Bachelorarbeit

Jascha Helbig

Entwicklung einer Augmented Reality Modelleisenbahn

Development of an augmented reality model railway

April 2018

Erstprüfer: Prof. Dr. Dr. habil. Karl-Heinz Zimmermann
Zweitprüfer: Prof. Dr. Philipp Jenke

Jascha Helbig

Entwicklung einer Augmented Reality Modelleisenbahn

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Computational Informatics
am Institut für Eingebettete Systeme
der Technischen Universität Hamburg-Harburg
in Zusammenarbeit mit dem Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg
(Ahoi.Digital)

Erstprüfer: Prof. Dr. Dr. habil. Karl-Heinz Zimmermann
Zweitprüfer: Prof. Dr. Philipp Jenke

Eingereicht am: 23.04.2018

Jascha Helbig

Thema der Arbeit
Entwicklung einer Augmented Reality Modelleisenbahn

Stichworte
Augmented Reality, Modelleisenbahn, ARCore, Unity3D

Kurzzusammenfassung
Schaut man sich die Preise von physischen Modelleisenbahnen und ihrer einzelnen Komponenten an, so wird schnell klar, dass die Zielgruppe eher wohlhabend sein muss. Um jedermann die Erfahrung mit einer Modelleisenbahn zu ermöglichen und da ein Augmented Reality fähiges Gerät in Zukunft zur Grundausstattung eines Haushalts gehören wird, soll in dieser Bachelorarbeit eine Augmented Reality Modelleisenbahn entwickelt werden.

Jascha Helbig

Title of the paper
Development of an augmented reality model railway

Keywords
Augmented Reality, Model Railway, ARCore, Unity3D

Abstract
Looking at the prices of physical model railways and their individual components, it quickly becomes clear that the target group has to be rather wealthy. In order to enable an every day experience with a model railway and since a device that is capable of augmented reality will be part of the basic equipment in a household in the near future, an augmented reality model railway shall be developed in this bachelor thesis.

Danksagung

Hiermit möchte ich Prof. Dr. Philipp Jenke meinen Dank aussprechen, der dieser Bachelorabeit mit der Begeisterung für sein Fachgebiet den entscheidenen Anstoß gegeben hat und mir uneigennützig jederzeit mit seinem Rat zur Seite stand. Außerdem möchte ich mich bei Prof. Dr. Zimmermann dafür bedanken, dass er so bereitwillig die Rolle des Erstprüfers übernommen und diese Arbeit damit ermöglicht hat.

Eigenständigkeitserklärung

Ich versichere, die vorliegende Bachelorarbeit selbstständig und lediglich unter Benutzung der angegebenen Hilfsmittel verfasst zu haben.

Ich erkläre weiterhin, dass diese Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Hamburg, 23. April 2018

Jascha Helbig

Inhaltsverzeichnis

Abbildungsverzeichnis

List of Algorithms

1 Einführung

1.1 Motivation

Der technische Fortschritt von Smartphones ist in den vergangenen Jahren kontinuierlich vorangeschritten und ermöglicht, dass man immer aufwändigere Spiele auf ihnen spielen kann. Über 95% der privaten Haushalte in Deutschland sind mit einem Mobiltelefon ausgestattet und der Anteil an Smartphones wird in Zukunft unweigerlich zunehmen [2]. Unabhängig davon ist auch der Markt für herkömmliches Spielzeug stetig gewachsen [3]. Erste Ansätze eine Verbindung zwischen Smartphones und herkömmlichem physischem Spielzeug zu schaffen gibt es schon seit längerem. Beim Projekt *The invisible train* [4] kann der Anwender beispielsweise über ein Augmented Reality(AR) fähiges Gerät einen virtuellen Zug auf einer echten Holzeisenbahnstrecke fahren lassen. Allerdings wird dafür neben einem Smartphone eine große Anzahl physischer Komponenten benötigt, die in ihrer Funktion nicht sehr variabel sind und einen erheblichen finanziellen Mehraufwand erfordern. Besser wäre es, den physischen Anteil auf das Smartphone zu beschränken und trotzdem einen entsprechenden Raum einnehmenden Effekt wie bei einer echten Modelleisenbahn zu erzielen. Die in der vorliegenden Arbeit entwickelte AR-Modelleisenbahn soll diesen Gedanken fortführen, indem sie einem Anwender nicht nur ermöglicht, einen virtuellen Zug zu fahren, sondern auch ein virtuelles Gleissystem zusammenzustellen. Mittels Augmented Reality lassen sich dafür geeignete 3D-Modelle darstellen. Transformationen sind ein elementarer Bestandteil der Computergrafik und erlauben es, 3D-Modelle zu manipulieren oder eine 3D-Umgebung auf einen zweidimensionalen Bildschirm zu projizieren.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines Prototypen einer AR-Modelleisenbahn. Dabei soll es dem Anwender möglich sein auf seinem Handydisplay eine geschlossene Linie zu zeichnen, die anhand der Punkte auf der Linie in eine virtuelle Gleisstrecke umgewandelt wird, welche eine gute Annäherung an die gezeichnete Linie darstellt. Anschließend soll das Gleissystem auf eine Ebene projiziert werden, die mithilfe der von Google Inc. bereitgestellten Software-Bibliothek ARCore [5] berechnet wird. Darüber hinaus wird die Möglichkeit geschaffen einen Zug auf die Gleise zu setzen und mit diesem auf ihnen zu fahren. Im Mittelpunkt der Arbeit stehen der Algorithmus, der aus einer geschlossenen Linie eine fertige Gleisstrecke generiert, und das strukturelle Problem, geschlossene Gleissysteme zusammenzustellen.

1.3 Struktur der Arbeit

Nach einer kurzen Einführung werden in Kapitel 2 die zum Verständnis der Arbeit nötigen Grundlagen und verwandte Arbeiten vorgestellt. Bei dieser Gelegenheit wird der Begriff Augmented Reality erklärt und es erfolgt eine Einführung in Transformationen.

In Kapitel 3 wird das Konzept der Ausarbeitung formuliert und verschiedene Ansätze zur Lösung werden gegeneinander abgewogen.

Das Kapitel 4 beschäftigt sich mit der Umsetzung des Konzepts. Dazu werden die verwendeten 3D-Modelle und die Oberfläche des Prototyps gezeigt sowie bestimmte Funktionen erläutert.

Kapitel 5 enthält eine Einschätzung des entwickelten Konzepts und seiner Umsetzung. Außerdem werden Ideen für mögliche Erweiterungen genannt.

Die Arbeit endet in Kapitel 6 mit einer Zusammenfassung und einem Ausblick in die Zukunft.

2 Grundlagen

Dieses Kapitel gibt eine Einführung in die Grundlagen der Augmented Reality und die in dieser Arbeit verwendete Plattform ARCore. Anschließend werden Transformationen erläutert und zwei Augmented Reality-Anwendungen vorgestellt, die einen Bezug zu dieser Arbeit haben.

2.1 Augmented Reality

Unter Augmented Reality versteht man die Erweiterung der Realitätswahrnehmung durch computergenerierte Inhalte. Diese können prinzipiell alle Sinne ansprechen. Im Allgemeinen beschränkt man sich aber auf die Ergänzung visueller Informationen wie Text, Bilder oder 3D-Modelle. Die Darstellung erfolgt dabei meist über eine AR-Brille oder ein Handydisplay. Abbildung 2.1 zeigt das Mixed-Reality-Spektrum, das Augmented Reality ins Verhältnis zur realen und zur virtuellen Welt setzt. Wie man sieht, steht Augmented Reality der realen Welt näher, da diese durch AR lediglich mit Informationen angereichert wird. Augmented Virtuality hingegen ist überwiegend computergeneriert und wird durch Anteile aus der realen Welt ergänzt. Eine rein computergenerierte Welt stellt schließlich Virtual Reality(VR) dar.

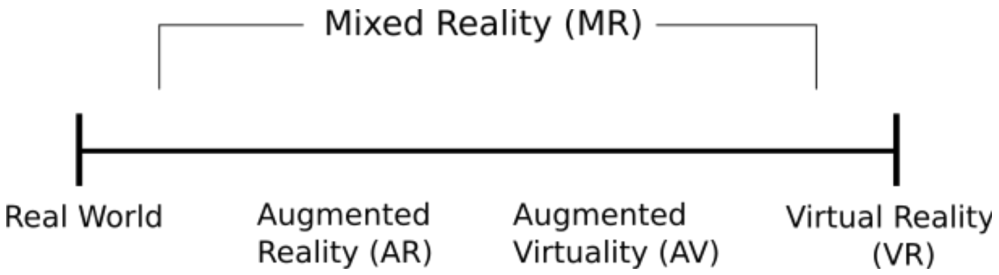

Abbildung 2.1: Mixed-Reality-Spektrum nach Milgram [1]

Nach Azuma [6] ist Augmented Reality durch folgende Charakteristika definiert:

- Kombination aus Realem und Virtuellem
- Interaktion in Echtzeit
- Dreidimensionaler Bezug von virtuellen und realen Objekten

Vergleicht man Augmented Reality nach diesen Charakteristika mit einem 3D-Film, der dem Zuschauer ein bewegtes Bild mit einer Tiefenwahrnehmung vermittelt, dann erkennt man, dass der virtuelle Anteil eines solchen Films bei seiner Entstehung die Aufnahmen zweier Kameras bzw. einer speziellen Kamera erfordert und aufwändig nachproduziert werden muss, um virtuelle Objekte oder Personen mit einem realistischen Tiefeneindruck darzustellen. Augmented Reality hingegen verfolgt den Anspruch den virtuellen Anteil in Echtzeit, also beispielsweise zeitgleich mit einer Bewegung, in korrekter Ausrichtung darzustellen. Um eine Betrachtung aus unterschiedlichen Blickwinkeln zu gewährleisten, muss Augmented Reality im Dreidimensionalen realisiert werden. Die zwei grundlegenden Techniken, die Augmented Reality ermöglichen, sind das Tracking und das Rendering und werden in den beiden folgenden Abschnitten erläutert. Abbildung 2.2 visualisiert wie aus der Realität in Kombination mit der Virtualität Augmented Reality wird. Durch Tracking kann ein virtuelles Objekt an einer gewollten Stelle, angedeutet durch den Punkt im mittleren Bild, platziert und durch Rendering dargestellt werden.

Realität

Augmented Reality

Virtualität[1]

Abbildung 2.2: Das Zusammenspiel von Tracking und Rendering (In Anlehnung an [7]).

[1]Quelle des 3D-Modells: Ronen Horovitz - https://poly.google.com/view/9RWB1a03z0r

2.2 Tracking

Um Augmented Reality-Anwendungen (AR-Anwendungen) zu ermöglichen, ist es notwendig, zunächst die reale Umgebung zu registrieren, um diese anschließend um virtuelle Objekte zu ergänzen. Eine Hard- bzw. Software, die diese Aufgabe erfüllt, wird als Tracker bezeichnet [7].

Bei einer Anwendung, die 3D-Modelle in AR darstellt, muss ein AR-Gerät jederzeit seine aktuelle Position und Orientierung in der realen Welt bestimmen können, damit die Position und Orientierung der abgebildeten virtuellen Welt der der realen Welt entspricht [8]. Ansonsten kann es geschehen, dass virtuelle Objekte nicht an gewünschter Stelle erscheinen oder die Blickrichtung auf ein virtuelles Objekt bei der Betrachtung aus einem anderen Blickwinkel nicht der bei der realen Betrachtung entspricht.

Bei einem Head-up-Display(HUD), einem virtuellen Anzeigesystem im Sichtfeld eines Nutzers, das beispielsweise bei Flugzeugen und zunehmend auch in Autos zur Anwendung kommt, ist es von großer Bedeutung, dass es nicht die Sicht des Nutzers behindert [9].

Es kann allgemein zwischen nichtvisuellem und visuellem Tracking unterschieden werden. Diese werden auf den beiden folgenden Seiten beschrieben.

Nichtvisuelles Tracking

Beim nichtvisuellen Tracking werden meist Signale über eine Kombination aus Sendern und Empfängern ausgetauscht und dadurch die Position oder die Orientierung eines Geräts bestimmt. Zu den nichtvisuellen Tracking-Verfahren zählen unter anderem [7] [10]:

- GPS
- Kompass
- Ultraschallsensoren
- Trägheitssensoren

Abbildung 2.3 zeigt ein Gyroskop, das auch bei einer Neigung seiner Achse seine Lage beibehält, solange es sich schnell genug dreht. Daraus lässt sich die Ausrichtung eines Geräts, in dem ein Gyroskop verbaut wurde, folgern.

Abbildung 2.3: Gyroskop [2]

[2]Quelle: Alega Skolmateriel AB - https://www.alega.se/sv/articles/2.202.1709/gyroskop

Visuelles Tracking

Unter dem visuellen Tracking versteht man Methoden zur Bestimmung von Position und Orientierung eines AR-Geräts oder von Objekte in einem Raum anhand dessen was sichtbar ist, beispielsweise anhand des Bildes einer Kamera. Hierfür gibt es zwei grundlegende Konzepte:

- Marker-basiert
  Die Position und Orientierung eines virtuellen Objekts wird durch einen Marker in der realen Umgebung, wie z.B. einen QR-Code (Abbildung 2.4), bestimmt. Dabei hält der Tracker gezielt Ausschau nach einem oder mehreren Markern, die im Vorfeld definiert wurden, und platziert darauf das virtuelle Objekt [11].

- Markerlos
  Der Tracker sucht nach auffälligen Stellen, den Feature Points, im Kamerabild und versucht dann aus einer Ansammlung davon Muster, wie Ecken und Kanten, zu erkennen, veranschaulicht in Abbildung 2.5 durch die blauen Punkte, um so auch bei einer Bewegung die Lage des Geräts relativ zu den Feature Points bestimmen zu können [12].

Abbildung 2.4: QR-Code

Abbildung 2.5: Suche nach Feature Points

2.3 Rendering

Rendering beschreibt die Darstellung einer 3D-Umgebung mittels Projektion auf einen zweidimensionalen Bildschirm [13]. Diese Umgebung, in der Literatur auch Szene genannt, besitzt ein eigenes Koordinatensystem und eine virtuelle Kamera. Man kann feststellen, dass ein virtuelles Objekt in einer virtuellen Umgebung vorhanden sein muss, um im Kamerabild angezeigt werden zu können.

Virtuelle Kamera

Eine virtuelle Kamera ermöglicht es, ein Objekt von allen Seiten und aus allen Entfernungen zu betrachten. Entweder bewegt man dafür das jeweilige Objekt oder aber die Kamera selber. Es ist möglich, die virtuelle Kamera in jede beliebige Richtung zu bewegen, die uns interessiert, und sie geeignet zu drehen um einen anderen Blickwinkel auf etwas zu bekommen. So wie eine Handykamera den Raum auf einem zweidimensionalen Bildschirm darstellt, so übernimmt die virtuelle Kamera diese Aufgabe für den virtuellen Raum. Damit Realität und Virtualität scheinbar miteinander verschmelzen können, wird der Raum der realen Welt durch ein Koordinatensystem definiert und man versucht ihn mit dem virtuellen Raum abzugleichen. Legt man die Handykamera und die virtuelle Kamera praktisch aufeinander, dann betrachten beide Kameras in ihren jeweiligen Umgebungen den gleichen Raum. Bewegt man nun die Handykamera entsprechend dem aufgespannten Koordinatensystem in eine Richtung, dann wird die virtuelle Kamera im virtuellen Raum in die gleiche Richtung bewegt. Stimmen die Systeme nicht überein wie in Abbildung 2.6 veranschaulicht, folgen daraus Fehler in der Betrachtung. Aus diesem Grund ist ein gutes Tracking wichtig.

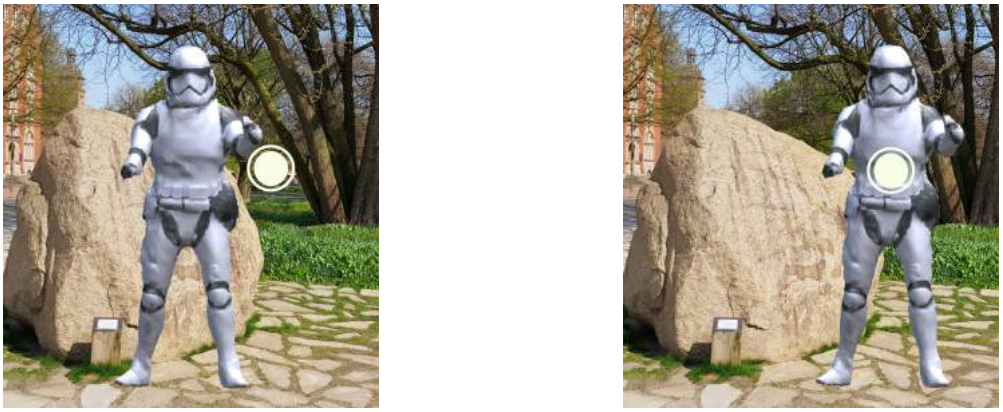

Abbildung 2.6: Auf dem linken Bild wird der Stormtrooper nicht an der gewollten Stelle dargestellt, da die virtuelle Kamera und die Kamera des Smartphones schlecht aufeinander registriert sind. Im Gegensatz dazu sind die Kameras auf dem rechten Bild gut aufeinander registriert (In Anlehung an [14]).

Projektion

Der von der virtuellen Kamera erfassbare Raum, welcher durch eine nahe und eine ferne Clipping-Ebene begrenzt ist, wird schließlich auf eine Ansichtsebene projiziert. Alle Objekte die sich vor der nahen bzw. hinter der fernen Clipping-Ebene befinden, werden nicht angezeigt, was die Rechenzeit erheblich verringert. Zwei grundlegende Projektionen sind die perspektivische Projektion (Abbildung 2.7) und die parallele oder orthographische Projektion (Abbildung 2.8). Die perspektivische Projektion ist mit dem menschlichen Auge oder einer Kamera zu vergleichen, da perspektivische Verkürzung einen Eindruck von Tiefe erzeugt. Um die Projektion eines 3D Punktes zu bestimmen, verbindet man ihn durch eine Gerade mit dem Projektionszentrum. Bei der parallelen Projektion werden Maßstab und Form eines Objekts beibehalten. Die Bildpunkte liegen beim Schnittpunkt der Ansichtsebene mit einem Strahl ausgehend vom Punkt eines Objekts mit einer festen Richtung.

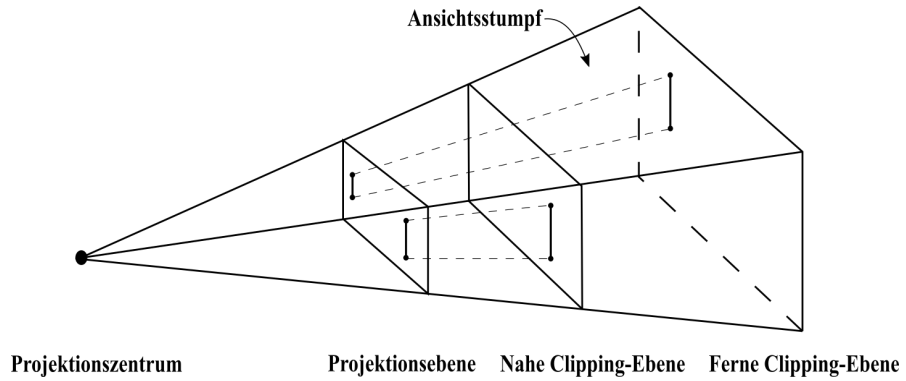

Abbildung 2.7: Perspektivische Projektion

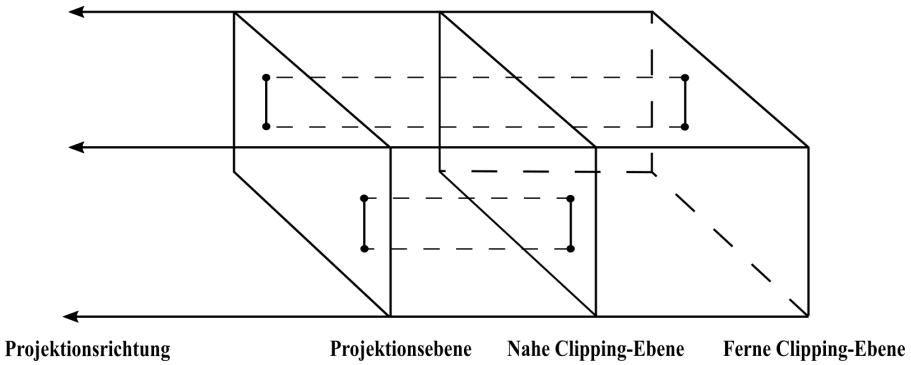

Abbildung 2.8: Parallele Projektion

2.4 Transformationen

Nachdem gezeigt wurde, wie sich ein Gerät in seiner Umgebung wiederfindet und virtuelle Objekte abbildet, wird nun beschrieben, wie Objekte in der virtuellen Umgebung bewegt werden können. Hierzu werden Transformationen verwendet. Sie ermöglichen unter anderem, dass wir Objekte zu einer bestimmten Position bewegen, sie skalieren oder rotieren lassen. Dafür wird jeder Punkt eines Objekts der gleichen Transformation unterzogen. Die gängigste Darstellung hierfür ist eine Transformationsmatrix.

In der Matrixnotation wird ein Punkt $\mathbf{v}$, dargestellt durch seinen Ortsvektor, durch eine Translation, eine Skalierung bzw. eine Rotation folgendermaßen transformiert [15]:

$\mathbf{v'} = \mathbf{v} + \mathbf{t}$
$\mathbf{v'} = \mathbf{S} \cdot \mathbf{v}$
$\mathbf{v'} = \mathbf{R} \cdot \mathbf{v}$

wobei $\mathbf{t}$ der Translationsvektor ist, und $\mathbf{S}$ und $\mathbf{R}$ Skalierungs- beziehungsweise Rotationsmatrizen sind.

Wie man sieht, wird die Translation durch eine Addition von Punkten bzw. deren Ortsvektoren mit einem Translationsvektor und eine Skalierung bzw. Rotation durch Multiplikationen dieser Ortsvektoren mit der Skalierungs- bzw. Rotationsmatrix beschrieben. Um eine einheitliche Schreibweise zu ermöglichen, verwenden wir ein homogenes Koordinatensystem, indem die Dimension um eins erhöht wird. Also von $\mathbb{R}^2$ nach $\mathbb{R}^3$ im zweidimensionalen Raum oder von $\mathbb{R}^3$ nach $\mathbb{R}^4$ im dreidimensionalen Raum. Nachfolgend werden die zuvor erwähnten Transformationen in ein homogenes Koordinatensystem im $\mathbb{R}^4$ überführt, aber die Beispiele werden im xy-Koordinatensystem veranschaulicht. Es folgt [15]:

Translation

Was vorher eine Addition war, kann nun als Multiplikation mit einer Translationsmatrix dargestellt werden. Eine Translation verschiebt ein Objekt.

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \Rightarrow \qquad \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Beispiel

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

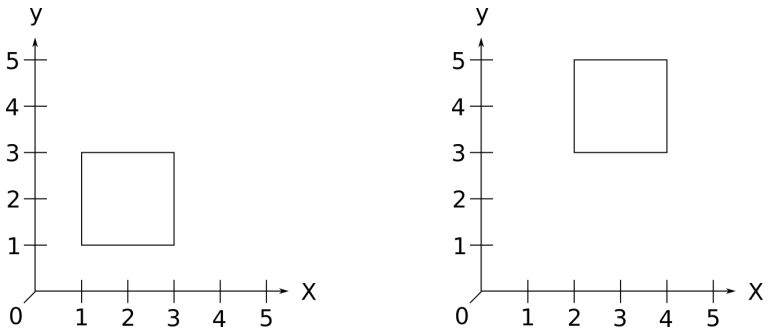

Abbildung 2.9: Objekt vor und nach der Translation

Skalierung

Multiplikation mit einer Skalierungsmatrix. Eine Skalierung streckt oder staucht ein Objekt entlang der Achsen mit dem jeweiligen Skalierungsfaktor. Bei einheitlicher Skalierung gilt $s_x = s_y = s_z$.

$$\mathbf{S} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \Rightarrow \qquad \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Beispiel

$$\mathbf{T} = \begin{pmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

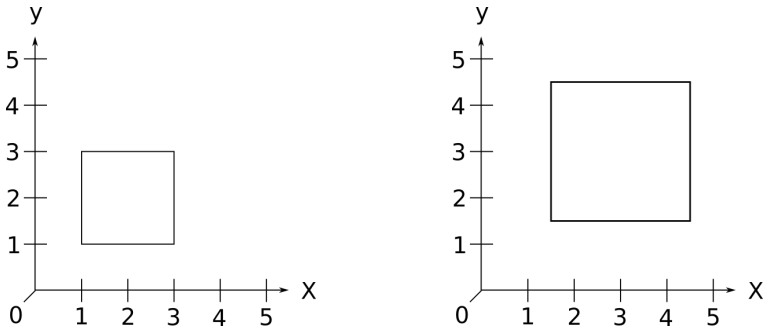

Abbildung 2.10: Objekt vor und nach der Skalierung

Rotation um eine Koordinatenachse

Um ein Objekt zu rotieren, benötigt man eine Rotationsachse. Theoretisch kann diese eine beliebige Ausrichtung haben, allerdings ist es einfacher, wenn man ein Objekt um eine der Koordinatenachsen rotieren lässt. Daher werden hier zunächst Rotationen um den Winkel θ um die verschiedenen Koordinatenachsen betrachtet.

$$\mathbf{R}x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}y = \begin{pmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}z = \begin{pmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Beispiel

$$\mathbf{T} = \begin{pmatrix} 0,866 & 0.5 & 0 & 0 \\ -0.5 & 0,866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

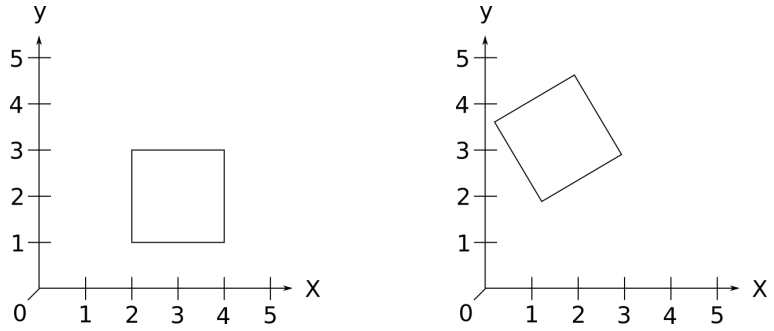

Abbildung 2.11: 30 Grad Rotation um den Ursprung

Abbildung 2.11 zeigt eine 30-Grad-Rotation um die z-Achse. Da das dargestellte Objekt ein Quadrat in der xy-Koordinatenebene ist, kann diese Rotation auch als eine um den Ursprung des xy-Koordinatensystems interpretiert werden. Durch eine Abfolge von Rotationen um verschiedene Koordinatenachsen lassen sich Objekte im dreidimensionalen Raum ausrichten. Für eine Rotation zu einer beliebigen Orientierung aus einer beliebigen anderen können Euler-Winkel verwendet werden, indem man drei Rotationen mit unabhängigen Drehwinkeln hintereinander ausführt, wobei sich die Achsen mitdrehen und sich für die jeweils folgende Drehung eine neue Rotationsachse mit x' != x bzw. x" != x' ergibt. Bei der Standard-x-Konvention(z, x', z") wird um die z-Achse mit dem Winkel α, dann um die daraus resultierende x-Achse x' mit dem Winkel β und um die daraus resultierende z-Achse z" mit dem Winkel γ rotiert [16]. Eine Besonderheit hierbei ist der sogenannte Gimbal Lock, der daraus resultiert, dass zwei Achsen nach einer Rotation parallel zueinander stehen, wodurch ein Freiheitsgrad verloren geht und eine Rotation auf zwei Dimensionen beschränkt wird. Eine Möglichkeit das Problem zu umgehen sind Quaternionen. Diese werden durch eine vierdimensionale hyperkomplexe Zahl, die die Lage der Rotationsachse mitsamt dem Drehwinkel repräsentiert, beschrieben [17].

Ein weiterer Vorteil beim Rechnen mit Quaterionen im Vergleich zu Matrizen ist der geringere Speicherverbrauch, da in einer Quaternion nur vier und in einer homogene Matrix sechszehn Zahlen gespeichert werden müssen. Allerdings sind Quaternionen intuitiv nicht leicht verständlich.

Rotation um eine beliebige Achse

Um ein Objekt um eine beliebige Achse zu rotieren, verschiebt man es in den Ursprung, rotiert es dann um einen Winkel θ um eine der Koordinatenachsen und verschiebt es anschließend zum Ausgangspunkt zurück. Zu diesem Zweck müssen zwei zusätzliche Matrixmultiplikationen durchgeführt werden. Abbildung 2.12 zeigt das beschriebene Vorgehen.

Beispiel

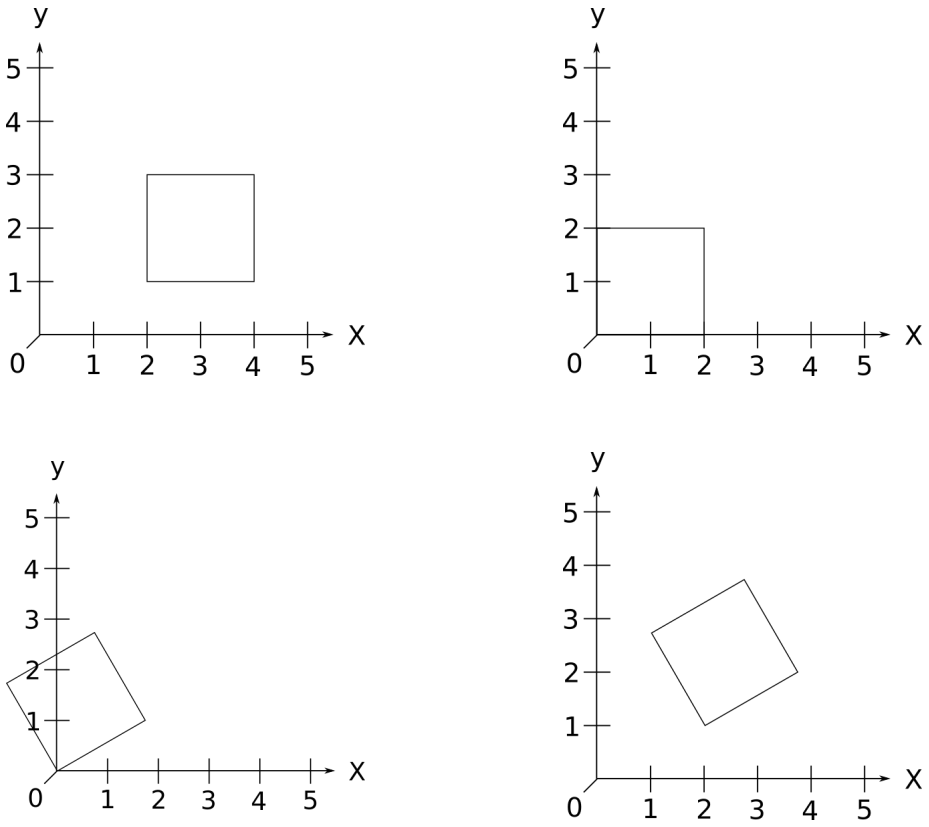

Abbildung 2.12: 30 Grad Rotation um eine beliebige Achse

Koordinatentransformation

Ein Modell wird bei seiner Entstehung in einem eigenen lokalen Koordinatensystem definiert, wodurch jeder seiner Punkte relativ zum Ursprung dieses Systems beschrieben werden kann. Möchte man mehrere Objekte in einer Umgebung darstellen, müssen diese in ein sogenanntes Weltkoordinatensystem transformiert werden, indem Verschiebungen, Skalierungen und Rotationen auf sie angewendet werden, um sie zu positionieren. Soll ein in System 1 festgelegter Punkt in System 2 überführt werden, dann wird erst die Position und Ausrichtung des Ursprungs von System 1 mit dem Ursprung von System 2 gleichgesetzt und dann die Inverse dieser Transformation auf den Punkt angewendet [15]. Koordinatentransformationen überführen Punkte aus einem Koordinatensystem in ein anderes.

Allgemein

Homogene Koordinaten ermöglichen die Vereinheitlichung und Kombination aller geometrischen Transformationen. [18]

Transformationen werden in der Reihenfolge $T_1, T_2, ..., T_n$ ausgeführt $\Rightarrow P' = T_n * ... T_2 * T_1 * \mathrm{P}$. Die Reihenfolge ist bei der Ausführung unterschiedlicher Transformationen von Bedeutung, da diese im Allgemeinen nicht kommutativ sind.

Generelle Transformationsmatrix in 3D und in 2D:

$$T = \begin{pmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Die obere linke 3x3-Teilmatrix in 3D bzw. 2x2-Teilmatrix in 2D enthält die Gesamtrotation bzw. die Skalierung, während $\mathbf{t}$ die Gesamttranslation beschreibt. [15] Weitere Transformationen, die sich durch Matrizen darstellen lassen, wie die Spiegelung und die Scherung, können ebenfalls in der generellen Transformatonsmatrix enthalten sein.

2.5 ARCore

Für die Entwicklung des Prototypen benötigen wir als Hilfsmittel noch eine Augmented Reality-Bibliothek, die das Tracking übernimmt und geeignete Werkzeuge bereitstellt, um virtuelle Objekte zu platzieren und zu handhaben. Aufgrund persönlicher Preferenzen, der Verbindung von ARCore zu Unity [19], das als Platform für die Umsetzung dieser Arbeit dient, und der Nähe von ARCore zu Smartphones mit einer Android Version von 7.0 und höher, wird in dieser Arbeit das Software Development Kit von Google verwendet. ARCore stellt Klassen, Strukturen und vorgefertigte Modelle bereit, um eine Augmented Reality-Anwendung möglich zu machen. Diese erlauben unter anderem die Verwendung der Handykamera, das Szenenmanagement im virtuellen Raum und das Tracking der gefundenen Objekte in der realen Welt. Vorab eingestellt ist bereits die Verbindung zu einer ARCore-Session, welche die Handykamera initialisiert und nach Oberflächen sucht. Wird eine Oberfläche gefunden, dann können ein oder mehrere Objekte darauf platziert werden. Die Verwendungsmöglichkeiten von ArCore beruhen hierbei auf drei Schlüsseltechnologien [20]:

Bewegungstracking

Bewegungstracking erlaubt es einem Gerät die eigene Position relativ zur echten Welt zu erfassen. Das ermöglicht beim Herumgehen um ein virtuelles Objekt das Betrachten des Objekts aus einem beliebigen Blickwinkel. ARCore kombiniert dabei mehrere Trackingverfahren. Zum einen berechnet ein Algorithmus visuell eindeutige Merkmale, nämlich die Feature Points, im Bild der Kamera und benutzt sie um die Veränderung der Position und Orientierung relativ zur Startposition auszurechnen, was als visuelle Odometrie oder visuelle Navigation bezeichnet wird. Zum anderen stellt die Inertial Measurement Unit (IMU) des Geräts, die unter anderem Beschleunigungs- und Drehratensensoren enthält, mittels Trägheitsmessung Veränderungen in der Ausrichtung und Beschleunigung fest, um ihrerseits die Position und Orientierung der Kamera relativ zur realen Welt auszuwerten. Die Kombination aus visueller Odometrie und einer IMU wird visuelle Trägheitsnavigation genannt [21]. Dadurch lassen sich die virtuelle Kamera mit der Kamera des Geräts abstimmen und die virtuellen Objekte in der richtigen Perspektive anzeigen.

Registrierung der Umgebung

Die Registrierung der Umgebung versetzt ein Gerät in die Lage horizontale Oberflächen und deren Größen zu erkennen. Dabei sucht ARCore nach Ansammlungen von Feature Points, die auf ein und derselben Oberfläche zu liegen scheinen. Bei einer gewissen Anzahl an Feature Points wird ein Bereich als Oberfläche erkannt und als Ebene in der App angezeigt (Abbildung 2.13). Erwähnenswert ist hierbei der RANSAC-Algortihmus, der aus einer Datenmenge mit Ausreißern iterativ mathematische Modelle abschätzt [22].

Dabei lassen sich Oberflächen mit auffälliger Textur besser erkennen als solche ohne Textur. Zusätzlich dazu kann ARCore durch die Ergänzung von Oriented Points die korrekte Orientierung virtueller Objekte auf nicht horizontalen Oberflächen bestimmen, ohne sie als solche zu erkennen, indem ihr Neigungswinkel relativ zur Fläche abgeschätzt wird.

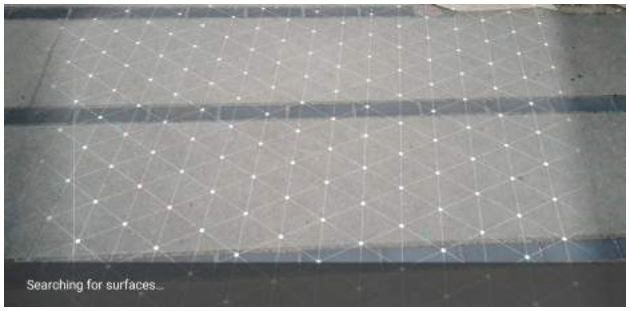

Abbildung 2.13: ARCore erkennt die Oberfläche eines Gehwegs und visualisiert sie als Ebene.

Abschätzung der Lichtverhältnisse

Die Abschätzung der Lichtverhältnisse erlaubt beispielsweise zwischen Tag und Nacht zu unterscheiden. ARCore bestimmt dafür die unterschiedlichen Lichtintensitäten des Kamerabildes und ermöglicht somit die Entwicklung noch realistischerer Anwendungen. Die Zufriedenheit der Sonnenblume aus Abbildung 2.14 ist von der Lichtintensität im Raum abhängig.

Abbildung 2.14: Abschätzung der Lichtverhältnisse[3]

[3]Quelle: Dan Miller - https://blogs.unity3d.com/2018/03/07/how-arcore-enables-you-to-create-brand-new-types-of-user-interaction/

2.6 Anwendungsbeispiele

Im Folgenden werden zwei Anwendungen vorgestellt, die die Möglichkeiten von AR verdeutlichen sollen.

Motion Stills

Mit der App *Motion Stills*[4] von Google Research lassen sich Videos und GIF-Bilder erstellen. Im AR-Modus kann der Anwender aus einer Anzahl virtueller Objekte auswählen und diese in der Kameraansicht vor sich platzieren. Zudem ist es mittels einer einfachen Handgeste und ohne Verwendung des Touchscreens möglich platzierte Objekte aufzunehmen und an anderer Stelle wieder abzusetzen (Abbildung 2.15). Motion Stills zeigt damit, dass es nicht nur möglich ist mit Objekten zu interagieren, sondern auch virtuelle Objekte mit ihrer unmittelbaren Umgebung interagieren zu lassen.

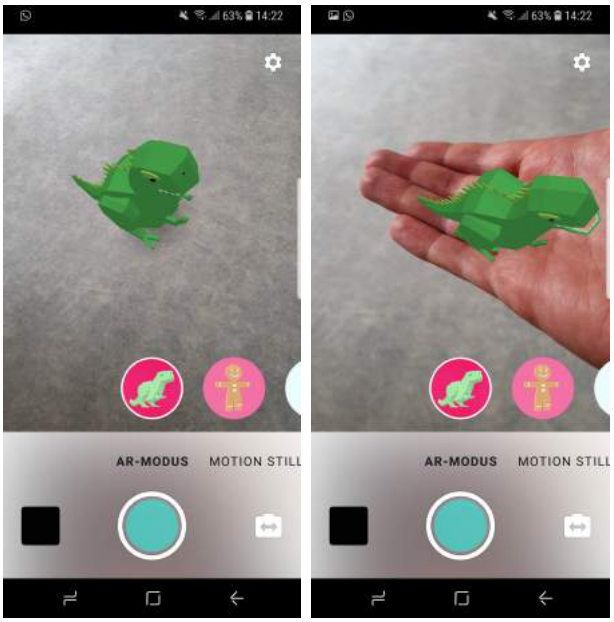

Abbildung 2.15: Die AR-App MotionStills

[4]MotionStills, Google Research - Verfügbar unter: https://get.google.com/motionstills/

Just a line

Mit *Just a line*[5] von Google Creative Lab lassen sich jederzeit und überall simple Zeichnungen in AR anfertigen. Bewegt man sich beim Zeichnen, entstehen dreidimensionale Bilder. Um einen Einfluss auf die Tiefe zu haben, ist es komplementär möglich eine Zeichnung aus einem anderen Blickwinkel heraus zu ergänzen. Ferner kann man von sich und seinen gezeichneten Linien ein Video anfertigen und es teilen. In Bezug auf diese Arbeit veranschaulicht Just a line, dass nicht zwingend eine von ARCore erkannte Oberfläche benötigt wird um eine Linie in AR zu zeichnen, da die Position einer Linie auch mittels anderer Tracking-Methoden festgehalten werden kann.

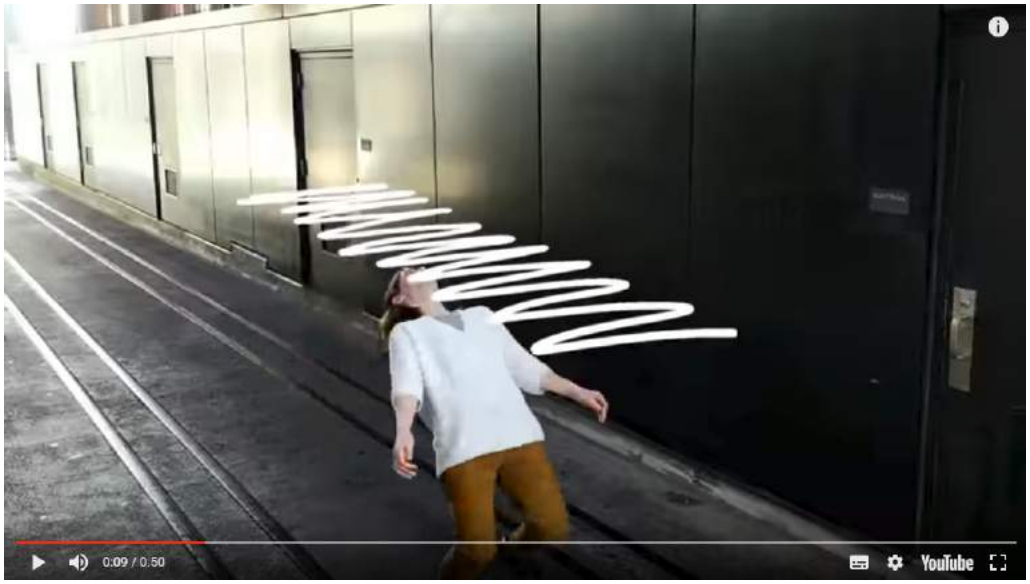

Abbildung 2.16: Die AR-App JustALine[6]

[5] JustALine, GoogleCreativeLab - Verfügbar unter: https://justaline.withgoogle.com/
[6] Quelle: Ausschnitt eines Youtube-Videos von JustALine von Google - Verfügbar unter: https://www.youtube.com/watch?v=M5zxNMAQVR0

3 Konzeption

In diesem Kapitel wird noch einmal die grundsätzliche Idee dieser Arbeit beschrieben. Anschließend werden die einzelnen Unterpunkte zur genaueren Begutachtung in Abschnitte unterteilt und mögliche Umsetzungen der Idee gegeneinander abgewogen.

In dieser Arbeit soll ein Prototyp einer AR-Modelleisenbahn entwickelt werden. Dazu skizziert man mit dem Finger auf dem Display eines Smartphones eine geschlossene Linie, die den gewünschten Verlauf der Gleise der Modelleisenbahn darstellen soll. Aus einem bestehenden Satz unterschiedlicher Schienensegmente soll dann ein Gleissystem aufgebaut werden, das möglichst genau der skizzierten Linie entspricht. Daraufhin wird eine Ebene, auf die das fertige Gleissystem projiziert werden soll, in der Umgebung identifiziert.

Abschließend soll ein Zug auf das Gleissystem gesetzt werden, mit dem ein Anwender in der Lage ist auf den Schienen zu fahren. Algorithmus 1 beschreibt den ungefähren Ablauf.

Algorithmus 1 Entstehung einer AR-Modelleisenbahn

1: Gleisverlauf im 2D auf dem Display skizzieren
2: Gleissystem aus dem Gleisverlauf generieren
3: Ebene in Umgebung identifizieren
4: Gleissystem auf Ebene projizieren
5: Zug platzieren
6: Zuf auf Gleissystem fahren lassen

3.1 Skizzierung eines Gleisverlaufs

Durch eine Touch-Bewegung soll eine Linie auf dem Display skizziert werden, die als Linienzug, also eine geordnete Liste von n Punkten, P = $\{p_i\}$, $p_i \varepsilon \mathbb{R}^2$ mit $p_0 = p_{n-1}$ repräsentiert wird und am Ende geschlossen ist. Dabei muss zwischen zwei Grundkonzepten unterschieden werden. Eines von ihnen besteht darin, die Linie in einer 2D-Umgebung zu zeichnen. In diesem Fall lässt sich die gewünschte Strecke intuitiv in eine Linie übersetzen. Die zweite Möglichkeit sieht vor, die Linie in der 3D-Kameransicht zu skizzieren. Ein Ansatz wie bei *Just a line* ist in der Umsetzung dann allerdings schwierig, weil der 3D-Aspekt hinzukommt. Dafür müssten die grundsätzliche Ausrichtung der Linie, also die Lage wo oben und unten ist, und Steigungen definiert werden. In Anbetracht dessen, dass bei der Erstellung eines Gleissystems mit Steigungen noch zusätzliche Gleisbausteine nötig wären, würde der 3D-Ansatz über den Rahmen dieser Arbeit hinausgehen. Würde man auf eine Achse verzichten und dadurch aus dem 3D-Ansatz einen 2D-ähnlichen Ansatz generieren, wäre die ursprünglich in 3D gezeichnete Linie zu einer Seite hin "platt gedrückt" und würde dem Anwender im Vergleich zu einer reinen 2D-Umgebung keinerlei Vorteil bieten. Folglich bleibt nur noch die Möglichkeit, eine Linie in einer 2D-Umgebung zu zeichnen und diese direkt auf eine vorhandene Ebene zu projizieren. Dadurch wird es möglich, den ungefähren Verlauf der späteren Strecke nachzuvollziehen und potentielle Hindernisse in ihrer Umgebung zu umkurven. Die fertige Strecke könnte allerdings von der Linie abweichen, weil sie mit einem vorgegebenen Satz aus Schienensegmenten nicht akkurat in ein Gleissystem umgewandelt werden kann. Bei beiden Optionen stellt sich die Frage, ob man eine Freihandzeichnung anfertigen soll oder eine Linie nur nach bestimmten Kriterien weiter gezeichnet wird. Beim Freihandzeichnen besitzt der Anwender die größte Freiheit, allerdings müssen Kontrollbedingungen ergänzt werden, um später eine valide Strecke zu gewährleisten. Hinsichtlich der Aufgabenstellung dieser Arbeit bedeutet dies, Linien mit zu spitzen Kurven entweder direkt zu glätten oder bei der Übersetzung in eine Gleisstrecke zusätzliche Hilfskurven einzubauen. Das direkte Glätten der Linie würde die Notwendigkeit zusätzlicher Hilfskurven allerdings nicht grundsätzlich ausschließen. Die andere Variante ließe sich durch ein Schachbrett- oder Hexagonmuster verwirklichen, indem man den Linieneintritt je nach vorheriger erlaubter Austrittsrichtung steuert. Jedoch wird der Anwender dadurch sehr eingeschränkt. Eine Hybridlösung könnte darin bestehen, eine freihändig gezeichnete Linie nach den Kriterien eines auferlegten Musters auszuwerten. Sollte der Finger des Anwenders beim Skizzieren eine gewisse Distanz d zu p_0 unterschreiten, kann die Linie automatisch mit $p_{n-1} = p_0$ geschlossen werden. Für den Fall, dass der Anwender keine geschlossene Linie zeichnet, muss der Gleisverlauf vervollständigt werden.

3.2 Generierung eines Gleissystems aus dem skizzierten Gleisverlauf

Gegeben sind eine geschlossene Linie P = $\{p_i\}$ und ein Satz unterschiedlicher Schienensegmente S = $\{S_j\}$. Beispielsweise könnte sein: S_0 = Gerade der Länge l_0 und S_1 = Kurve der Länge l_1 mit Ausrichtung a_1. Ein Gleissystem G = $\{G_i\}$ ist eine Liste von Schienensegmenten, z.B. G = [S_0,S_1,S_1,S_0]. Es gilt nun zu entscheiden, wie die Punkte der Linie ausgewertet werden. Je mehr verschiedene Kurvenstücke in einem Satz von Schienensegmenten vorhanden sind, desto komplexer wird die Auswertung der Punkte. Am intuitivsten ist es, die Kante zweier benachbarter Punkte als Schienensegment zu interpretieren. Ist der Abstand der gezeichneten Punkte ungefähr gleich der Länge eines Schienensegments, dann passt sich das Gleissystem größentechnisch der gezeichneten Linie an. Ansonsten kann es auch merklich länger oder kürzer als die Linie ausfallen, was die Benutzung der Anwendung unschöner macht. Sollte man trotzdem einen beliebigen Abstand zwischen zwei Punkten verwenden wollen, muss Folgendes beachtet werden. Ist der Abstand zweier Punkte kleiner als die Länge eines Schienensegments, dann muss entweder das Modell der Schiene in Richtung der Längsachse gestaucht oder der Abstand eines der Punkte zum darauf folgenden Punkt gewählt werden. Ersteres hat unterschiedlich gestreckte Schienen zur Folge, was optisch nicht ansprechend wäre. Sollte hingegen der Abstand zweier Punkte größer als die Länge eines Schienensegments sein, dann muss der restliche Abstand vom Ende eines Schienensegments zu dem zweiten Punkt mit weiteren Schienen aufgefüllt werden. Darüber hinaus stellt sich die Frage, ab wann ein Linienabschnitt eine echte Kurve ist. Dementsprechend wird dann ein gekrümmtes oder gerades Schienensegment verbaut werden. Zuvor wurde bereits herausgearbeitet, dass in dieser Arbeit das Zeichnen in einer 2D-Umgebung am sinnvollsten ist. Entsprechend muss der Abstand zweier Punkte in der xy-Ebene betrachtet werden. Die Auswertung der Punkte zeigt Algorithmus 2, der aus dem Abstand von p_i und p_{i+1} die Ausrichtung a_i eines Schienensegments bestimmt. Abbildung 3.1 stellt dar, in welchen Bereich der Abstand zweier Punkte für die jeweilige Ausrichtung fallen muss. Dadurch lässt sich eine Richtungsänderung im Gleisverlauf erkennen und das nächste Schienensegment herleiten.

Algorithmus 2 Auswertung der Punkte des Gleisverlaufs

```
Gegeben sind n Punkte
for i = 0 to n − 1 do
  if i = n − 1 then
    a_i = a_0
  end if
  if Differenz aus p_i und p_{i+1} in Bereich 0 then
    a_i = 0
  else
    Teste andere Bereiche bis der richtige gefunden ist
  end if
end for
```

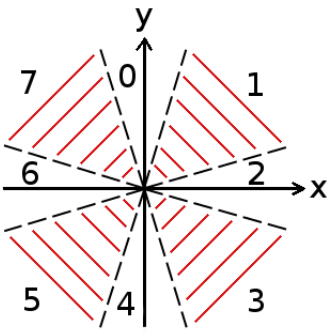

Abbildung 3.1: Definitionsbereiche der Ausrichtungen

Ein Schienensegment ist wie folgt definiert:

- l: Länge l
- a: Ausichtung a (a = [0-7])
- e: Endpunkt e
- prev: Referenz auf das vorherige Gleis
- next: Referenz auf das nachfolgende Gleis

Es fällt auf, dass der Startpunkt des Schienensegments fehlt. Dies ergibt sich daraus, dass wir das Gleissystem als Verkettung von Schienensegmenten betrachten. Der Endpunkt des vorherigen Schienensegments legt den Startpunkt des nächsten fest. Die Ausrichtung entspricht dabei der Endausrichtung des vorangegangenen Gleisstückes. Aus der Länge kann die korrekte Verschiebung in eine Richtung abgeleitet werden. Algorithmus 3 setzt nun das Gleissystem zusammen. Das erste Gleisstück muss dabei gesondert behandelt werden, weil es kein vorheriges besitzt.

Algorithmus 3 Generierung des Gleissystems

```
for i = 0 to n − 1 do
  if i = 0 then
    Positionierung des ersten Gleises
  else
    Prüfung des nächsten Gleises
  end if
end for
```

Positionierung des ersten Gleises

Für das erste Gleisstück muss festgelegt werden, ob es Teil einer Geraden (Im Folgenden kurz Gerade genannt) oder eine gekrümmte Kurve (Im Folgenden kurz Kurve) sein soll. Der Einfachheit halber kann das erste Gleis als Gerade definiert werden. Es wird angenommen, dass der Bezugspunkt eines Schienensegments in der Mitte des Modells liegt. Der Bezugspunkt der Geraden S_0 wird nun auf Punkt p_0 gelegt, woraufhin das Gleisstück noch in die zuvor definierte Ausrichtung a_0, verdeutlicht in Abbildung 3.2, verschoben und dann entsprechend dieser rotiert wird. Hierfür wird angenommen, dass die Anfangsausrichtung eines Schienensegments die Richtung 0 nach oben zeigend ist. Daher führt die Rotation aus der Multiplikation der Ausrichtung mit der Minimalrotation zur nächsten Ausrichtung, bei 8 Möglichkeiten also 45 Grad, zur tatsächlich benötigten Ausrichtung. Das Gleis kann sozusagen als Kante zwischen p_0 und p_1 angesehen werden. Der Endpunkt e_0 des Gleises entspricht dem Orstvektor von p_0 addiert mit der Länge l_0 der Geraden in Verschiebungsrichtung. Algorithmus 4 beschreibt das beispielhaft.

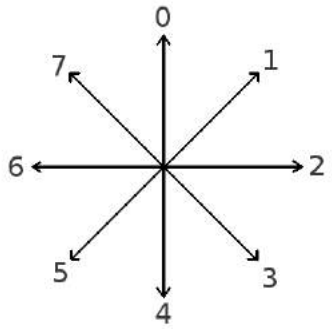

Abbildung 3.2: Verschiebungsrichtung eines Gleissegments. Beispielsweise in positive x-Richtung für Ausrichtung 2 und in negative x- sowie negative y-Richtung für Ausrichtung 5.

Algorithmus 4 Positionierung des ersten Gleises

Input: p_0, S_0
Output: G_0
1: **if** $a_0 = 0$ **then**
2: Lege Verschiebung in Richtung 0 um $l_0/2$ fest
3: **else**
4: Lege andere Verschiebungen fest
5: **end if**
6: Setze S_0 auf p_0 + die festgelegte Verschiebung
7: Rotiere S_0 um Minimalrotation $\cdot$ a_0
8: Endpunkt $e_0 = p_0 + l_0$ in Verschiebungsrichtung

Positionierung der weiteren Gleise

Bevor ein weiteres Gleis hinzugefügt werden kann, muss mithilfe von Algorithmus 5 überprüft werden, ob die Richtungsänderung zum vorherigen Gleis noch durch eine Kurve aus dem gegebenen Satz von Schienensegmente dargestellt werden kann. Sollte dem nicht so sein, müssen Hilfsgleise eingefügt werden.

Algorithmus 5 Prüfung des nächsten Gleises

Input: a_i, a_{i-1}
Output: Hilfskurve ja oder nein
1: Prüfe Richtungsänderung von a_{i-1} zu a_i
2: **if** Richtungsänderung nicht durch Kurve darstellbar **then**
3: Füge Hilfskurve ein bis Richtungsänderung durch Kurve darstellbar ist
4: **else**
5: Füge keine Hilfskurve ein
6: **end if**

Das Hinzufügen eines Schienensegments durch Algorithmus 6 ähnelt stark dem Hinzufügen des Startgleises durch Algorithmus 4. Der entscheidende Unterschied zwischen beiden liegt darin, dass der Schienentyp, die Position und die Rotation jetzt allesamt von der Position und Ausrichtung des vorherigen Gleises abhängig sind.

Algorithmus 6 Positionierung des nächstes Gleises

Input: G_{i-1}, a_i
Output: G_i
1: Lege zu verwendendes Schienensegment S_j fest
2: **if** $a_i = 0$ **then**
3: Lege Verschiebung in Richtung 0 um $l_i/2$ fest
4: **else**
5: Lege andere Verschiebungen fest
6: **end if**
7: Setze S_j auf e_{i-1} + der festgelegten Verschiebung
8: Rotiere S_j um Minimalrotation $\cdot\ a_{i-1}$
9: Endpunkt $e_i = e_{i-1} + l_i$ in Verschiebungsrichtung

3.3 Projektion des Gleissystems auf eine Ebene

Wie man in Abschnitt 2.5 gesehen hat, sucht ARCore mittels Kamerabild nach Ebenen in der Umgebung. Durch Berührung der Ebene, kann ein 3D-Modell an der getroffenen Stelle platziert werden. Eine Berührung wird durch einen Raycast ermittelt. Dabei wird, ausgehend von einer Touch-Berührung des Handydisplays, ein Strahl ausgesendet und überprüft wo dieser auftrifft. Der Auftreffpunkt des Strahls auf der Ebene dient als Ausgangspunkt des 3D-Modells. Die in dieser Arbeit verwendete Plattform Unity [19] stellt Funktionen zum Manipulieren von 3D-Modellen bereit. Das schließt eine Funktion zum Rotieren ein. Genaugenommen ließe sich dadurch eine Projektion umgehen, lediglich die Rotation des Modells muss beachtet werden. Zuvor haben wir zwei konkrete Ansätze herausgearbeitet, wie die Linie gezeichnet werden kann. In dem Fall, dass sie direkt auf der Ebene zu sehen sein soll, müsste das Gleissystem nicht mehr projiziert werden, weil dies bereits bei der Linie geschehen ist. Im anderen Fall, dass die Linie erst in einer 2D-Umgebung gezeichnet und daraus dann das Gleissystem generiert wird, könnte man es noch projizieren. Allerdings würde das einen unnötigen zusätzlichen Aufwand erfordern, zumal das Gleissystem stattdessen einfach mit einer Unity-Funktion rotiert werden kann. Somit kann auf eine tatsächliche Projektion des Gleissystems verzichtet werden, nur die Projektion der Linie steht noch zur Diskussion.

3.4 Platzierung eines Zuges auf dem Gleissystem

Für die Platzierung des Zuges auf dem Gleissystem gibt es drei zu verfolgende Ansätze. Beim ersten kann der Zug per Klick auf einen Button auf eine beliebige Schiene gesetzt werden. Die Rotation und die Position des Modells müssen entsprechend der Schiene angepasst werden. Position bedeutet hier einerseits, dass sich der Zug nicht in den Schienen befindet, sondern optisch ansprechend auf die Schienen gesetzt wird und andererseits, dass er der Länge nach an die Schiene angepasst wird. Bei einer längeren Geraden mag das nicht von Bedeutung sein, sollte der Zug sich aber in einer Kurve befinden, dann könnte es so aussehen als liege er nicht ganz auf den Schienen. Der Einfachheit halber kann der Zug auf die erste Schiene gesetzt werden, da diese eine Gerade ist. Beim zweiten Ansatz kann man mit einer Touch-Berührung auf die Ebene das nächstgelegene Gleis bestimmen und den Zug darauf platzieren, was abermals eine Überprüfung der Rotation und der Position zur Folge hat. Die letzte und sicher auch ansprechendste Lösung besteht darin, den Zug von einem Menüpunkt aus per Drag-and-drop auf einem der Gleise zu platzieren. Dabei könnte er noch während der Drag-Phase seine Rotation den unter ihm liegenden Gleisen anpassen. Im Gegensatz zum vorherigen Ansatz wird dem Anwender so schon eine Vorschau des Zuges präsentiert. Den Mehraufwand gilt es dabei abzuschätzen.

3.5 Fortbewegung des Zuges auf dem Gleissystem

Ist der Zug auf der Strecke platziert worden, dann soll er natürlich auch fahren können. Das Modell des Zuges soll dabei entlang der jeweiligen Gleise mittig in die korrekte Richtung bewegt werden, wobei der Zug einer vorgegebenen Linie folgen kann. Die ursprünglich gezeichnete Linie ist dafür allerdings ungeeignet, da sich ihr Verlauf durch die vorangegangene Glättung höchstwahrscheinlich von dem des Gleissystems unterscheidet. Eine weitere Möglichkeit besteht darin, die Fahrtrichtung von dem aktuell befahrenen Gleis abhängig zu machen. Zu diesem Zweck muss es allerdings möglich sein, die Entfernung des Zuges von einem Gleis zu bestimmen. Unity vereinfacht diese Möglichkeit, indem man dem Zug und alle Gleise mit einer Collider-Komponente versieht, deren Status bezüglich einer Überschneidung mit einem anderen Collider abgefragt werden kann. Außerdem soll das Modell des Zuges auch entsprechend der Fahrtrichtung rotiert werden. Am einfachsten wäre es, den Zug in die Richtung des aktuell befahrenen Gleises zu drehen, was leicht abrupt aussehen würde. Eine sanfte Kurvenfahrt ist durch ein Drehen über die Zeit möglich. Dabei müssen zusätzlich die Fahrtzeit des Zuges auf einem Kurvenstück ermittelt und die Rotation in das passende Verhältnis gesetzt werden.

4 Umsetzung

Als Plattform für diese Arbeit wird Android verwendet. Die Entwicklungsumgebung ist Unity3D. Zur Erkennung einer Ebene in der Umgebung wird die AR-Bibliothek ARCore von Google benutzt.

4.1 Skizzierung eines Gleisverlaufs

Direkt nach dem Start der App kann man in einem 2D-Bereich eine Linie für die spätere Gleisstrecke zeichnen. Auf das Bild der Kamera und die Projektion der Linie auf eine von ARCore erkannte Oberfläche wurde in diesem Teil verzichtet. Die intuitive Art zu zeichnen und ein Gefühl für die exakte Strecke zu haben überwiegen gegenüber der direkten Vorschau und der Möglichkeit potenzielle Hindernisse zu umkurven, auch vor dem Hintergrund, dass man die spätere Gleisstrecke mit absehbarem Mehraufwand mittels Touch-Geste verschieben, skalieren und rotieren können möchte. Legt der Anwender einen Finger auf den zum Zeichnen vorgesehenen Bereich und vollzieht er eine Bewegung, so wird ihr Ausgangspunkt zum Startpunkt p_0 der Linie und jeder weitere Punkt dementsprechend im Index um eins erhöht. Die Punkte werden dabei mittels Line Renderer von Unity durch eine Linie verbunden. Wenn beim Zeichnen ein weiterer Punkt hinzugefügt werden soll, dürfen zwischen dem letzten Punkt und der Position des Fingers eine Minimaldistanz nicht unterschritten und eine Maximaldistanz nicht überschritten werden. Der durchschnittliche Abstand zwischen Punkt und Finger wird dadurch ungefähr auf die Länge eines Gleissegments angepasst. Bewegt man sich auf der gezeichneten Linie zurück, lassen sich Punkte aus der Linie entfernen - die Linie wird sozusagen wegradiert. Sie wird automatisch geschlossen, wenn sich der Finger des Anwenders in die Nähe des Startpunkts p_0 begibt. Daraufhin ist der letzte Punkt gleich dem ersten. Somit wird die Linie, wie in 3.1 entworfen, als Linienzug, also als eine geordnete Liste von n miteinander verbundenen Punkten, $P = \{p_i\}$, $p_i \, \varepsilon \, \mathbb{R}^2$ mit $p_0 = p_{n-1}$ repräsentiert.

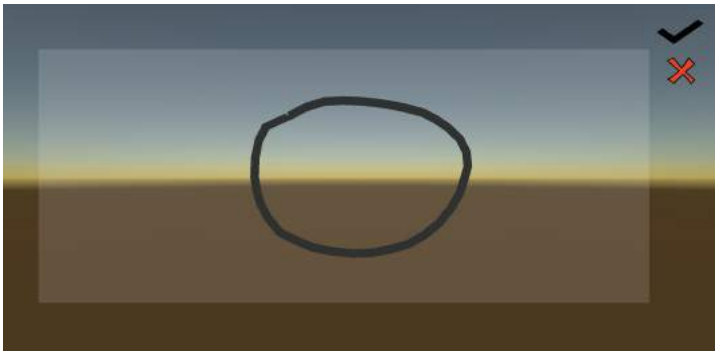

Abbildung 4.1: Ein in der 2D-Umgebung skizzierter Gleisverlauf

Abbildung 4.1 zeigt die zum Skizzieren verwendete Oberfläche mit einer geschlossenen Linie. Die Linie kann neu skizziert werden, indem auf das rote X gedrückt wird. Berührt man hingegen den schwarzen Haken, so wird aus der Linie ein Gleissystem generiert.

4.2 Generierung eines Gleissystems aus dem skizzierten Gleisverlauf

Der Satz von Gleissegmenten beinhaltet eine Gerade und je ein um 45 Grad nach links bzw. nach rechts gekrümmtes Gleisstück, im Folgenden kurz 45 Grad Kurve genannt. Dadurch wird die Möglichkeit verschiedene Strecken zu konzipieren nicht allzu stark eingeschränkt und der dafür nötige Aufwand liegt noch im Rahmen dieser Arbeit. Abbildung 4.2 zeigt die verwendeten Gleissegmente. Als Basis für die Schienen und später auch für den Zug dient das 3D-Model *Toy train* (Abbildung 4.3.) von Jair Trejo[7]. Für die 45 Grad-Kurven wurde die 90 Grad-Kurve des Basismodells zugeschnitten und gespiegelt.

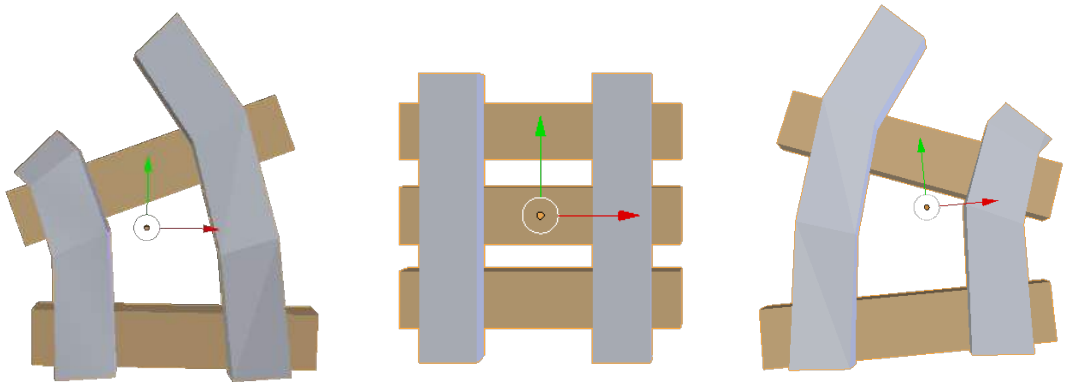

Abbildung 4.2: Der Satz von Gleissegmenten

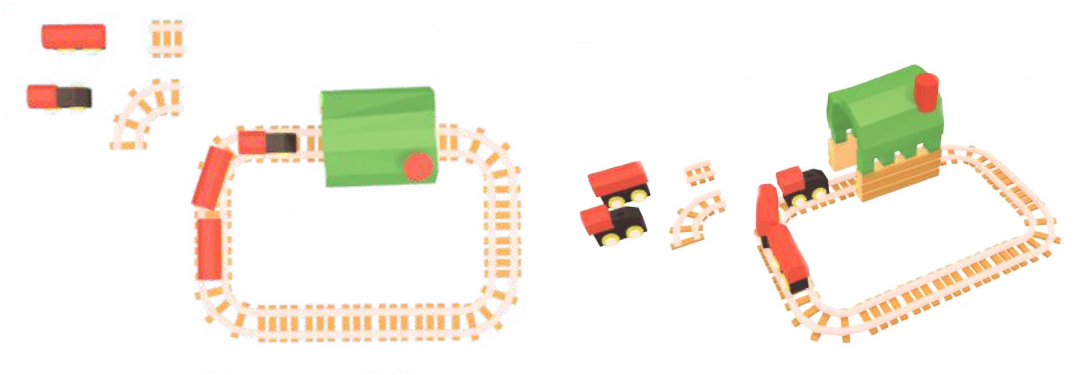

Abbildung 4.3: 3D-Basismodell der Schienen und des Zuges

[7]Toytrain von Jair Trejo, https://poly.google.com/view/2jFwyCzAAaP

Sobald sich der Anwender für eine Linie entschieden hat, wird einer Klasse “Track” die Liste der Punkte übergeben. Diese führt Algorithmus 2 aus Abschnitt 3.2 aus, der für jeden Punkt p_i, abhängig vom darauffolgenden Punkt p_{i+1}, eine Ausrichtung a_i bestimmt und diese für die jeweiligen Gleise speichert. Dabei wird die Differenz der x-Koordinaten und der y-Koordinaten beider Punkte ausgewertet, indem über die Variable “trackPrecision” ein Bereich für jede Richtung festgelegt wird. Da wir 45 Grad-Kurven haben, gibt es 8 Richtungen. Anders als im Konzept beschrieben, wird der Endpunkt eines Gleises nochmals um die Hälfte der Länge eines Schienensegments, also $l_i/2$, in die jeweilige Richtung verschoben, so dass bereits die nötige Verschiebung des darauffolgenden Gleises berücksichtigt wird und der tatsächliche Endpunkt nicht mit dem visuellen übereinstimmt. Diese Umsetzung sorgt dafür, dass nur noch die Rotation eines Gleissegments benötigt wird, um die Gleise aneinander zu rücken. Die Positionierung des Startgleises und der weiteren Gleise orientieren sich stark am Konzept.

Positionierung des ersten Gleises

Das Startgleis wird als Gerade festgelegt und auf den Punkt p_0 gesetzt. Daraufhin wird das Gleis in eine durch die Ausrichtung bestimmte Richtung verschoben und entsprechend rotiert. Für ungeradzahlige Richtungen, also für 1,3,5 und 7 muss wie in Abbildung 3.2 ersichtlich ein Gleissegment sowohl in x- als auch in y-Richtung bewegt werden, was rotationsbedingt eine geringere Verschiebung und eine zusätzliche Berücksichtigung zur Folge hat.

Positionierung der weiteren Gleise

Eine mögliche Richtungsänderung zum vorherigen Gleis ergibt sich aus der Differenz der Ausrichtungen a_i und a_{i-1}. Ist diese Null, dann wird eine Gerade verbaut, ist sie von Null verschieden, bestimmt ihr Vorzeichen, welche der beiden Kurven aus dem Satz von Schienensegmenten verbaut wird. Kontrollbedingungen für Sonderfälle, wie beispielsweise eine Richtungsänderung von 0 auf 7 oder von 0 auf 6, werden gesondert behandelt und Algorithmus 5 hinzugefügt. Sollte die Linie zu scharfe Kurven enthalten, so dass eine Richtungsänderung nicht mehr durch eine einzige Kurve überbrückt werden kann, werden so lange Hilfskurven verbaut, bis das wieder möglich ist. Die Hilfskurven werden dem Gleissystem wie die übrigen Gleise durch einen leicht abgewandelten Algorithmus 6 hinzugefügt. Die Position bzw. Ausrichtung entsprechen dabei dem Endpunkt e_{i-1} bzw. der Ausrichtung a_{i-1} des vorherigen Gleises G_{i-1}. Beim Ermitteln des Endpunkts e_i werden Kurven und Geraden unterschieden, da die benötigten Verschiebungen nicht übereinstimmen. Das Zuschneiden der 90 Grad Kurve des Basismodells hat dazu geführt, dass eine 45 Grad Kurve auf einer Seite nicht mehr optisch korrekt an ein anderes Gleissegment anschließen kann. Daher wurde bei Kurven zu der Position, nämlich dem Endpunkt des vorangehenden Gleises, noch eine zusätzliche Verschiebung ergänzt, um die Aneinanderkettung möglichst ansprechend zu gestalten, worauf man bei idealen Modellen verzichten könnte. In Abbildung 4.4 ist ein derart erstelltes Gleissystem zu sehen.

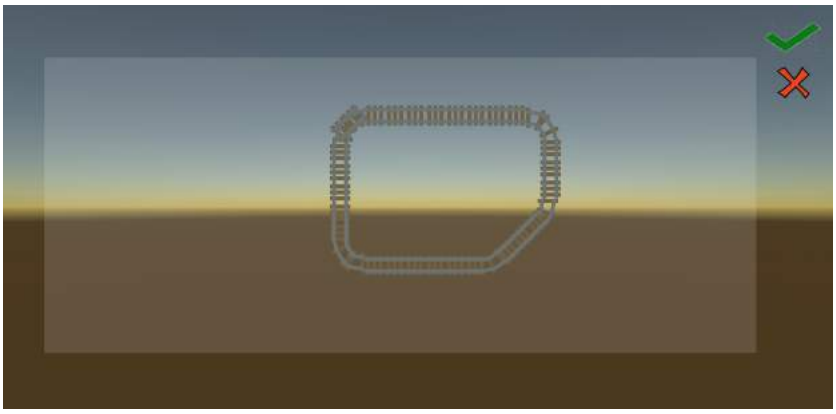

Abbildung 4.4: Gleissystem generiert aus dem Gleisverlauf aus Abbildung 4.1

4.3 Projektion des Gleissystems auf eine Ebene

Ist das gewünschte Gleissystem gefunden, wechselt man in den Kameramodus. ARCore sucht daraufhin in der Umgebung nach Oberflächen, auf denen unser Gleissystem dargestellt werden könnte. Wir verzichten auf die Projektion des Gleissystems, da Unity die Möglichkeit bietet 3D-Modelle als Ganzes zu rotieren. In Abbildung 4.5 ist eine Gleisstrecke zu sehen, die auf einer von ARCore erkannten Oberfläche liegt.

```
railwayObject.transform.rotation = Quaternion.Euler(90f,
                                   railwayObject.transform.rotation.eulerAngles.y,
                                   railwayObject.transform.rotation.eulerAngles.z);
```

Eine Rotation wird in Unity intern als Quaternion gespeichert [23]. Da ein Umgang mit Euler-Winkeln intuitiver ist, übergeben wir der Funktion "Quaternion.Euler" die Winkel, die ein Objekt mit den jeweiligen Achsen einnehmen soll, und lassen sie die Drehung durchführen. Wir rotieren das Gleissystem also um 90 Grad um die x-Achse und übergeben der Funktion die alten Winkel für die y- und z-Achse. Dadurch legen wir nach einer Touch-Berührung einer der gefundenen Ebenen, das Gleissystem dort praktisch ab. Zudem wird das Gleissystem ein Kindknoten des sogenannten Ankers der berührten Ebene. Dieser ist ein fester Punkt in der echten Welt und bewahrt das Gleissystem davor sich zu bewegen, falls ARCore die Oberfläche aktualisiert.

```
railwayObject.transform.parent = anchor.transform;
```

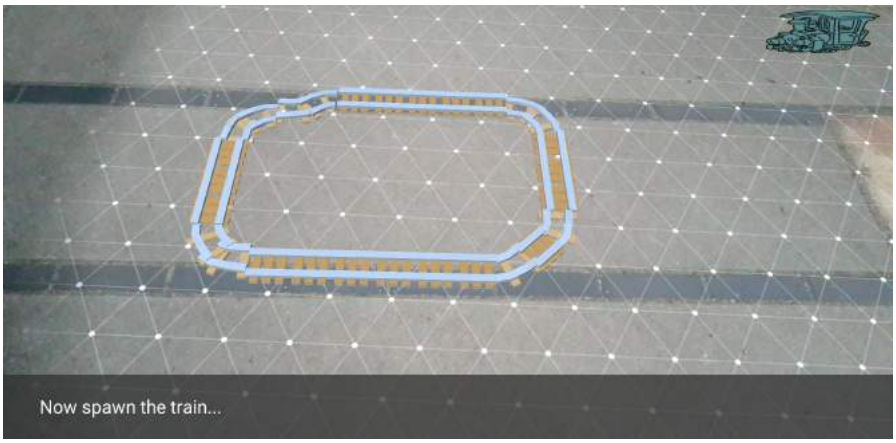

Abbildung 4.5: Eine generierte Gleisstrecke auf einer von ARCore erkannten Oberfläche. Zusätzlich erscheint ein Zugsymbol, um einen Zug auf die Strecke setzen zu können.

4.4 Platzierung eines Zuges auf dem Gleissystem

Nach der Platzierung des Gleissystems auf einer Ebene erscheint in der rechten oberen Ecke des Displays ein Zugsymbol (Abbildung 4.5). Drückt man darauf erscheint der Zug auf dem ersten Gleis wie in Abbildung 4.6 zu sehen ist. Dabei übernimmt er die Position und die Orientierung des ersten Gleisstücks. Lediglich die Höhe muss der des Gleises noch angepasst werden, damit der Zug auf den Schienen liegend erscheint. Zusätzlich wird auch er ein Kindknoten des Ankers.

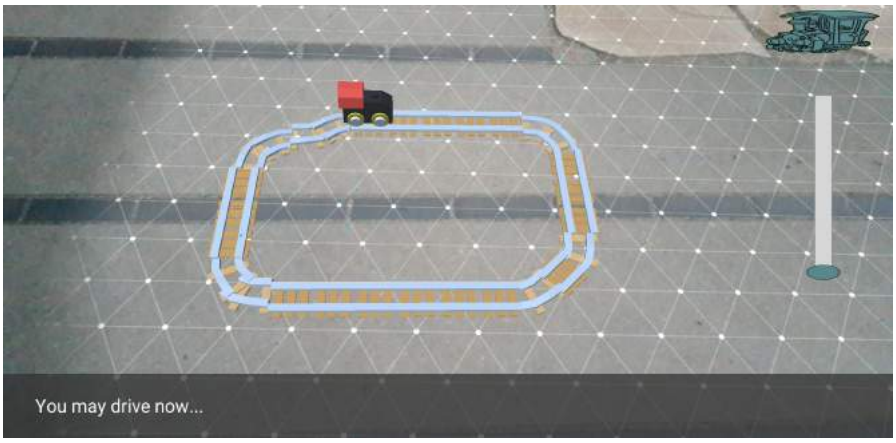

Abbildung 4.6: Ein Zug wurde auf das Startgleis der Gleisstrecke gesetzt.

4.5 Fortbewegung des Zuges auf dem Gleissystem

Zusammen mit dem Zug erscheint auf dem Display auch ein Geschwindigkeitsregler zum Fahren des Zuges. Schiebt man ihn nach oben, wird der Zug beschleunigt und fährt in Blickrichtung. Dabei ist die Geschwindigkeit abhängig von der Stellung des Reglers. Wie es aussehen kann wenn der Zug fährt wird von Abbildung 4.7 dargestellt.

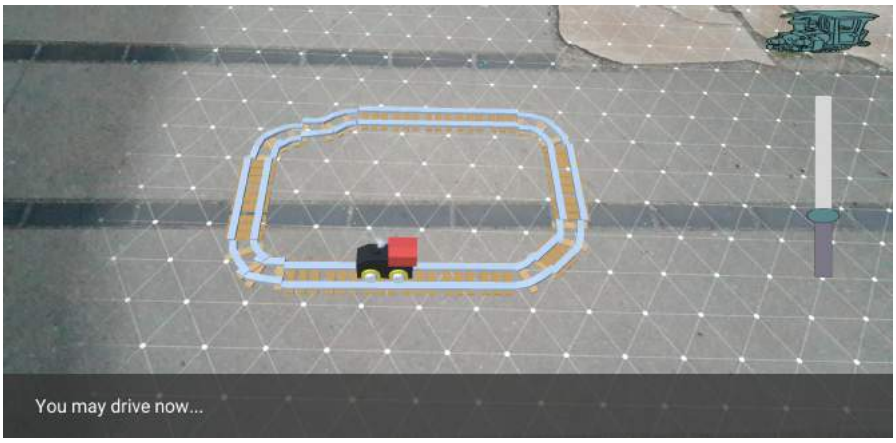

Abbildung 4.7: Ergänzend zur Bewegung visualisiert Dampf die Fahrt des Zuges.

```
Vector3 moveDirection = -transform.right;
transform.position = transform.position + (moveDirection * speed * slider * Time.deltaTime);
```

Unitys transform.right steht für die x-Achse im Weltkoordinatensystem. Intuitiver wäre transform.forward, die z-Achse. Allerdings ist das lokale Koordinatensystem eines Modells von der Ausrichtung abhängig, die in der Modelldatei definiert ist. Im Nachhinein wäre, auch bezüglich der Kurvensegmente, eine intensivere Beschäftigung mit der Bearbeitung von 3D-Modellen wünschenswert gewesen. Beim Befahren einer Kurve ändert sich die Blickrichtung des Zuges, welche die neue Richtung darstellt.

```
transform.Rotate(rotIdentity + rotMinimum * (newDirection - currentDirection));
```

Unity bietet die Möglichkeit, eine Collider Komponente an ein Objekt anzuhängen. Haben zwei Objekte einen Collider, dann lassen sich mit ihm die Zeitpunkte des Beginns und des Austritts aus einer Kollision sowie deren Dauer abfragen. Stellt der Collider des Zuges also den Beginn einer Überschneidung mit einer Kurve fest, so wird seine Orientierung geändert.

5 Evaluation

Sowohl das Konzept als auch die darauf aufbauende Umsetzung zur Entwicklung einer AR-Modelleisenbahn werden in diesem Kapitel begutachtet. Darauf aufbauend werden Weiterentwicklungsmöglichkeiten dargestellt.

5.1 Evaluation des Konzepts

In einigen Teilbereichen des Kozepts wurden bereits unterschiedliche Ansätze einer Umsetzung erörtert und benötigen keiner weiteren Evaluierung. Aus diesem Grund wird in diesem Abschnitt speziell auf die Generierung eines Gleissystems eingegangen.

Der skizzierte Gleisverlauf wird durch eine geordnete Liste miteinander verbundener Punkten beschrieben. Des Weiteren wird jedem Punkt eine der acht beschriebenen Ausrichtungen (Abbildung 3.2) zugeordnet, die anhand des Winkels einer Geraden vom aktuellen Punkt zum nachfolgenden Punkt beschrieben und mithilfe derer das Gleissystem zusammengestellt wird. Das hat zur Folge, dass der Verlauf der Strecke vorbestimmt und keine zusätzlichen Anpassungen an den Gleisverlauf mehr möglich sind. Die Abstände der Punkte auf der Linie entsprechen in etwa der Länge eines Schienensegmentes. Einerseits führt dies dazu, dass das Gleissystem dem Gleisverlauf sehr ähnlich ist. Anderseits vermindert es die Flexibilität, falls Schienensegmente unterschiedlicher Länge vorhanden sein sollen.

Außerdem kann das Gleissystem stark vom gezeichneten Gleisverlauf abweichen, wenn der Winkel des gezeichneten Streckenabschnitts nahe der Übergangsgrenzen zwischen zwei Ausrichtungsbereichen liegt (Abbildung 3.1). Mittels einer Abstandsfunktion, die Punktweise den Abstand des Gleissystems vom gezeichneten Gleisverlauf berechnet, könnte bei Überschreitung eines Grenzwertes nachkorrigiert und eine höhere Ähnlichkeit erreicht werden. Des Weiteren würde eine Abstandfunktion verschiedene Abfragebedingungen unnötig machen. So wäre es in der Umsetzung z.B. nicht mehr notwendig die Übergänge in der Ausrichtung wie von 0 auf 7 oder das Einfügen von Hilfsgleisen zusätzlich zu behandeln, da mit jeder Iteration einfach das optimale Schienensegment ausgemacht werden kann, damit sich die Strecke wieder dem Verlauf anpasst.

Für die Positionierung und Ausrichtung der Gleise eignen sich zudem Transformationsmatrizen, die unter Umständen leichter zu handhaben und übersichtlicher als die im Konzept beschriebene Variante sind. Beide Varianten bewirken im Grunde genommen das Gleiche, aber spätestens im Dreidimensionalen sollten Transformationsmatrizen verständlicher sein, da diese die benötigten Informationen für die korrekte Positionierung und Ausrichtung kompakt darstellt, anstatt beispielsweise eine Transformation als Abfolge jedes Zwischenschritts aufzuschreiben.

5.2 Evaluation des Prototyps

In diesem Abschnitt wird der Prototyp anhand seiner Funktionalität ausgewertet. Das Erkennen einer Ebene in der Umgebung bzw. die Projektion des Gleissystems auf diese Ebene werden größtenteils von der Bibliothek ARCore bzw. der Platform Unity erfüllt und sind somit nicht Gegenstand der Beurteilung. Es verbleiben die Einschätzungen zum Skizzieren eines Gleisverlaufs, zur Generierung des Gleissystems und zum Fahren des Zuges.

Auf dem Display des Smartphones lässt sich ohne weiteres mit einem Finger eine Linie zeichnen. Es kann jedoch zu Problemen führen, wenn der Anwender seinen Finger beim Zeichnen zu schnell über das Touch-Display bewegt. Wird die maximal zulässige Distanz zwischen dem letzten gespeicherten Punkt und der aktuellen Position des Fingers vom Anwender überschritten, dann wird das Zeichnen der Linie beendet. Die Anwendung einer Abstandsfunktion hätte einen großzügigeren Maximalabstand zugelassen und so den möglichen Abbruch des Zeichenvorgangs vermieden.

Aus dem gezeichneten Gleisverlauf lässt sich direkt das Gleissystem ableiten, jedoch schließen Kurven, wie in 4.2 erwähnt, auf einer Seite nicht optimal an ein anderes Gleisstück an. Die Ähnlichkeit des Gleissystems zum Gleisverlauf ist zwar gut erkennbar, allerdings wird das Gleissystem nicht korrekt geschlossen. Es kann passieren, dass das Ende des Gleissystems nicht an den Start anschließt und teilweise noch eine ungewollte Kurve nimmt oder das Ende über den Start hinausgeht und einen Teil der Strecke überlagert. Daher sind die Anforderungen an das Generieren des Gleissystems nicht erfüllt und Bedarf der Optimierung.

Der Zug wird passend auf dem Startgleis platziert und kann mithilfe eines Geschwindigkeitsreglers beschleunigt werden, woraufhin er in Blickrichtung fährt. Bei Kurven dreht sich der Zug komplett um 45 Grad, was etwas abrupt erfolgt und den Zug mitunter leicht von den Schienen rückt bis er unter Umständen komplett von der Bahn abkommt. Ein stetige Kurvenfahrt, perfekt angepasst an eine Kurve, oder das Abfahren einer vorgegebenen Linie hätten zu einem besseren Ergebnis geführt.

5.3 Weiterentwicklung

Für den entwickelten Prototyp sind viele Erweiterungen denkbar. Ein Ansatz könnte sein, den Prototyp um zahlreiche Komponenten einer typischen Zugstrecke zu ergänzen. Bahnhöfe und Tunnel würden das Spielerlebnis schon mit wenig Mehraufwand steigern. Dafür sind lediglich passende 3D-Modelle nötig und die Möglichkeit die Komponenten selbstbestimmt zu platzieren. Eine zufällige Platzierung ist zwar möglich, aber dem Spielerlebnis weniger zuträglich. Daher wäre eine Art Baueditor, in der der Anwender aus einer Reihe von Elementen wählen und diese per Drag-and-drop auf oder neben der Zugstrecke platzieren kann, vorstellbar. Dieser könnte sowohl zur späteren Bearbeitung der Strecke als auch direkt zu Beginn, alternativ zum Zeichnen einer Linie, eingesetzt werden. Weichen, Ampeln, Waggons, zusätzliche Zugtypen und Landschaftselemente wie Windmühlen, Bäume und Gewässer sind weitere Elemente die ein Baueditor enthalten könnte. Simpel animierte Tiere sowie Hubschrauber oder Flugzeuge, die sich in einem gewissen Radius um die Strecke bewegen, würden die Szenerie zusätzlich lebendiger machen. Mit zunehmender Kulisse ist eine Zugfahrt aus der Sicht des Lokführers denkbar. Allerdings beschränkte sich das nur auf eine rein virtuelle Fahrt, weil die Idee mit nur einer Handykamera nicht realisierbar ist.

Eine weitere Möglichkeit ist, die Umsetzung nicht nur auf eine Modelleisenbahn zu beschränken, sondern die gezeichnete Linie optional in eine Rennstrecke mit Rennauto oder ähnlichem zu verwandeln. Auch hierfür ist der Aufwand gering, da im Grunde genommen nur 3D-Modelle ausgetauscht werden müssen. Ein anderer Ansatz wäre, den Prototypen um Lerninhalte zu erweitern. Elemente könnten bei näherer Betrachtung oder beim Anwählen allgemeine Informationen liefern. Beispielsweise könnte beim Anwählen des Zuges ein Fenster erscheinen, das Angaben zum Gewicht und der Größe realer Modelle macht und einen Link zu weiteren Informationen enthält. Des Weiteren ließen sich visuelle und akustische Effekte, wie das Bremsen der Eisenbahn, hinzufügen. Im Großen und Ganzen muss man sagen, dass die Möglichkeiten nahezu endlos sind.

6 Zusammenfassung und Ausblick

Zusammenfassung

Ziel dieser Arbeit war es eine AR-Modelleisenbahn zu entwickeln. Dazu wurde Googles Augmented Reality-Plattform ARCore verwendet, wodurch das Programm für Android-Geräte mit der nötigen technischen Spezifikation verfügbar wird.

Zunächst wurde der technische Rahmen in Bezug auf Augmented Reality und ARCore beleuchtet. Auf dieser Grundlage konnte aus der Idee eine AR-Modelleisenbahn zu entwickeln ein Konzept erstellt werden, für dessen Details potenzielle Umsetzungen diskutiert wurden. Im Anschluss daran wurde die Umsetzung des Konzepts in Form einer prototypischen Implementierung vorgestellt und dabei die Entscheidung für eine der möglichen Realisierungen erklärt. Die Arbeit endete mit einer kritischen Hinterfragung des vorgestellten Konzepts und seiner Realisierung sowie mit der Darlegung möglicher Erweiterungen.

Ausblick

Augmented Reality ist ein stark wachsendes Feld, nicht nur in der Industrie, sondern auch in privaten Haushalten. Obgleich Augmented Reality noch vergleichsweise jung ist, gibt es schon heute etliche Anwendungsfälle für sie und man kann nur schwer erahnen, wie viel Potenzial tatsächlich in ihr steckt. Sicher ist nur, dass dieses immens groß ist. Von einem AR-Klassenzimmer mit unterstützenden Koordinatensystemen, die das Verständnis für Mathematik erleichtern, bis zum Operationssaal, in dem Ärzten bei schwierigen Eingriffen alle wichtigen Informationen bereitgestellt werden, wird schon in vielerlei Richtung geforscht. Auch im Hinblick auf die Fülle möglicher Erweiterungen sind AR-Ausstellungen oder AR-Themenparks, ähnlich wie bei Hamburgs Miniatur Wunderland mit herkömmlichen Modelleisenbahnen und Modellspielzeugen, in Zukunft möglich. Erste Anfänge dafür bieten schon heutige Freizeitparks mit 4D-Kinos und den dort angebotenen Reisen durch virtuelle Welten. Angesichts der Verfügbarkeit von AR-Bibliotheken wie Googles ARCore für Android oder ARKit für iOS, gepaart mit der Käuflichkeit AR-fähiger Geräte, steigt die Anzahl der AR-Interessierten und AR-Entwickler fortwährend stark an. Das hat zur Folge, dass sich Augmented Reality in Zukunft noch schneller weiterentwickeln wird als bisher. Abschließend darf behauptet werden, dass Augmented Reality auf die eine oder andere Weise wohl auf das Leben eines Jeden von uns Einfluss nehmen und letztlich nicht mehr wegzudenken sein wird.

Literatur

[1] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. *Augmented reality: a class of displays on the reality-virtuality continuum*. 1995.

[2] Statistische Bundesamt. Einkommen, konsum, lebensbedingungen. Verfügbar unter: https://www.destatis.de/DE/Publikationen/StatistischesJahrbuch/EinkommenKonsumLeben.pdf, 2018.

[3] BVS Handelsverband Spielwaren. Marktdaten. Verfügbar unter: http://www.bvspielwaren.de/News/Marktdaten.php, 2018.

[4] D Wagner, T. Pintaric, F. Ledermann, and D. Schmalstieg. Towards massively multi-user augmented reality on handheld devices. 2005.

[5] Google LLC. Arcore. Verfügbar unter: https://play.google.com/store/apps/details?id=com.google.ar.core&hl=de, 2018.

[6] Ronald T. Azuma. *A Survey of Augmented Reality*. Malibu, 1997.

[7] Lothar Steiger. *Augmented Reality : Theorie und Praxis*. München, 2011.

[8] G Klein and T. Drummond. Robust visual tracking for non-instrumented augmented reality. 2003.

[9] M. Tonnis, C. Sandor, G. Klinker, C. Lange, and H. Bubb. Experimental evaluation of an augmented reality visualization for directing a car driver's attention. 2005.

[10] J. Rolland, Y. Baillot, and A. Goon. *A Survey of Tracking Technology For Virtual Environments*. Orlando, 2001.

[11] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. 1999.

[12] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. 2008.

[13] Marcus Tönnis. *Augmented Reality : Einblicke in die Erweiterte Realität*. Berlin, Heidelberg, 2010.

[14] Alan B. Craig. *Understanding Augmented Reality : Concepts and Applications*. Amsterdam u.a., 2013.

[15] Alan H. Watt. *3D Computer Graphics*. Pearson, Harlow u.a., 3. edition, 2000.

[16] John Vince. *Rotation Transforms for Computer Graphics*. London, 2011.

[17] Todd A. Ell. *Quaternion fourier transforms for signal and image processing*. London s.l., 2014.

[18] Prof. Stefan Schlechtweg. *Computergraphik Grundlagen Skript der Hochschule Anhalt*. 2010.

[19] Unity Technologies. Unity3d.
Verfügbar unter: https://unity3d.com/, 2018.

[20] Google LLC. Google developers.
Verfügbar unter: https://developers.google.com/ar/, 2018.

[21] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 2015.

[22] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981.

[23] Unity Technologies. Unitydocs.
Verfügbar unter: https://docs.unity3d.com/Manual/index.html, 2018.