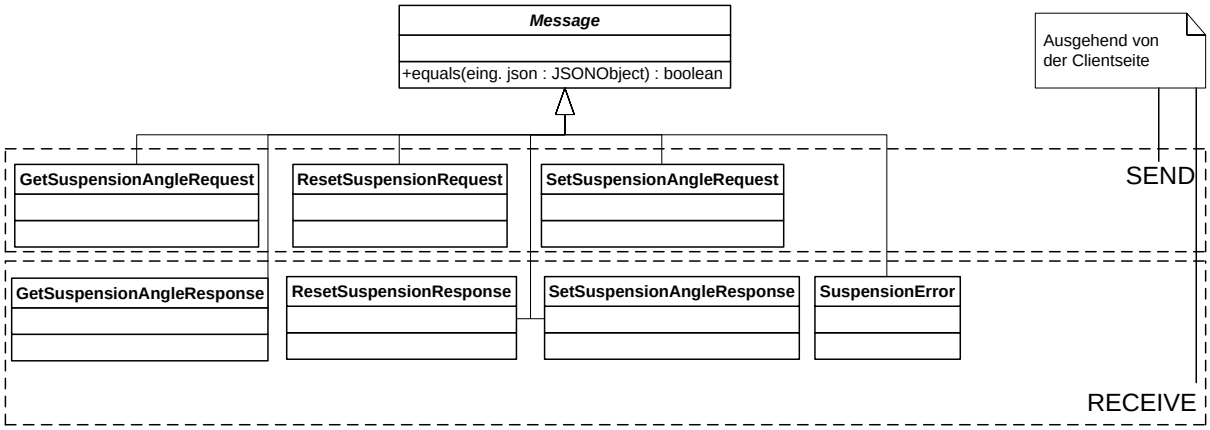


Abbildung 39 JSON-Nachrichten des Pakets suspension

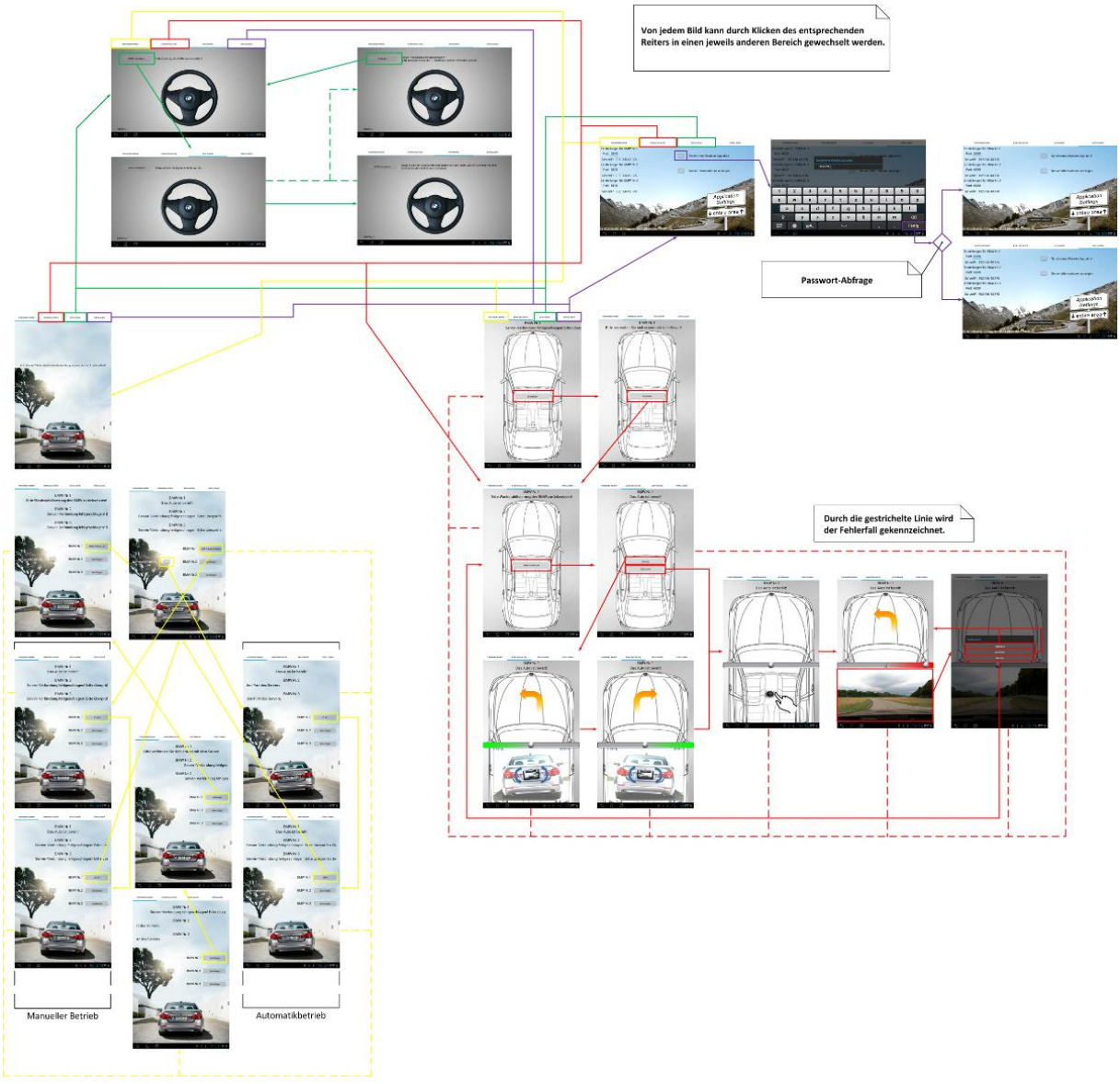


Abbildung 40 Screenflow des Demonstrators, siehe CD-ROM

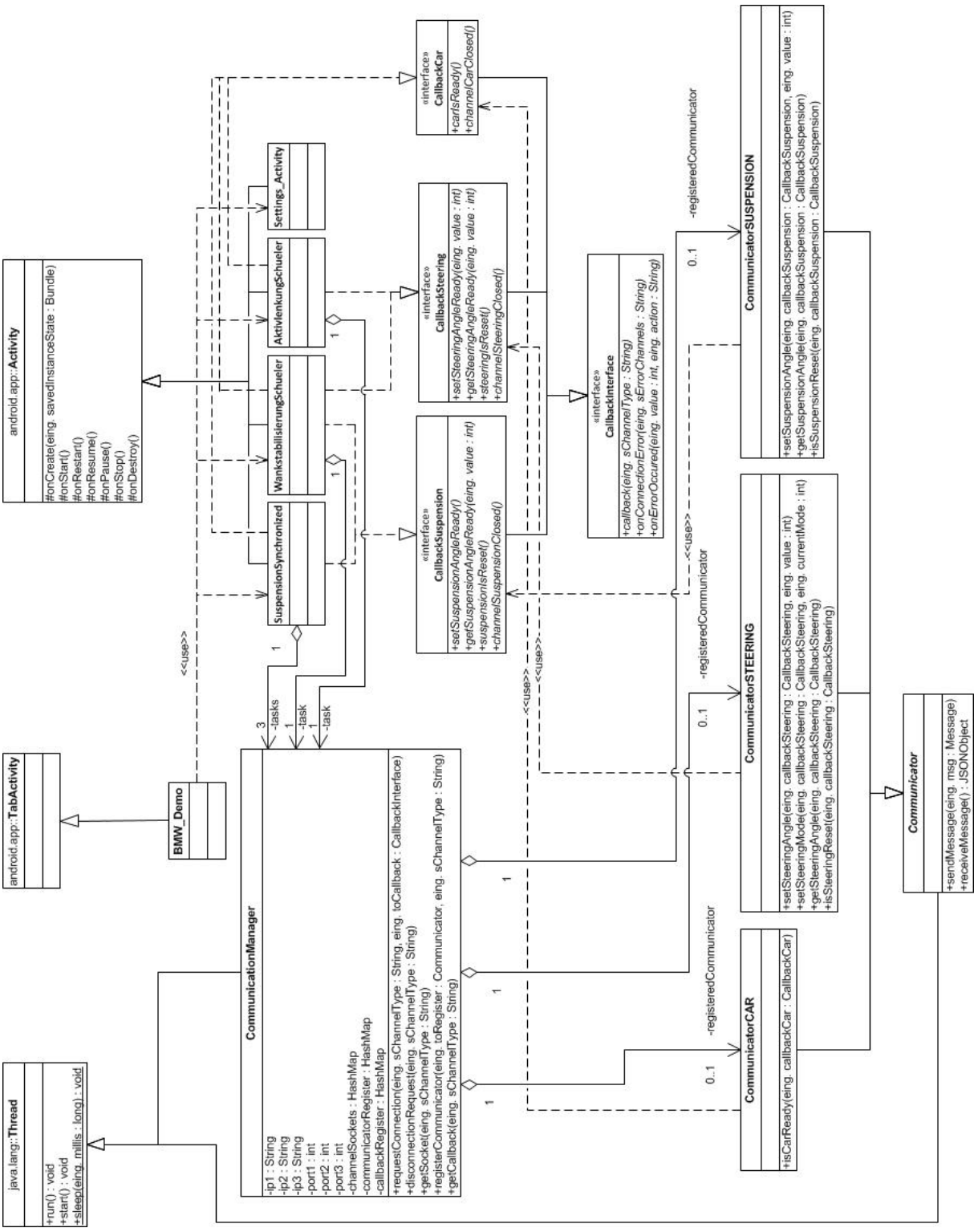


Abbildung 41 UML-Klassendiagramm des Demonstrators mit Einschränkungen