Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Ruben Christian Buhl

Evaluation des Potenzials von Augmented Reality für interaktive Brettspiele

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Ruben Christian Buhl

Evaluation des Potenzials von Augmented Reality für interaktive Brettspiele

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: Prof. Dr. Michael Schäfers

Eingereicht am: 30. Mai 2018

Ruben Christian Buhl

Thema der Arbeit
Evaluation des Potenzials von Augmented Reality für interaktive Brettspiele

Stichworte
Augmented Reality, Brettspiel, Tracking, iOS, SceneKit, VisionLib, OpenCV, Aruco

Kurzzusammenfassung
Diese Ausarbeitung beschäftigt sich mit der Frage, inwiefern Brettspiele von Augmented Reality profitieren können und welche Tracking-Verfahren dabei in Frage kommen. Dabei wurden verschiedene vorhandene Spielkonzepte betrachtet, mit den bereits unterschiedliche Ansätze untersucht wurden. Anschließend wurde ein prototypisches Brettspiel konzipiert und entwickelt. In diesem Prototypen wurde sowohl das Tracking einer zweidimensionalen Oberfläche, als auch eines dreidimensionalen Modells angewandt.
Ruben Christian Buhl

Title of the paper
Evaluation of the potential of Augmented Reality for interactive board games

Keywords
Augmented Reality, Board Game, Tracking, iOS, SceneKit, VisionLib, OpenCV, Aruco

Abstract
This work deals with the question of how board games can benefit from Augmented Reality and which tracking methods are suitable. In doing so, various existing game concepts were considered, with which already different approaches were examined. Subsequently, a prototypical board game was designed and developed. In this prototype, both the tracking of a two-dimensional surface and a three-dimensional model were applied.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1

2 Grundlagen **3**
2.1 Augmented Reality . . . 3
2.1.1 Definition . . . 3
2.1.2 Video See-Through . . . 3
2.1.3 Optisches See-Through . . . 3
2.1.4 Head Mounted Displays . . . 4
2.1.5 Handheld-Geräte . . . 4
2.1.6 Anwendungsfälle . . . 4

3 Stand der Technik **5**
3.1 Related Work . . . 5
3.1.1 Monopoly . . . 5
3.1.2 BattleBoard 3D . . . 6
3.1.3 TARBoard . . . 6
3.1.4 Quest: Zeit der Helden . . . 6
3.1.5 Art of Defense . . . 7
3.2 Augmented Reality Games . . . 8
3.2.1 Kommerzielle Brettspiele . . . 8

4 Konzept **10**
4.1 Game Design . . . 10
4.2 Use Cases . . . 11
4.2.1 Spielerstellung . . . 11
4.2.2 Spielablauf . . . 13
4.3 Anforderungen . . . 14
4.3.1 Funktionale Anforderungen . . . 14
4.3.2 Nicht funktionale Anforderungen . . . 16

5 Umsetzung **17**
5.1 Plattform . . . 17
5.1.1 iOS . . . 17
5.1.2 Swift und Objective-C . . . 17
5.2 Tracking Libraries . . . 18
5.2.1 VisionLib . . . 18

5.2.2 OpenCV . . . 19
5.2.3 Aruco . . . 19
5.2.4 Alternativen . . . 22
5.3 Software . . . 23
5.3.1 Xcode . . . 23
5.3.2 Multipeer Connectivity . . . 24
5.3.3 SceneKit . . . 24
5.3.4 GameplayKit . . . 25
5.3.5 Blender . . . 27
5.4 Editor . . . 28
5.5 Spielfigur . . . 29
5.5.1 Format . . . 30
5.5.2 Ausdruck in 3D . . . 30
5.5.3 Ausdruck auf Papier . . . 30
5.6 Spielbrett . . . 31
5.6.1 Format . . . 31
5.6.2 Ausdruck . . . 32

6 Evaluation **33**
6.1 Funktionalität . . . 33
6.1.1 Tracking des Aruco Board . . . 33
6.1.2 Tracking der Spielfiguren . . . 34
6.2 Nutzerstudie . . . 35
6.2.1 Beobachtung . . . 35
6.2.2 Auswertung . . . 35

1 Einleitung

Brettspiele erfreuen sich heutzutage noch immer großer Popularität. Trotz der seit Jahrzehnten wachsenden Präsenz von Videospielen, sind die Brettspiele noch nicht vom Markt verschwunden. Es gibt sogar Anzeichen, die das Gegenteil vermuten lassen. Vor Allem komplexere Brettspiele mit höherem Schwierigkeitsgrad scheinen an Beliebtheit zu gewinnen. Der Verein Spiel des Jahres verleiht, erst seit dem Jahr 2011, auch den Preis "Kennerspiel des Jahres"[1]. Damit werden gezielt Spiele ausgezeichnet, die für erfahrene Spieler gedacht sind.
Derartig anspruchsvolle Spiele bringen natürlich auch umfangreichere Regelwerke mit sich. Sodass einige Brettspiele sogar mehrere Stunden Zeit erfordern, bis alle Spieler die Spielregeln verstanden haben und die erste Partie begonnen werden kann. Wie zum Beispiel bei "Robinson Crusoe: Abenteuer auf der verfluchten Insel"[2] oder "Civilization: Das Brettspiel"[3].
Leider werden viele potenzielle Spieler von diesem hohen Lernaufwand abgeschreckt. Die Vermittlung eines komplexen Regelwerkes kann in Videospielen häufig eleganter gelöst werden. Oft geschieht dies durch eine Reihe von Einführungsmissionen. In diesen Missionen können die Spielmechaniken schrittweise erweitert werden. Dadurch lernt der Spieler eine Regel nach der anderen und muss nicht sofort mit einem dicken Regelbuch konfrontiert werden. Stattdessen kann das Spiel mit dem Spieler kommunizieren und Hinweise geben. Fehlerhafte Spielzüge werden vom Spiel erkannt oder sind gar nicht erst möglich.

1.1 Motivation

Die These, Brettspiele seien geselliger als Videospiele, lässt sich nur schwer widerlegen. Als Gruppe um ein Spielbrett zu sitzen und über den nächsten Spielzug zu grübeln, hat doch einen gewissen Charme. Zumal sich dabei mehr Gelegenheiten bieten seinen Mitspielern in die

[1] Internetseite der Kritikerpreise von Spiel des Jahres:
http://www.spiel-des-jahres.com/de/preise (30. Mai 2018)
[2] Produktseite von Robinson Crusoe: Abenteuer auf der verfluchten Insel:
http://www.pegasus.de/detailansicht/51945g-robinson-crusoes-vermaechtnis/ (30. Mai 2018)
[3] Produktseite von Civilization: Das Brettspiel:
https://www.fantasyflightgames.com/en/products/civilization/ (30. Mai 2018)

Augen zu blicken, als es bei typischen Videospielen der Fall ist.
Die Vorteile von Videospielen will man jedoch nicht leichtfertig aufgeben. Daher stellt sich die Frage, wie sich die Vorteile von Videospielen in ein Brettspiel integrieren lassen. Augmented Reality ist vielleicht die Antwort. Die Entwicklung von Datenbrillen ist inzwischen so weit vorangeschritten, dass bereits eine Auswahl an Geräten auf dem Markt ist. Wie zum Beispiel die Mircosoft HoloLens[4] oder die Meta 2[5]. Die Möglichkeit ein Brettspiel mit beliebigen digitalen Inhalten zu erweitern, ist daher durchaus denkbar.
Durch die Datenbrille könnten die Spieler laufend Hinweise dazu erhalten, was sie als nächstes zu tun haben oder sogar wie das Spielfeld aufgebaut werden muss. Neben dieser virtuellen Anleitung, werden durch die computerisierte Unterstützung, noch ganz andere Szenarien möglich. Beispielsweise kann der Computer, als neutrale Instanz, einen virtuellen Schiedsrichter, Spielleiter oder auch Mitspieler verkörpern. So müssten verschiedene organisatorische Aufgaben nicht mehr von den Spielern selbst erledigt werden. Wie zum Beispiel Karten mischen oder Punktestände aktualisieren. Auch die Steuerung feindlicher Spielfiguren könnte dann intelligenter funktionieren. Ohne dass die Figuren nur primitiven Markierungen auf dem Spielbrett folgen oder ein menschlicher Spieler die Rolle des Bösen übernehmen muss. Aber die vielleicht wichtigste Bereicherung für das Spielerlebnis, könnten die vielen visuellen Effekte darstellen, die man mit Augmented Reality ergänzen würde. Animationen an den richtigen Stellen können eine Spielwelt sehr viel lebendiger erscheinen lassen.

[4] Produktseite der HoloLens: https://www.microsoft.com/de-de/hololens (30. Mai 2018)
[5] Produktseite der Meta 2: http://www.metavision.com (30. Mai 2018)

2 Grundlagen

2.1 Augmented Reality

In diesem Abschnitt werden grundlegende Begrifflichkeiten von Augmented Reality diskutiert.

2.1.1 Definition

Unter Augmented Reality versteht man die unmittelbare Erweiterung der Realitätswahrnehmung. Wobei die reale Umgebung mit virtuellen Inhalten angereichert wird. Diese virtuell erweiterten Inhalte richten sich in Echtzeit an realen Objekten aus.
Laut Azuma (1997) sind drei Kriterien ausschlaggebend. Es werden Realitat und Virtualität kombiniert, Interaktionen sind in Echtzeit möglich und es werden dreidimensionale Objekte erfasst, um Erweiterungen anzubringen.
Von Milgram u. a. (1995) wird der Übergang von Realitat zu Virtualität genauer betrachtet. Mixed Reality bezeichnet alles was sich zwischen realer und virtueller Umgebung befindet. Augmented Reality wird dabei ein tendenziell größerer realer Anteil zugeordnet. Im Gegensatz zu Augmented Virtuality, wo der virtuelle Anteil stärker ist.

2.1.2 Video See-Through

Von Dörner u. a. (2014) werden verschiedene Herangehensweisen zur Erweiterung der Realität aufgezeigt.
Bei der als Video See-Through bezeichneten Technik, wird das Bild der Umgebung mit einer Videokamera aufgezeichnet. Anschließend wird das Videobild mit den virtuellen Inhalten überlagert und auf dem Ausgabegerät ausgegeben.

2.1.3 Optisches See-Through

Im Gegensatz zum Video See-Through, ist beim optischen See-Through eine Videoaufnahme nicht zwingend erforderlich. Stattdessen wird die reale Umgebung von dem Benutzter direkt wahrgenommen. Sodass lediglich die virtuellen Inhalte vom Ausgabegerät wiedergegeben

werden müssen. Dafür ist ein semi-transparentes Display erforderlich, damit der Benutzer die dahinter liegende Realität sehen kann, während nur die virtuellen Erweiterungen auf dem Display eingeblendet werden.

2.1.4 Head Mounted Displays

Head Mounted Displays werden, wie es der Name schon vermuten lässt, auf dem Kopf des Benutzers getragen. Im Kontext von Augmented Reality versteht man darunter eine Datenbrille, die optisches See-Through umsetzt.

2.1.5 Handheld-Geräte

Mit Handheld-Geräten werden insbesondere Smartphone und Tablets assoziiert, die heutzutage alle Voraussetzungen erfüllen, um Video See-Through zu ermöglichen. Die rückseitige Kamera filmt die reale Umgebung und die enthaltene Hardware ist stark genug, um die Position der virtuellen Inhalte zu berechnen und auf dem Display das erweiterte Bild wiederzugeben.

2.1.6 Anwendungsfälle

Das Spektrum an möglichen Anwendungen wächst stetig. Zum Beispiel kann Augmented Reality zu Schulungszwecken in der Industrie verwendet werden. Alle nötigen Arbeitsschritte werden dann direkt in das Sichtfeld des Lehrlings eingeblendet. Dazu benötigt der Lehrling nur die Datenbrille und keine umfangreichen Dokumentationsunterlagen. Die Firma ioxp[1] bietet bereits derartige Lösungen an. Eine entsprechende Implementierung findet man bei der Firma Wilo[2].
Ein weiteres Anwendungsszenario für Augmented Reality findet man in der App IKEA Place[3]. Die App ermöglicht Möbelstücke von IKEA direkt in die Umgebung zu projizieren. Damit soll der Kunde besser beurteilen können, wie ein Produkt in seine eigenen Räumlichkeiten passt.

[1] Internetseite der Firma ioxp: http://www.ioxp.de (30. Mai 2018)
[2] Bericht über ioxp Implementierung bei Wilo: https://youtu.be/iz7nujZuCxA (30. Mai 2018)
[3] IKEA Place im App Store:
https://itunes.apple.com/de/app/ikea-place/id1279244498?mt=8 (30. Mai 2018)

3 Stand der Technik

3.1 Related Work

Das Spektrum an Geräten, die zur Erweiterung der Realität dienen können, ist sehr vielfältig. Dadurch ergeben sich auch viele verschiedene Möglichkeiten, auf welche Art und Weise ein Brettspiel von Augmented Reality profitieren kann. Dabei kann sich auch das Verhältnis von realen zu virtuellen Anteilen, von Spiel zu Spiel, stark unterscheiden.
Einige Herangehensweisen zur Integration von Augmented Reality in Brettspielen wurden in der Vergangenheit bereits untersucht. Eine Auswahl davon wird im Folgenden präsentiert. Begonnen wird mit einem Beispiel, wie ein bereits existierender Brettspielklassiker visuell erweitert werden kann, ohne dabei in das Regelwerk einzugreifen.

3.1.1 Monopoly

Molla und Lepetit (2010) haben das Brettspiel Monopoly[1] so erweitert, dass verschiedene virtuelle Avatare auf die realen Spielfiguren projiziert werden. Außerdem werden die Häuser und Hotels ebenfalls auf das Spielbrett projiziert.

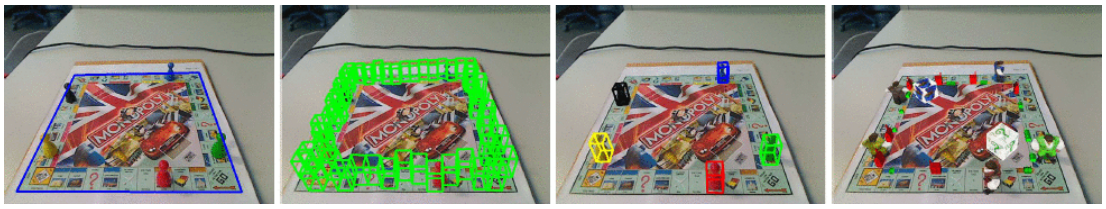


Abbildung 3.1: Tracking-Prozess von Molla und Lepetit (2010)

Zur Observierung des Spielbrettes dient eine einfache Webcam. Die klassischen Spielfiguren wurden gegen einfachere eingetauscht. Sodass alle Spielfiguren die gleiche Form haben und

[1] Produktseite von Monopoly:
https://www.hasbro.com/de-de/product/monopoly-classic-game:7EABAF97-5056-9047-F577-8F4663C79E75 (30. Mai 2018)

sich nur in der Farbe unterscheiden. Die Erkennung der Spielfiguren ist auf die Bereiche der validen Spielfelder beschränkt. So wurde ausreichend Rechenleistung eingespart, um die Spielfiguren aller Spieler gleichzeitig zu erkennen.

3.1.2 BattleBoard 3D

BattleBoard 3D von Andersen u. a. (2004) ist ein Brettspiel, das ähnlich wie Schach aufgebaut ist. Zwei Spieler duellieren sich mit Spielfiguren auf einem karierten Spielbrett. Ziel des Spiels ist es, die Truhe des Gegners zu erobern.
Einer der Spieler trägt eine prototypische Datenbrille, während der andere Spieler das Spielgeschehen auf einem Monitor verfolgen muss. Der Monitor zeigt das erweiterte Bild einer zusätzlichen Webcam. Spielfiguren und Truhen sind virtuell und werden erst auf dem Monitor, beziehungsweise in der Datenbrille, sichtbar. Die realen Gegenstücke der Spielfiguren sind lediglich durch bewegliche Marker repräsentiert.

3.1.3 TARBoard

Die Sichtbarkeit von Markern kann die Immersion eines Spiels sehr stören. In dem System TARBoard von Lee u. a. (2005) sind diese Marker versteckt. Für TARBoard wurde ein Kartenspiel für zwei Spieler implementiert. Hinter jeder Karte verbirgt sich eine Kreatur. Ziel des Spiels ist es, alle Kreaturen des Gegners zu besiegen. Die Kämpfe der Kreaturen werden in der Mitte des Spielfeldes animiert. Die Marker zur Identifikation der Karten sind auf den Kartenrücken abgebildet.
TARBoard verwendet zum Erkennen der Karten und zum Erweitern des Spielgeschehens zwei separate Kameras. Gespielt wird auf einem Glastisch. Unter dem Tisch befindet sich ein Spiegel, worin sich die untere Seite des Spielfeldes spiegelt. So ist das gesamte Spielfeld für die erste Kamera, die ebenfalls unter dem Tisch positioniert ist, sichtbar. Karten die nun aufgedeckt auf den Glastisch gelegt werden, können durch den Marker auf der Rückseite, von der ersten Kamera erkannt werden. Die zweite Kamera ist oberhalb des Tisches positioniert. Sie ist für die Aufnahme des Bildes zuständig, in dem der Spielinhalt erweitert wird.
Durch diesen Aufbau sind die Marker aus dem Blickfeld der Spieler verschwunden. Außerdem können die Marker so nicht mehr versehentlich durch die Spieler verdeckt werden.

3.1.4 Quest: Zeit der Helden

Eine modernere Möglichkeit die Marker zu verstecken wäre vielleicht die Verwendung eines Multi-Touch-Tisches. So ein Gerät haben Giebler-Schubert u. a. (2013) verwendet um das Brett-

spiel "Quest: Zeit der Helden"[2] mit Augmented Reality zu anzureichern. Der Multi-Touch-Tisch übernimmt dabei die Kontrolle und Auswertung verschiedener Spielmechaniken. Wie zum Beispiel das Abmessen von Distanzen. Außerdem werden visuelle und auditive Effekte ergänzt, die das Spielgeschehen unterstreichen.
Zusätzlich verfügt jeder Spieler über ein Tablet-PC. Dort kann jeder Spieler bestimmte Informationen einsehen oder Eingaben tätigen. Möglich ist auch der Zugang zu geheimen Informationen, die nur für einen bestimmten Spieler verfügbar sein sollen.
Geheime Informationen müssen in konventionellen Brettspielen häufig einfacher gehalten werden. Ansonsten ließen sie sich nur äußerst schwierig in analoger Form repräsentieren. Wie zum Beispiel in Form von Karten.

3.1.5 Art of Defense

Bei Art of Defense von Huynh u. a. (2009) wurde Augmented Reality sehr vielseitig angewandt. Dieses Spiel wäre als Brettspiel ohne Augmented Reality kaum vorstellbar. Art of Defense ist ein kooperatives Spiel für zwei Personen und kann dem Genre Tower Defense[3] zugeordnet werden. Zur Erweiterung der Realität werden Mobiltelefone mit Kameras genutzt.

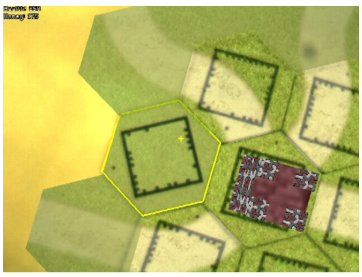


Abbildung 3.2: Spielfeld zu Art of Defense von Huynh u. a. (2009)

Ziel des Spiels ist es ein virtuelles Gebäude in der Mitte des Spielfelds gegen herannahende Feinde zu verteidigen. Die Feinde bewegen sich in Echtzeit und sind nur auf den Mobiltelefonen sichtbar. Das reale Spielfeld besteht aus sechseckigen Karten, ähnlich wie bei Catan[4] und kleineren runden Spielsteinen. Die sechseckigen Karten werden aneinander angelegt und

[2] Produktseite von Quest: Zeit der Helden:
http://www.pegasus.de/quest-zeit-der-helden/ (30. Mai 2018)
[3] Definition von Tower Defense:
https://de.wikipedia.org/wiki/Tower_Defense (30. Mai 2018)
[4] Produktseite von Catan:
https://www.kosmos.de/spielware/spiele/catan/7492/catan-das-spiel
(30. Mai 2018)

bestimmen den Bereich, in dem Feinde sichtbar werden. Außerdem ist auf jeder Karte ein quadratischer, hohler Marker abgebildet. Diese Marker dienen zur Positionserkennung. In den Hohlräumen der Marker können Spielsteine platziert werden, welche die Türme repräsentieren. Die Türme bekämpfen Feinde in der Nähe.
Art of Defense ist ein gutes Beispiel, wie sich eine neutrale Instanz, die durch Augmented Reality ermöglicht wird, als großer Vorteil entfalten kann. Der Computer koordiniert die automatischen Angriffe der Türme und steuert gleichzeitig die Feinde, auch wenn sie für die Spieler unsichtbar sind. Diese Aufgaben in Echtzeit zu bewältigen ist in herkömmlichen Brettspielen kaum realisierbar. Die virtuelle Präsenz der Feinde hat dabei nicht nur einen ästhetischen Mehrwert. Sondern ermöglicht auch die Position der Feinde aufzudecken oder zu verbergen, wenn eine Karte neu platziert wird. Trotzdem geht der Charakter eines Brettspiels nicht verloren. Die Spieler agieren in erster Linie mit Spielsteinen und Karten und müssen währenddessen miteinander kommunizieren.

3.2 Augmented Reality Games

Seit der Veröffentlichung von ARKit[5] und ARCore[6], ist die wachsende Präsenz an Augmented Reality Games in den App Stores nicht mehr zu übersehen. Allerdings handelt es sich dabei meist um Spiele mit relativ großem virtuellen Anteil. Häufig ist die reale Umgebung lediglich im Hintergrund der Spielwelt zu sehen. Wie es zum Beispiel bei "Conduct AR"[7] oder "Euclidean Lands"[8] der Fall ist. Das kann den Grund haben, dass diesen Spielen greifbare Spielelemente fehlen. Fertige Spielfiguren lassen sich schließlich auch nicht in einem App Store herunterladen.

3.2.1 Kommerzielle Brettspiele

Tatsächlich werden auch schon ein paar Brettspiele mit Augmented Reality-Komponente von verschiedenen Verlagen vertrieben. Zu diesen Brettspielen wird in der Regel eine zusätzliche App für das Smartphone zur Verfügung gestellt. Diese Apps verfolgen unterschiedliche Strategien, um das Spielerlebnis zu ergänzen.

[5] Dokumentationsseite von ARKit: https://developer.apple.com/arkit/ (30. Mai 2018)
[6] Dokumentationsseite von ARCore: https://developers.google.com/ar/ (30. Mai 2018)
[7] Conduct AR im App Store:
https://itunes.apple.com/de/app/conduct-ar/id1256506674?mt=8 (30. Mai 2018)
[8] Euclidean Lands im App Store:
https://itunes.apple.com/de/app/euclidean-lands/id1181212221?mt=8
(30. Mai 2018)

Scotland Yard Master

Bei "Scotland Yard Master"[9] von Ravensburger wird Augmented Reality genutzt, um den Detektiven Hinweise auf die Position von Mister X anzuzeigen. Außerdem wird das Spielbrett mit dreidimensionalen Modellen von Londoner Sehenswürdigkeiten angereichert.

World of Yo-Ho

In "World of Yo-Ho"[10] von Iello wird das Smartphone zu einem Teil des Spielbrettes. Das Smartphone wird immer auf den Ausschnitt des Spielbrettes gelegt, auf dem sich aktuell das eigene Schiff befindet. Dieser Ausschnitt wird auf dem Display erweitert wiedergegeben. Neben dem virtuellen Schiff, werden die verschiedenen Bedienelemente des Spiels eingeblendet. Wenn nun ein Spieler in seinem Zug die virtuelle Position seines Schiffes ändert, muss er auch sein Smartphone auf den neuen Ausschnitt bewegen.

Smartplay: Das magische Museum

Einige Brettspiele nutzen die zusätzliche App, um das Spiel lediglich auf auditiver Ebene zu erweitern. Auf visuelle Erweiterung wird verzichtet. Ein Beispiel dafür ist "Smartplay: Das magische Museum"[11] von Ravensburger. Dort wird das Smartphone von einem Stativ gehalten, sodass sich das gesamte Spielbrett stets im Blickfeld der Kamera des Smartphone befindet. So können die Ergebnisse der Würfelwürfe und die Positionen der Spielfiguren und Spielplättchen auf dem Spielbrett überwacht werden. Die App wertet die Aktionen der Spieler aus und bestimmt das resultierende Ereignis. Alle nötigen Informationen bekommen die Spieler per Sprachausgabe mitgeteilt.

[9] Produktseite von Scotland Yard Master:
https://www.ravensburger.de/produkte/spiele/familienspiele/scotland-yard-master-26602/index.html (30. Mai 2018)
[10] Produktseite von World of Yo-Ho:
http://www.iellogames.com/World_of_Yoho.html (30. Mai 2018)
[11] Produktseite von Smartplay:
https://www.ravensburger.de/produkte/ravensburger-marken/smartplay/index.html (30. Mai 2018)

4 Konzept

Nun soll sich der Frage angenommen werden, wie sich das Potential von Augmented Reality vielleicht noch besser für Brettspiele ausschöpfen lässt. Die im vorherigen Kapitel erwähnten Implementierungen haben in erster Linie auf das Tracking zweidimensionaler Muster zurückgegriffen. Bei einigen Spielen war dieses Verfahren auch durchaus zielführend. Allerdings lassen sich zweidimensionale Marker selten unauffällig in die Oberfläche eines Spielbrettes oder einer Spielfigur integrieren. Sichtbare Marker können wiederum der Immersion des Spiels schaden. Daher wäre es lohnenswert ein Verfahren für das Tracking zu finden, welches das Erscheinungsbild des Spiels nicht einschränkt.
Das Tracking verschiedener Objekte zu implementieren kann mit sehr viel Aufwand verbunden sein. Es werden jedoch inzwischen mehrere Tools angeboten, die diese Aufgabe übernehmen. Das Bekannteste davon ist vermutlich Vuforia[1]. Neueste Entwicklungen dieser Tools erlauben sogar das Tracking dreidimensionaler Objekte. Damit wäre es vielleicht sogar möglich Spielfiguren zu erkennen. Was einen entscheidenden Mehrwert für das Spielerlebnis bieten könnte. Zum Beispiel ließe sich das in Kapitel 3.2 erwähnte Spiel Art of Defense aufwerten. In dem Fall könnten die Spielsteine, die die Türme repräsentieren sollen, gegen Miniaturen echter Türme ausgetauscht werden.
Ob das Tracking von dreidimensionalen Spielfiguren machbar ist und welchen Nutzen das bringen könnte, soll nun, mit der Entwicklung eines prototypischen Brettspiels, untersucht werden. Dazu wurde zuerst ein minimales Game Design erarbeitet, welches ein zweidimensionales Spielbrett und dreidimensionale Spielfiguren beinhaltet.

4.1 Game Design

Das Spielbrett wird auf einer flachen Oberfläche ausgebreitet und nicht bewegt. Auf dem Spielbrett ist ein kariertes Muster abgebildet. Jedes Quadrat des Musters bildet ein Spielfeld für die Spielfiguren. Auf jedem Spielfeld kann immer nur eine Spielfigur stehen. Jeder Spieler kontrolliert eine eigene Spielfigur. Außerdem verfügt jeder Spieler über ein Gerät, womit das

[1] Produktseite von Vuforia: https://www.vuforia.com (30. Mai 2018)

durch Augmented Reality erweiterte Spielfeld betrachtet werden kann.
Die Spielfiguren starten auf verschiedenen Spielfeldern am äußeren Rand des Spielbrettes. Die Spieler sind abwechselnd am Zug. Jeder Zug besteht aus zwei Aktionen. Als Erstes wählt der Spieler ein neues Feld für seine Spielfigur. Das gewählte Spielfeld darf maximal drei Felder vom aktuellen Spielfeld entfernt sein. Als Zweites wird das neue Spielfeld untersucht.
Auf einem zufällig bestimmten Spielfeld ist ein Schatz versteckt. Auf allen anderen Spielfeldern sind Hinweise auf das Spielfeld mit dem Schatz versteckt. Der Schatz und die Hinweise werden mittels Augmented Reality auf das Spielbrett projiziert und sind daher nur durch das Gerät des Spielers zu sehen. Der Spieler, der zuerst den Schatz findet, gewinnt das Spiel.
Wenn ein Spieler ein Spielfeld untersucht, wird der Hinweis oder der Schatz, der sich unter diesem Feld verbirgt, sichtbar. Der Hinweis wird nur für den Spieler sichtbar, der grade am Zug ist. Jeder Hinweis verrät dem Spieler eine Gruppe von Feldern, die das Spielfeld mit dem Schatz beinhaltet. So können die Spieler mit der Zeit immer mehr Felder ausschließen, bis der Schatz gefunden wurde.

4.2 Use Cases

Aus dem beschriebenen Game Design wurden zwei Use Cases abgeleitet. Der Erste schildert den Prozess zum Erstellen einer Spielpartie. Und der zweite Use Case stellt den hauptsächlichen Spielablauf dar.

4.2.1 Spielerstellung

Bei der Spielerstellung öffnet ein Spieler eine Partie, die er dann konfigurieren kann. Währenddessen können andere Spieler der Partie beitreten. Wenn sich alle Mitspieler verbunden haben

und die Konfiguration abgeschlossen ist, kann der Ersteller der Partie das Spiel starten.

Akteure	Spieler A und Spieler B
Vorbedingung	Das Spielbrett ist ausgebreitet, die Spielfiguren liegen bereit und beide Spieler haben die App zum Spiel auf ihren Geräten installiert.
Nachbedingung	Eine Partie ist gestartet.
Hauptszenario	1. Beide Spieler starten die App zum Spiel. 2. Beide Spieler geben ihre Spielernamen ein. 3. Spieler A öffnet das Menü zum Erstellen einer Partie. 4. Spieler A wählt das Spielbrett, auf dem gespielt werden soll. 5. Spieler A wählt die Spielfigur, die er verwenden möchte. 6. Spieler B öffnet das Menü zum Suchen nach Partien. 7. Das Spiel sucht im Bluetooth- oder WLAN-Netzwerk nach offenen Partien. 8. Das Spiel findet die von Spieler A geöffnete Partie und zeigt sie Spieler B an. 9. Spieler B tritt der von Spieler A geöffneten Partie bei. 10. Spieler B wählt die Spielfigur, die er verwenden möchte. 11. Spieler A startet die Partie.
Ausnahmeszenarien	9. Das von Spieler A gewählte Spielbrett ist nicht für zwei Spieler geeignet. 9.1. Der Versuch von Spieler B der Partie beizutreten, wird mit einer Fehlermeldung abgebrochen. 9.2. Spieler A wählt ein anderes Spielbrett, auf dem zwei Spieler Platz finden. 9.3. Spieler B tritt der von Spieler A geöffneten Partie bei.

4.2.2 Spielablauf

Die Schritte 5 bis 8 des Hauptsuenarios stellen einen Spielzug dar. Diese führen Spieler A und Spieler B nacheinander abwechselnd durch, bis das Spiel beendet ist.

Akteure	Spieler A und Spieler B
Vorbedingung	Eine Partie ist gestartet.
Nachbedingung	Eine Partie ist beendet.
Hauptszenario	1. Beide Spieler richten die Kameras ihrer Geräte auf das Spielbrett. 2. Das Spiel erkennt das Spielbrett und markiert das Startfeld des jeweiligen Spielers. 3. Beide Spieler positionieren ihre Spielfigur auf dem markierten Spielfeld. 4. Beide Spieler richten die Kameras ihrer Geräte auf ihre Spielfiguren und lassen sich die korrekte Positionierung vom Spiel bestätigen. 5. Der Spieler, der am Zug ist, bewegt seine Spielfigur auf ein neues Spielfeld und betrachtet anschließend die Situation durch seine Kamera. 6. Das Spiel erkennt die neue Position der Spielfigur und prüft die Gültigkeit des Zuges. 7. Wenn der Spieler eine gültige Position gewählt hat, bestätigt er seinen Zug. Ansonsten muss der Spieler es erneut versuchen und Schritt 5 wiederholen. 8. Das Spiel verrät dem Spieler was unter dem neuen Spielfeld verborgen ist. 9. Das Spielfeld enthält einen Hinweis. Auf dem Bildschirm des Spielers wird eine Gruppe von Feldern markiert, unter welchen sich möglicherweise der Schatz befinden kann.
Alternativszenarien	9. Das Spielfeld enthält den Schatz. 9.1. Dem Spieler wird der Schatz angezeigt und zum Sieg gratuliert. 9.2. Das Spiel wird beendet und die Spieler können zum Hauptmenü zurückkehren.

4.3 Anforderungen

Auf der Basis der Use Cases wurden die spezifischen Anforderungen generiert. Diese dienten als Grundlage bei der Umsetzung des Prototypen.

4.3.1 Funktionale Anforderungen

Nummer	Titel	Beschreibung
FA01	Spielernamen bestimmen	Die Anwendung soll dem Benutzer die Möglichkeit bieten einen Spielernamen zu bestimmen.
FA02	Spielpartie erstellen	Die Anwendung muss dem Benutzer die Möglichkeit eine neue Spielpartie zu erstellen.
FA03	Mitspieler suchen	Die Anwendung muss neu geöffnete Spielpartien im Bluetooth- oder WLAN-Netzwerk für potentielle Mitspieler zugänglich machen.
FA04	Spielpartie suchen	Die Anwendung muss dem Benutzer die Möglichkeit bieten im Netzwerk nach bereits geöffneten Spielpartien zu suchen.
FA05	Spielpartien anzeigen	Die Anwendung muss alle im Netzwerk gefundenen Spielpartien anzeigen.
FA06	Spielpartie beitreten	Die Anwendung muss dem Benutzer die Möglichkeit bieten einer gefundenen Spielpartie beizutreten.
FA07	Spielfigur wählen	Die Anwendung soll allen Benutzen, die Teilnehmer einer Spielpartie sind, die Möglichkeit bieten eine Spielfigur zu wählen.
FA08	Gleiche Spielfiguren	Die Anwendung soll verhindern, dass mehrere Teilnehmer einer Spielpartie die gleiche Spielfigur wählen.
FA09	Spielbrett wählen	Die Anwendung soll ausschließlich dem Benutzer, der die Spielpartie erstellt hat, die Möglichkeit bieten ein Spielbrett zu wählen.
FA10	Spiel starten	Die Anwendung soll ausschließlich dem Benutzer, der die Spielpartie erstellt hat, die Möglichkeit bieten das Spiel zu starten.

FA11	Spielbrett erkennen	Die Anwendung muss die Position des Spielbrettes erkennen können.
FA12	Spielfigur erkennen	Die Anwendung muss die Position der eigenen Spielfigur erkennen können.
FA13	Position erkennen	Die Anwendung muss die Position der eigenen Spielfigur auf dem Spielbrett bestimmen können.
FA14	visuelle Erweiterungen	Die Anwendung muss das Spielbrett und die eigene Spielfigur mit dreidimensionalen virtuellen Inhalten visuell erweitern können.
FA15	Texte einblenden	Die Anwendung muss auf dem Kamerabild Texte einblenden können.
FA16	Spielablauf	Die Anwendung muss den Spielablauf steuern, wie er im Use Case Spielablauf beschrieben ist.
FA17	Spielfigur positionieren	Die Anwendung muss die Positionierung der Spielfigur steuern, wie sie im Hauptszenario des Use Case Spielablauf, in den Schritten 1 bis 4, beschrieben ist.
FA18	Spielfigur bewegen	Die Anwendung muss die Bewegung der Spielfigur auf ein neues Spielfeld steuern, wie sie im Hauptszenario des Use Case Spielablauf, in den Schritten 5 bis 7, beschrieben ist.
FA19	Spielfelder generieren	Die Anwendung muss jedes Spielfeld mit einem Inhalt füllen. Jedes Spielfeld beinhaltet entweder einen Hinweis oder den Schatz.
FA20	Spielfelder zählen	Die Anwendung muss die Spielfelder zwischen dem alten und dem neuen Feld, auf dem sich die Spielfigur befindet, zählen können.
FA21	Spielfelder aufdecken	Die Anwendung muss das Spielfeld, auf das die Spielfigur bewegt wurde, aufdecken.
FA22	Schatz darstellen	Die Anwendung muss die Entdeckung des Schatzes darstellen können.
FA23	Hinweis darstellen	Die Anwendung muss die Entdeckung eines Hinweises darstellen können.

FA24	Spielzug bestätigen	Die Anwendung muss dem Benutzer die Möglichkeit bieten die Bewegung seiner Spielfigur zu bestätigen.

4.3.2 Nicht funktionale Anforderungen

Nummer	Titel	Beschreibung
NFA25	Anleitung	Die Anwendung soll dem Benutzer während des Spiels Anleitungen geben, sodass auch Benutzer, die über keine Kenntnis der Spielregeln verfügen, den Sinn des Spiels verstehen.
NFA26	intuitive Bedienung	Die Anwendung soll möglichst intuitiv bedienbar sein, sodass dem Benutzer alle zur Verfügung stehenden Optionen klar sind.
NFA27	Hauptmenü	Das Hauptmenü besteht aus • einem Button der zum Menü für die Erstellung einer neuen Spielpartie navigiert • und einem Button der zum Menü für die Suche nach bereits geöffneten Spielpartien navigiert.
NFA28	Suchmenü	Das Menü für die Suche nach bereits geöffneten Spielpartien enthält • eine Liste der Partien, die bereits gefunden worden sind, • einen Button zum Abbrechen • und einen Indikator, der anzeigt, dass im Moment nach Partien gesucht wird.
NFA29	Gefundene Spielpartien	Jeder Eintrag in der Liste der bereits gefundenen Spielpartien enthält • den Spielernamen des Benutzers, der die Partie erstellt hat • und einen Button zum Beitreten der Partie.

5 Umsetzung

In diesem Kapitel wird die Implementierung des Prototypen geschildert. Begonnen wird mit der Entscheidung für eine bestimmte Plattform und Programmiersprache. Danach wird auf die Werkzeuge eingegangen, die das Tracking übernommen haben, was für Augmented Reality ausschlaggebend ist. Außerdem wird ausgeführt welche Software noch genutzt wurde, um die wichtigsten Anforderungen umzusetzen. Zusätzlich wurde ein Editor entwickelt, der dazu dient Spielfiguren und Spielbretter für das Spiel vorzubereiten. Mit der Beschreibung der virtuellen und realen Repräsentationen dieser Spielelemente schließt das Kapitel ab.

5.1 Plattform

Früh war klar, dass der Prototyp auf einer mobilen Plattform umgesetzt werden soll. Aktuelle Smartphones und Tablets sind inzwischen so leistungsstark, dass sie den Anforderungen von Augmented Reality an die Hardware genügen. Dazu kommt der große Vorteil der Verfügbarkeit und weiten Verbreitung der Geräte. So sind viele potentielle Spieler schon jetzt mit der erforderlichen Hardware versorgt.

5.1.1 iOS

Es wurde entschieden den Prototypen speziell für die Plattform iOS zu entwickeln. Das hatte in erster Linie den Grund, dass für die Entwicklung unter iOS bereits mehr Expertise vorhanden war. Außerdem bietet VisionLib eine umfangreichere Unterstützung für iOS, in Form einer nativen Programmierschnittstelle.

5.1.2 Swift und Objective-C

Die Entscheidung für iOS grenzt die Auswahl der Programmiersprachen auf Swift und Objective-C ein. Swift bildet die modernere Alternative zu dem in die Jahre gekommenen Objective-C. Daher wurde der größte Teil des Prototypen mit Swift entwickelt. Außerdem sind Swift und Objective-C kombinierbar. Von dieser Kompatibilität wurde auch gebraucht, da stellenweise der Einsatz von Objective-C nötig war. Zum Einen für den Aufruf von Objective-C Libraries,

wie VisionLib und zum Anderen für den Einsatz von C++. Der Einsatz von C++ war wiederum notwendig, um auf die OpenCV Library zuzugreifen.

5.2 Tracking Libraries

Für das Tracking der Spielfiguren und des Spielbrettes wurde nach einer Lösung gesucht, die, wenigstens für den Entwicklungszeitraum, kostenlos zur Verfügung stehen konnte. Außerdem sollten die Spielfiguren als dreidimensionales Objekt erfasst und verfolgt werden. Um das 3D Object Tracking nutzen zu können, verlangen die Anbieter der Tracking Tools in der Regel hohe Lizenzgebühren. Das ist beispielsweise auch bei Vuforia der Fall. Glücklicherweise wurde eine kostenlose Lizenz für die akademische Nutzung von VisionLib gewährt. Mit VisionLib ist das Tracking von zweidimensionalen Postern und dreidimensionalen Objekten möglich. Allerdings können damit nicht zwei Dinge gleichzeitig verfolgt werden. Daher musste für das Tracking des Spielbrettes eine ergänzende Lösung gefunden werden.
Um ein Bild zu erhalten, greifen die meisten Tracking Tools direkt auf die Kamera zu. Diese Ressource kann jedoch nicht von zwei Quellen gleichzeitig beansprucht werden kann. Daher ist das Spektrum an Tools, die ergänzend für das Tracking des Spielbrettes genutzt werden können, stark eingeschränkt. Es wurde eine Library benötigt, die das Tracking von zweidimensionalen Oberflächen, unabhängig vom Kamerazugriff, durchführen kann. OpenCV erfüllt dieses Kriterium.

5.2.1 VisionLib

Um die Spielfigur zu erkennen und damit die Anforderung Nummer FA12 zu erfüllen, wurde VisionLib[1] von Visometry verwendet. Wie bereits erwähnt, ermöglicht VisionLib das Tracking von zweidimensionalen Postern und dreidimensionalen Objekten. Bevor ein dreidimensionales Objekt verfolgt werden kann, muss es jedoch in einer bestimmten Position von der Kamera erfasst werden.
VisionLib wird auf den Plattformen Android, iOS, macOS, Windows und HoloLens unterstützt. Für die Implementierung werden Unity[2] Packages bereit gestellt. Alternativ kann auch auf ein C Interface zugegriffen werden. Zusätzlich wird für iOS und macOS noch ein natives Objective-C SDK angeboten. Dieses wurde bei der Umsetzung des Prototypen eingesetzt. Das war notwendig, da über das Unity Plugin nicht auf das Kamerabild zugegriffen werden konnte.

[1] Internetseite von VisionLib: https://visionlib.com (30. Mai 2018)
[2] Internetseite von Unity: https://unity3d.com/de (30. Mai 2018)

Es war wichtig das Kamerabild von VisionLib zu erhalten, um es mit OpenCV weiterverarbeiten zu können.

Tracking Configuration

Alle Parameter zur Konfiguration von VisionLib sind in einem Tracking Configuration File[3] gebündelt. In dieser Datei wird abhängig davon, ob ein zweidimensionales Poster und ein dreidimensionales Objekt verfolgt werden soll, eine entsprechende Bild- oder Modelldatei angegeben. Neben weiteren wichtigen Parametern, wie der initialen Position, in der das Objekt von der Kamera erfasst werden muss, finden sich einige Parameter zur Anpassung und Optimierung des Tracking. Zum Beispiel kann ein Laplace Threshold und ein Normal Threshold konfiguriert werden. Diese beide Parameter steuern mit welcher Genauigkeit, die Kanten auf dem Modell beim Tracking berücksichtigt werden. Niedrige Werte, die zu besseren Ergebnissen führen können, beanspruchen mehr Leistung. Daher sollte immer ein angemessener Kompromiss zwischen Performance und Präzision gefunden werden.
Laut Anforderung Nummer FA07 soll es möglich sein, eine von mehreren Spielfiguren zu wählen. Im Tracking Configuration File kann jedoch nur ein Modell angegeben werden. Deshalb wird das gewählte Modell erst zur Laufzeit, beim Start einer Partie, in die Datei eingetragen.

5.2.2 OpenCV

OpenCV[4] steht für Open Source Computer Vision Library. Bradski und Kaehler (2008) beschreiben OpenCV als Bildverarbeitungsbibliothek, die speziell für Echtzeitanwendungen entwickelt wurde. Die Library wurde in C und C++ geschrieben und ist unter Anderem auch als iOS Framework verfügbar. Für das Tracking des Spielbrettes, gemäß Anforderung Nummer FA11, wurde OpenCV mit dem zusätzlichen Modul Aruco kompiliert.

5.2.3 Aruco

Aruco[5] ist eine Open Source Library für das Tracking zweidimensionaler Marker. Die Funktionsweise wird von Garrido-Jurado u. a. (2014) und Garrido-Jurado u. a. (2016) genauer ausgeführt. Neben dem Tracking einzelner Aruco Marker, kann auch eine Gruppe mehrerer Marker, als Aruco Board, erkannt werden. In dem Fall müssen nur wenige Marker aus der Gruppe sichtbar

[3] Dokumentationsseite von VisionLib zum Tracking Configuration File: https://visionlib.com/documentation/tracking-configuration-file/ (30. Mai 2018)
[4] Internetseite von OpenCV: https://opencv.org (30. Mai 2018)
[5] Internetseite von Aruco: http://www.uco.es/investiga/grupos/ava/node/26 (30. Mai 2018)

sein, um die Position des Aruco Boards bestimmten zu können.
So ein Aruco Board wird verwendet um die Postion des Spielbrettes erkennen zu können. Für den Prototypen wurden die Aruco Marker an den äußeren Rändern des Spielbrettes platziert. So sind aus möglichst vielen Blickwinkeln wenigstens ein paar Marker für die Kamera sichtbar.

Kamerabild

Das Kamerabild wird von VisonLib empfangen und als Hintergrund der virtuellen Szene eingesetzt. Um dieses Bild mit OpenCV und Aruco weiterverarbeiten zu können, muss es zunächst in ein passendes Format konvertiert werden. VisionLib liefert das Kamerabild im Format der Schnittstelle MTLTexture[6], die zum Metal[7] Framework von Apple gehört. Dieses Objekt muss in die für OpenCV übliche Klasse Mat[8] überführt werden.
Als erster Zwischenschritt, wird von MTLTexture zu UIImage[9] konvertiert, was inzwischen mit den hauseigenen Bibliotheksfunktionen von Swift möglich ist. Die Übersetzung von UIImage in eine OpenCV Mat ist bereits besser dokumentiert[10]. Im Wesentlichen ist dabei zu beachten, dass das Objekt von UIImage in 4 Farbkanälen mit je 8 Bit vorliegt[11]. Und da in OpenCV in der Regel mit RGB gearbeitet wird, musste der Alpha Channel verworfen werden.

Kalibrierung

Damit die Position eines Objektes im Raum möglichst präzise abgeschätzt werden kann, muss in der Regel die verwendete Kamera kalibriert werden. Das geschieht meistens durch das Fotografieren oder Filmen eines karierten Mustern, aus möglichst vielen Perspektiven. Wie von Garrido-Jurado u. a. (2016) beschrieben, wird dabei die entscheidende Kameramatrix M generiert.

[6] Dokumentationsseite von MTLTexture:
https://developer.apple.com/documentation/metal/mtltexture (30. Mai 2018)
[7] Dokumentationsseite von Metal: https://developer.apple.com/metal/ (30. Mai 2018)
[8] Dokumentationsseite von Mat:
https://docs.opencv.org/3.1.0/d3/d63/classcv_1_1Mat.html (30. Mai 2018)
[9] Dokumentationsseite von UIImage:
https://developer.apple.com/documentation/uikit/uiimage (30. Mai 2018)
[10] Dokumentationsseite von OpenCV zur Konvertierungen zwischen UIImage und Mat:
https://docs.opencv.org/2.4/doc/tutorials/ios/image_manipulation/image_manipulation.html (30. Mai 2018)
[11] Dokumentationsseite von VisionLib zur Schnittstelle für das Kamerabild:
https://visionlib.com/wp-content/uploads/visionlib/documentation/protocolvl_frame_listener_interface_01-p.html#ad700999d7deb181378f9d4c75c10405e (30. Mai 2018)

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Wobei f_x und f_y die Brennweite darstellen und c_x und c_x den Bildhauptpunkt repräsentieren. Diese Kameramatrix wird auch von Aruco, bei der Bestimmung der Position des Spielbrettes, verlangt[12]. VisionLib muss ebenfalls mit einer derartigen Kameramatrix arbeiten, um die Position der Spielfigur zu bestimmen. Daher sind die Parameter f_x, c_x, f_y und c_y für einige Geräte, die von VisionLib unterstützt werden, bereits berechnet worden und in der Library enthalten. Diese Parameter können sogar auch abgegriffen werden. Dafür bietet VisionLib ebenfalls eine entsprechende Schnittstelle[13] an.
Da die Parameter für die Kameramatrix in normierter Form bei VisionLib vorliegen, müssen diese vor der Verwendung noch an die Auflösung der neuen Umgebung angepasst werden. Die relevanten Werte dafür werden von SceneKit definiert und können sich von Gerät zu Gerät unterscheiden. Interessant sind die Höhe s_h und die Breite s_w, mit der im SceneKit View[14] gerechnet wird. Mit diesen Werten wird die normierte Kameramatrix M auf die passende Kameramatrix M_s skaliert, die zu SceneKit passt.

$$M_s = \begin{bmatrix} s_w & 0 & 0 \\ 0 & s_h & 0 \\ 0 & 0 & 1 \end{bmatrix} \times M$$

Durch diese Vorgehensweise musste keine eigene Kalibrierung der Kamera mehr vorgenommen werden.

SceneKit View Matrix

Das Ergebnis der Positionsberechnung von Aruco sind ein Translations- und ein Rotationsvektor. Diese Vektoren müssen in eine Transformationsmatrix übersetzt werden, die von SceneKit, zur Positionierung virtueller Objekte, verwendet werden kann. Als erster Schritt wird der Rotationsvektor nach Rodrigues in eine Rotationsmatrix umgewandelt. Anschließend wird die

[12] Dokumentationsseite von OpenCV zur Bestimmung der Position eines Aruco Boards: https://docs.opencv.org/3.4/d9/d6a/group__aruco.html#gabb2578b9e18b13913b1d3e0ab1b554f9 (30. Mai 2018)
[13] Dokumentationsseite von VisionLib zur Schnittstelle für die Kameraparameter: https://visionlib.com/wp-content/uploads/visionlib/documentation/protocolvl_frame_listener_interface_01-p.html#a390e563938a9c9eab27500a021df0b4f (30. Mai 2018)
[14] Dokumentationsseite von SceneKit zum SceneKit View: https://developer.apple.com/documentation/scenekit/scnview (30. Mai 2018)

Rotationsmatrix R zusammen mit dem Translationsvektor t in eine View Matrix V überführt[15].

$$V = \begin{bmatrix} R_{11} & R_{21} & R_{31} & t_1 \\ R_{12} & R_{22} & R_{32} & t_2 \\ R_{13} & R_{23} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Das ist die fertige View Matrix in der Welt von OpenCV.

Als Nächstes ist die Konvertierung zu SceneKit nötig. Zum Einen, da OpenCV und SceneKit nicht mit dem gleichen Koordinatensystem arbeiten. Und zum Anderen, da die Matrizen unterschiedlich angeordnet sind. Im Vergleich zu SceneKit, ist bei OpenCV die Y- und Z-Achse umgekehrt. Daher wird die OpenCV View Matrix V mit einer entsprechenden Matrix multipliziert, die diese beiden umgekehrten Achsen ausgleicht. Daraus ergibt sich allerdings vorerst die transponierte SceneKit View Matrix V_s^T. Das liegt daran, dass in OpenCV die Matrizen zeilenweise gespeichert werden und in SceneKit spaltenweise.

$$V_s^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times V$$

Durch einfaches Transponieren der Matrix V_s^T, ergibt sich dann die fertige View Matrix V_s für SceneKit. Diese muss dann nur noch aus der OpenCV Mat in eine SCNMatrix4[16] kopiert werden. Als SCNMatrix4 ist die Matrix dann in vollem Umfang von SceneKit nutzbar.

5.2.4 Alternativen

Eine mögliche Alternative zu VisionLib wäre vielleicht das weiter verbreitete Vuforia gewesen. Allerdings stand kein kostenfreier Zugriff auf das CAD Model Tracking von Vuforia zur Verfügung. Ein unentgeltlicher Zugang zu VisionLib, wurde hingegen relativ unkompliziert gewährt. Erste Tests mit dem Model Tracking von VisionLib verliefen schon sehr vielversprechend, weswegen die Suche nach weiteren Alternativen, für das Tracking der Spielfiguren, nicht weiter verfolgt wurde.

Als Alternative zu dem auf Markern basierte Tracking von Aruco, existieren schon mehr brauchbare Ansätze. Die Implementierung eines Verfahrens zum Erkennen des Spielbrettes, das ohne Marker auskommt, wäre jedoch mit erheblich mehr Entwicklungsaufwand verbunden

[15] Forenseite von OpenCV zur Konvertierung der View Matrix:
http://answers.opencv.org/question/23089/opencv-opengl-proper-camera-pose-using-solvepnp/?answer=23123#post-id-23123 (30. Mai 2018)

[16] Dokumentationsseite von SCNMatrix4:
https://developer.apple.com/documentation/scenekit/scnmatrix4
(30. Mai 2018)

gewesen. Außerdem kann das Marker Tracking auch noch relativ stabil funktionieren, wenn der Fokus der Kamera mal nicht auf der Oberfläche des Spielbrettes liegt. Was durchaus passieren kann, da gleichzeitig die Spielfigur erkannt werden muss, die tendenziell mehr Aufmerksamkeit von der Kamera benötigt. Zudem hat das Tracking eines Aruco Boards den Vorteil, dass nur eine relativ geringe Anzahl an Markern tatsächlich sichtbar sein muss, um einwandfrei zu funktionieren. Dadurch können auch problemlos Teile des Spielbrettes, zum Beispiel durch die Spielfigur, verdeckt werden.
Generell wurde das Tracking einer dreidimensionalen Spielfigur, ohne zusätzliche Marker, höher priorisiert. Daher wurden die Marker auf dem Spielbrett, natürlich auch wegen der Stabilität und vergleichsweise leichten Implementierung, für den Prototypen in Kauf genommen.

5.3 Software

In diesem Teil wird weitere Software beschrieben, die, neben den oben genannten Tracking Libraries, bei der Umsetzung zum Einsatz kam. Zum größten Teil wird darauf eingegangen, welche Aufgaben verwendete Swift Libraries erfüllen, die bereits mit den für iOS notwendigen Entwicklungswerkzeugen mitgeliefert werden. Vor allem die enthaltenen Libraries für die Entwicklung von Spielen, waren für den Prototypen durchaus praktikabel. Sodass dafür nicht auf Software von Drittanbietern zurückgegriffen werden musste.

5.3.1 Xcode

Um native iOS App entwickeln und kompilieren zu können wird die Entwicklungsumgebung Xcode[17] benötigt. Xcode wird von Apple selbst entwickelt und ist nur für macOS verfügbar. Neben dem Interface Builder[18], mit dem Benutzeroberflächen von Apps gestaltet werden können, ist auch ein Simulator enthalten. Mit dem Simulator können iOS Apps getestet werden, ohne das die Apps auf externen Geräten installiert werden müssen.
Davon konnte bei der Entwicklung des Prototypen jedoch nicht profitiert werden, da die App Zugriff auf die Kamera eines realen Gerätes benötigt. Vom Interface Builder wurde hingegen Gebrauch gemacht, um unter Anderem die Menüs gemäß der Anforderungen NFA27, NFA28 und NFA29 zu konfigurieren.

[17] Internetseite von Xcode: https://developer.apple.com/xcode/ (30. Mai 2018)
[18] Internetseite vom Xcode Interface Builder:
https://developer.apple.com/xcode/interface-builder/ (30. Mai 2018)

5.3.2 Multipeer Connectivity

Das Multipeer Connectivity[19] Framework implementiert die Kommunikation zwischen Geräten mittels WLAN oder Bluetooth. Dabei unterscheidet das Framework selbst, je nach Verfügbarkeit, ob eine WLAN- oder eine Bluetooth-Verbindung aufgebaut werden soll.
Wenn ein Benutzter, eine Spielpartie öffnet, wird diese im Netzwerk automatisch sichtbar (FA03). Andere Geräte, die nun das Menü zur Suche nach Spielpartien betreten, beginnen automatisch im Netzwerk nach offenen Partien zu browsen (FA04). Wird eine offene Partie gefunden, wird sie, in der entsprechenden Liste, dem Benutzter angezeigt (FA05). Der suchende Benutzter kann dann entscheiden, ob er der Partie beitreten möchte (FA06).
Die weitere Kommunikation im Spielverlauf dient dazu, den aktuellen Spielzustand, nach jedem Zug eines Spielers, allen anderen Spielern bekannt zu machen. Sodass jeder weiß wer momentan an der Reihe ist.

5.3.3 SceneKit

SceneKit[20] ist das 3D Graphics Framework von Apple und verfügt auch über eine Physics Engine. Alle virtuellen Inhalte, die im Kamerabild erweitert werden, sind in der SceneKit Scene[21] enthalten. Dort werden auch die von VisionLib und Aruco gelieferten Transformationen, auf die virtuellen Repräsentationen der Spielfelder und Spielfigur angewandt. Wodurch auch die virtuellen Positionen in der Physics Engine, immer mit den realen Positionen synchron sind. Weitere virtuelle Objekte können beliebig hinzugefügt und relativ zur Position von Spielbrett oder Spielfigur verankert werden, sodass sie als visuelle Erweiterungen wirken (FA14). Entsprechend kann auch der Schatz dargestellt werden (FA22).

Contact Detection

Die Physics Engine wird innerhalb einer SceneKit Scene von einer Physics World[22] repräsentiert. Dort können Kontakte und Kollisionen, zwischen in der Szene enthaltenen Objekten, simuliert werden.
Um gemäß der Anforderung Nummer FA13 die Position der Spielfigur auf dem Spielbrett

[19] Dokumentstionsseite vom Multipeer Connectivity Framework: https://developer.apple.com/documentation/multipeerconnectivity (30. Mai 2018)
[20] Dokumentstionsseite von SceneKit: https://developer.apple.com/scenekit/ (30. Mai 2018)
[21] Dokumentstionsseite von SceneKit zur SceneKit Scene: https://developer.apple.com/documentation/scenekit/scnscene (30. Mai 2018)
[22] Dokumentstionsseite von SceneKit zur Physics World: https://developer.apple.com/documentation/scenekit/scnscene/1522643-physicsworld (30. Mai 2018)

ermitteln zu können, muss geprüft werden können, ob ein Kontakt zwischen der Spielfigur und einem Spielfeld vorhanden ist. Jedes Objekt kann einen beliebig geformten Physics Body[23] enthalten. Der Physics Body eines Spielfeldes entspricht den Konturen auf dem Spielbrett. Bei der Spielfigur ist das anders.

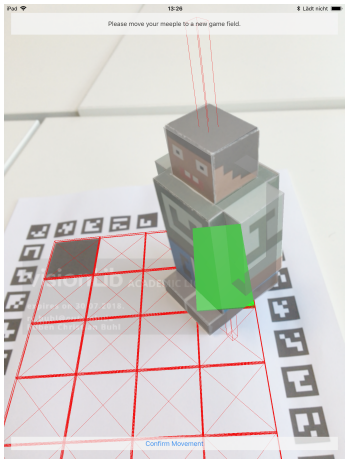


Abbildung 5.1: Physics Body der Spielfigur und der Spielfelder

Wenn eine Spielfigur auf einem Spielfeld steht, entsteht kein deutlicher Kontakt. Daher wird den Spielfiguren ein senkrechten Balken als Physics Body zugewiesen. Dieser Balken ist schmaler, aber höher, als das eigentliche Modell der Spielfigur, sodass er unten und oben aus der Spielfigur herausragt. Steht die Spielfigur nun auf einem Spielfeld, durchstößt der unten herausragende Balken den Physics Body des Spielfeldes und es entsteht ein eindeutiger Kontakt. Außerdem muss, durch den geringeren Durchmesser des Balkens, die Spielfigur nicht exakt zentriert auf dem Spielfeld platziert werden. Weil, durch den geringeren Durchmesser des Balkens, auch die Wahrscheinlichkeit reduziert ist, versehentlich mehrere Spielfelder gleichzeitig zu kontaktieren.

5.3.4 GameplayKit

Das GameplayKit[24] Framework stellt eine Reihe von Werkzeugen bereit, die bei der Entwicklung von Spielen gebräuchlich sind. Dazu zählt beispielsweise ein Entity Component System oder

[23] Dokumentstionsseite von SceneKit zum Physics Body:
https://developer.apple.com/documentation/scenekit/scnnode/1407988-physicsbody (30. Mai 2018)

[24] Guide zu GameplayKit:
https://developer.apple.com/library/content/documentation/General/

auch Unterstützung bei der Implementierung von Agenten. Bei dem Prototypen wird allerdings lediglich von der State Machine und den Möglichkeiten zum Pathfinding Gebrauch gemacht.

Pathfinding

Laut der Anforderung Nummer FA20 müssen die Spielfelder zwischen zwei Positionen gezählt werden, um festzustellen, ob die Spielfigur von dem Spieler zu weit bewegt wurde. Dabei soll natürlich der kürzeste Pfad, zwischen den beiden Feldern, betrachtet werden. Damit der kürzeste Pfad mittels GameplayKit gesucht werden kann, müssen die Spielfelder in einem Graph[25] organisiert sein. Jedes Spielfeld im Graph verkörpert einen Knoten[26]. Beim Start der Spielpartie werden alle Spielfelder mit ihren benachbarten Knoten verbunden, sodass später problemlos der Pfad zwischen zwei beliebigen Knoten abgefragt werden kann.

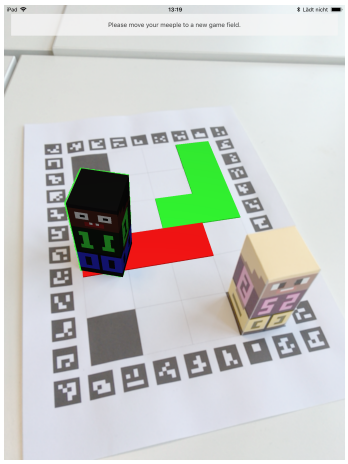


Abbildung 5.2: Pathfinding zwischen zwei Spielfeldern

Conceptual/GameplayKit_Guide/index.html#//apple_ref/doc/uid/TP40015172 (30. Mai 2018)

[25] Dokumentationsseite von GameplayKit zum Graph: https://developer.apple.com/documentation/gameplaykit/gkgraph (30. Mai 2018)

[26] Dokumentationsseite von GameplayKit zum Knoten: https://developer.apple.com/documentation/gameplaykit/gkgraphnode (30. Mai 2018)

State Machine

Die Steuerung des Spielablaufs, entsprechend der Anforderung Nummer FA16, bildet den Kern des Systems. Die Anforderungen zur Positionierung (FA17) und Bewegung (FA18) der Spielfigur und zum Aufdecken eines Spielfeldes (FA21), werden von je einem der fünf Zustände repräsentiert.

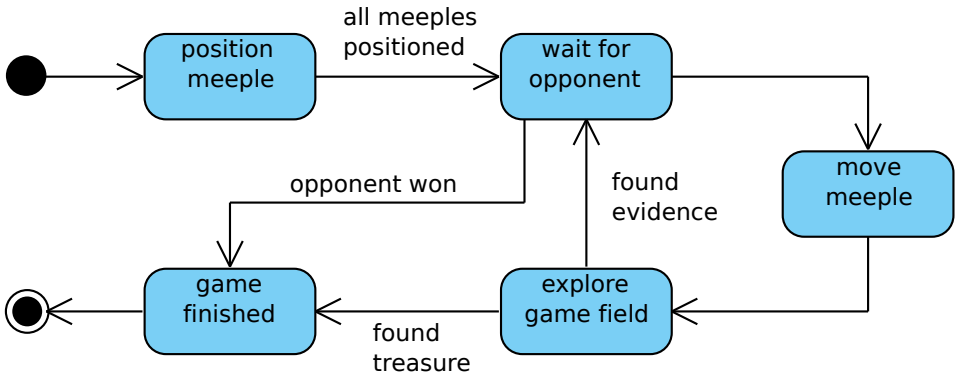


Abbildung 5.3: State Machine zum Spielablauf

Im Wesentlichen spiegelt die State Machine aus Abbildung 5.3 den Use Case zum Spielablauf aus Kapitel 4.2.2 wieder. Für die Umsetzung dieser State Machine wurde ebenfalls auf GameplayKit zurückgegriffen. Die State Machine[27] von GameplayKit unterstützt die unabhängige Implementierung der einzelnen Zustände, aus denen anschließend die State Machine zusammengesetzt werden kann.

5.3.5 Blender

Blender[28] ist ein Open Source 3D Creation Tool, mit dem die Spielfiguren des Prototypen modelliert wurden. Der Funktionsumfang von Blender kann noch mit Add-ons erweitert werden. Ein Add-on, das installiert wurde, ergänzt eine weitere Exportfunktion, die den Export als Paper Model[29] ermöglicht. Auf dieses Feature wird in Kapitel 5.5.3 näher eingegangen.

[27] Dokumentationsseite von GameplayKit zur State Machine: https://developer.apple.com/documentation/gameplaykit/gkstatemachine (30. Mai 2018)
[28] Internetseite von Blender: https://www.blender.org (30. Mai 2018)
[29] Dokumentations von Blender zum Paper Model Add-on: https://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Import-Export/Paper_Model (30. Mai 2018)

5.4 Editor

Das digitale Datenmodell des Spielbrettes muss konsistent mit der realen Repräsentation sein. Damit diese Konsistenz gewährleistet ist und außerdem leichter mit verschiedenen Spielbrettern experimentiert werden kann, wurde eine zusätzliche Software entwickelt. Es handelt sich dabei um ein Editor Tool, das die Aufgabe hat Spielbretter und Spielfiguren für die spätere Verwendung in der App zu serialisieren.

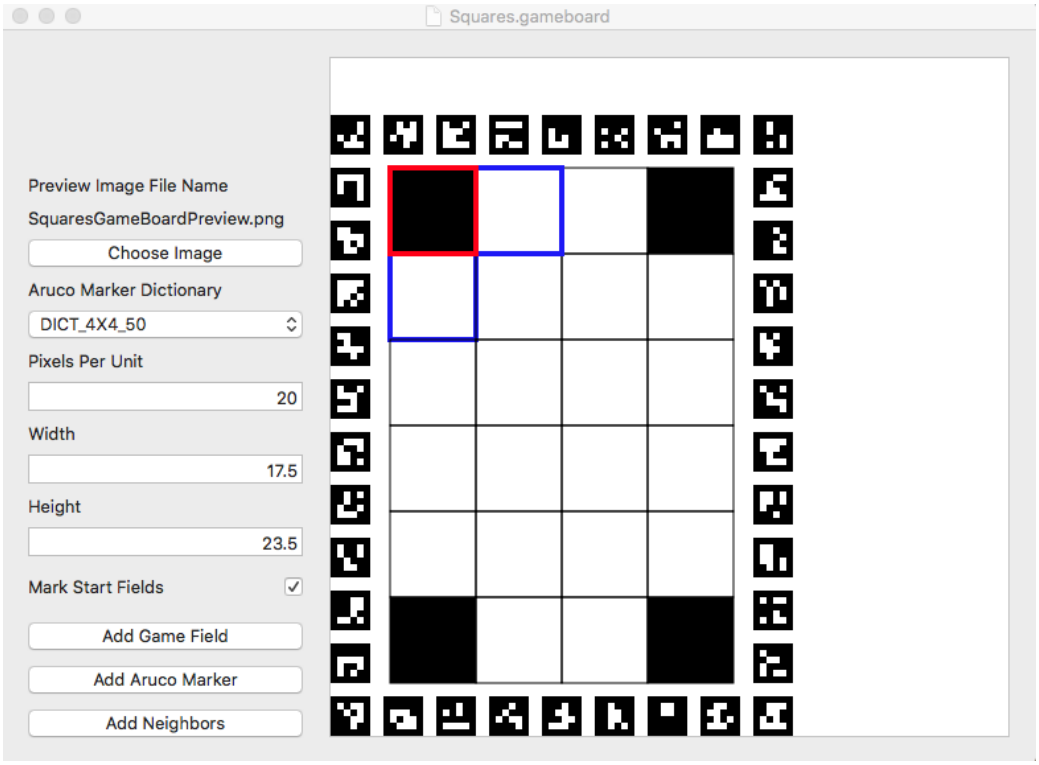


Abbildung 5.4: Screenshot aus dem Editor

Der Editor wurde für macOS entwickelt und bietet folgende Features an:

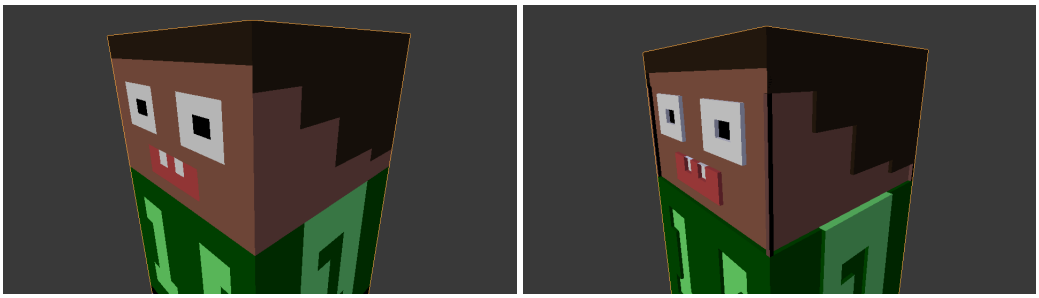
1. Serialisierung von Spielbrettern
   - Wählen eines Vorschaubildes für das Auswahlmenü der Spielbretter
   - Platzieren von Spielfeldern, durch Eingabe der Eckpunkte und gewünschter Anzahl
   - Löschen einzelner Spielfelder
   - Markieren von Spielfeldern, auf den eine Spielfigur starten kann
   - Bestimmen der Nachbarn einzelner Spielfelder
   - Platzieren von Aruco Markern, durch Eingabe der Eckpunkte und gewünschter Anzahl

- Löschen einzelner Aruco Marker
- Auswahl des Formats der Aruco Marker
- Bestimmen der Auflösung des Spielbrettes
- Ändern der Große des Spielbrettes
- Exportieren des Spielbrettes als PNG

2. Serialisierung von Spielfiguren
   - Wählen eines Vorschaubildes für das Auswahlmenü der Spielfiguren
   - Wählen eines 3D-Modells, das von VisionLib für das Tracking verwendet wird
   - Wählen eines 3D-Modells, das die virtuelle Spielfigur mit ihren Erweiterungen darstellt

5.5 Spielfigur

Wie bereits angedeutet, können für das Tracking mit VisionLib und für die visuelle Erweiterung zwei unterschiedliche 3D-Modelle verwendet werden. Das hat den Grund, dass das Model Tracking von VisionLib sich hauptsächlich an der Kanten orientiert, die im 3D-Modell enthalten sind. Gemusterte Texturen auf der Oberfläche der 3D-Modelle werden nicht automatisch berücksichtigt. Um auch die Textur auf der Oberfläche mit in das Tracking einzubeziehen, müssen kleine Kanten im Muster ergänzt werden. Dazu wurden entsprechende Konturen im Muster leicht in das 3D-Modell eingedrückt oder hervorgehoben, sodass kleine Stufen oder Fugen entstehen.



(a) 3D-Modell ohne zusätzliche Kanten (b) 3D-Modell mit zusätzlichen Kanten

Abbildung 5.5: 3D-Modell einer Spielfigur

Außer diesen 3D-Modellen gehört, zu jeder Spielfigur ein Vorschaubild, das bei der Erstellung einer Spielpartie angezeigt wird, damit die entsprechende Spielfigur ausgewählt werden kann.

5.5.1 Format

Zu jeder Spielfigur werden die Dateinamen der beiden 3D-Modelle und des Vorschaubildes serialisiert. Es werden lediglich die Dateinamen serialisiert, weil die komplexen Objekte, die das 3D-Modell zur Laufzeit in SceneKit repräsentieren, keine Serialisierung unterstützen. Das heißt das die 3D-Modelle und Vorschaubilder manuell in das App Bundle kopiert werden müssen, wenn neue Spielfiguren hinzugefügt werden sollen.

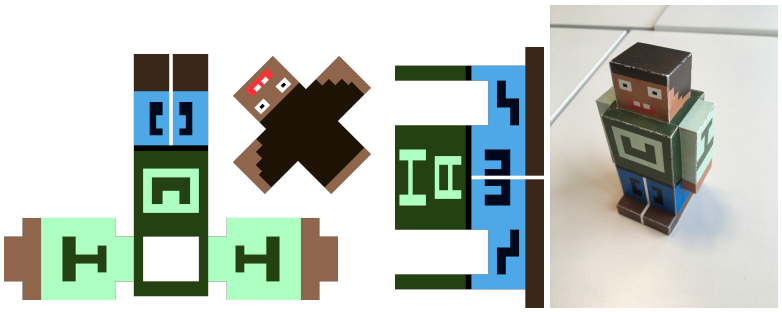
5.5.2 Ausdruck in 3D

Eine Möglichkeit ein reales Gegenstück des 3D-Modells, das als Spielfigur dienen soll, zu erzeugen, wäre der Ausdruck durch einen 3D-Drucker. Dabei ist zu beachten, dass die, für das Tracking mit VisionLib, relevanten Kanten am Ende deutlich sichtbar sind und einen entsprechenden Kontrast aufweisen.
Wenn für VisionLib ein eigenes 3D-Modell mit zusätzlichen Kanten erstellt wurde, ist es nicht unbedingt zielführend, dieses auch für den 3D-Druck zu verwenden. Da die ergänzten Kanten sehr filigran sein können und Produkte aus 3D-Druckern meist einfarbig sind, besteht die Gefahr, dass diese Kanten wieder mit dem restlichen Modell verschwimmen.
Alternativ könnte man die gedruckten Spielfiguren anschließend, entsprechend dem originalen 3D-Modell, kolorieren. Oder man druckt die Texturen auf Aufklebern, die man hinterher am 3D-Druck anbringt. Wie es bereits bei vielen Spielwaren gehandhabt wird.

5.5.3 Ausdruck auf Papier

Eine andere Möglichkeit das 3D-Modell in die Realität zu übertragen, ergibt sich durch das Paper Model Add-on von Blender. Dieses Add-on ermöglicht eine neue Exportfunktion, mit der ein 3D-Modell als Papiermodell exportiert werden kann.
Für diesen Export werden am 3D-Modell Schnittkanten gesetzt. Anhand der Schnittkanten wird das 3D-Modell dann in mehrere Fragmente aufgeteilt. Diese Fragmente sind in dem exportierten Dokument auf zweidimensionaler Fläche abgebildet. Sodass das Dokument mit einem herkömmlichen Papierdrucker gedruckt werden kann. Die Fragmente können ausgeschnitten, gefaltet und wieder zu der ursprünglichen Form zusammengeklebt werden.

Dieses Verfahren hat den Vorteil, dass die Texturen auf dem 3D-Modell direkt erhalten bleiben und nicht anschließend wieder aufgetragen werden müssen. Der Nachteil dabei ist, dass das verwendete 3D-Modell möglichst nur glatte Oberflächen, ohne hohen Detailgrad, enthalten sollte. Sonst wäre es sehr schwer, die auf Papier gedruckten Fragmente, anschließend wieder



(a) Fragmente (b) Figur

Abbildung 5.6: 3D-Modell von Paper Model Export

zum Ganzen zusammenzuführen.
Für Tests des Prototypen, mit sehr einfach gehaltenen Spielfiguren, hat sich diese Methode bewährt.

5.6 Spielbrett

Jedes Spielbrett besteht aus mehreren Spielfeldern, einem Aruco Board und dem Vorschaubild, das bei der Erstellung einer Spielpartie angezeigt wird. Die dreidimensionalen virtuellen Repräsentationen der Spielfelder, die für die Contact Detection nötig sind, werden erst zur Laufzeit beim Start einer Spielpartie generiert. Das Gleiche gilt für die verborgenen Inhalte der einzelnen Spielfelder. Die virtuellen Spielfelder werden auf Basis der zweidimensionalen Eckpunkte konstruiert.

5.6.1 Format

Jedes Spielfeld beinhaltet eine Liste der zweidimensionalen Eckpunkte und die Information, ob es sich um ein mögliches Startfeld handelt, auf dem zu Beginn einer Partie, eine Spielfigur platziert werden kann. Außerdem eine Liste der benachbarten Spielfelder, die mit dem nächsten Schritt erreicht werden könnten.
Das Aruco Board besteht aus mehreren Aruco Markern und der ID des Dictionary, in dem das Format der Marker definiert ist. Jeder Aruco Marker besteht aus einem Rechteck, welches die zweidimensionalen Eckpunkte des Markers repräsentiert und der ID, die den einzelnen Marker innerhalb des zugehörigen Dictionary identifiziert.
Zu jedem Spielbrett wird neben den enthaltenen Spielfeldern und dem Aruco Board, ebenfalls

der Dateiname der Vorschaubildes serialisiert.

5.6.2 Ausdruck

Durch der im Editor enthaltenen Funktion für den Export als PNG, können die Spielbretter einfach ausgedruckt werden. Zu beachten ist dabei nur, dass das gedruckte Spielbrett der im Editor angegebenen Größe entspricht. Außerdem ist zu berücksichtigen, dass die Maßeinheit, mit der im Editor gearbeitet wurde, mit der Maßeinheit, in der die Spielfigur vorliegt, übereinstimmen muss.
Das exportierte Spielbrett kann selbstverständlich, vor dem Druck, noch mit anderen Programmen bearbeitet werden. So könnte man das Spielbrett gegebenenfalls noch ansprechender gestalten, indem beispielsweise ein Hintergrund hinzugefügt wird, der zum Thema passt.

6 Evaluation

6.1 Funktionalität

Damit sich die Spieler ungestört auf das Spiel konzentrieren können, ist ein stabiles Tracking sehr wichtig. Wenn das Tracking nicht zuverlässig funktioniert und die Objekt nicht richtig erkannt werden, kann das zu Frustration bei den Spieler führen. Was dem Spielerlebnis natürlich sehr schaden kann. Daher wird im Folgen die Funktionalität der verwendeten Tracking-Verfahren beleuchtet.

6.1.1 Tracking des Aruco Board

Performance

Das Tracking der Aruco Marker hat sich als sehr stabil erwiesen. Sobald sich nach einer hastigen Bewegung die Unschärfe wieder gelegt hat, wird das Aruco Board unmittelbar wieder erkannt. Voraussetzung dafür ist natürlich, dass sich genügend Marker im Blickfeld der Kamera befinden. Wobei in der Regel schon zwei bis drei sichtbare Aruco Marker ausreichen, um ein stabiles Tracking zu erhalten. Je mehr die Kamera bewegt wird, desto unpräziser wird die Positionserkennung.
Bei einem ruhigen Kamerabild, stehen auch bei schlechten Lichtverhältnissen, die Chancen immer noch gut, dass die Position genau erkannt wird.

Einschränkungen

Die größte Einschränkung ist natürlich durch den Einsatz der Marker gegeben. Zum Einen kann das Tracking, bei der Verwendung weniger Marker, empfindlich gestört werden, wenn ein Teil der Marker verdeckt wird. Zum Anderen können die Gestaltungsmöglichkeiten des Spielbrettes von den Markern sehr behindert werden. Auch die Ästhetik der Marker an sich, passt nicht unbedingt zu jedem Spiel.

6.1.2 Tracking der Spielfiguren

Performance

Für das Tracking der Spielfigur sind gute Lichtverhältnisse erforderlich. Dafür funktioniert das Model Tracking von VisionLib sehr gut, wenn das Licht stimmt. Die Erfassung der Spielfigur kann sehr schnell passieren. Allerdings hängt die Erfassung auch stark von den Fähigkeiten des Benutzers ab. Jemanden mit Erfahrung kann die Erfassung in unter zwei Sekunden gelingen. Wo hingegen jemanden, der wenig Übung mit technischen Geräten hat, es sehr schwer fallen kann, das virtuelle Objekt an die Position des realen Objektes zu bewegen. Wenn das nicht gelingt, hat VisionLib keine Chance das Objekt zu erfassen.
Ist die Spielfigur erstmal erfasst, kann sie in der Regel mit einer sehr hohen Genauigkeit erweitert werden. Das virtuelle Modell liegt dann haargenau auf dem Realen. Dabei kann die Kamera auch in einem zufriedenstellenden Rahmen bewegt werden, ohne das die Verfolgung abreißt. Bei guten Voraussetzungen, kann die Kamera sogar auch mal zwei Meter von der Spielfigur entfernt sein.
Ein Phänomen, das gelegentlich auch aufgetreten ist war, dass während der Erfassung, scheinbar versucht wurde, die Spielfigur irgendwo in der Umgebung zu erkennen. Das trat häufiger bei Spielfiguren auf, die eine verhältnismäßig einfache Form hatten. Ergänzend muss noch betont werden, dass bei den Experimenten das Optimierungspotenzial, welches sich durch die in Kapitel 5.2.1 erwähnte Tracking Configuration ergibt, höchstwahrscheinlich nicht voll ausgeschöpft wurde. Zum Beispiel kann das Tracking von ARKit unterstützt werden, wodurch das Objekt auch schneller wiedergefunden werden kann, wenn es mal vom Tracking verloren wurde.

Einschränkungen

Der größte Nachteil beim Tracking der Spielfigur ist die notwendige Erfassung. Durch das dabei erforderliche Geschick, kann eine Barriere entstehen, die bestimmte Nutzergruppen ausschließt.
Die zweite Einschränkung ist, dass immer nur eine einzelne Spielfigur verfolgt werden kann. Das war für den Prototypen kein großes Problem, da das Game Design entsprechend ausgerichtet wurde. Trotzdem würden sich viele neue Möglichkeiten ergeben, wenn mehrere Objekte gleichzeitig verfolgt werden könnten.

6.2 Nutzerstudie

Um zu sehen wie Spieler in der Praxis mit dem Prototypen interagieren, wurde eine Stichprobe an Nutzern, mit unterschiedlich höher Affinität zu Technik, beim Spielen beobachtet. Als Testgerät erhielt jeder Spieler ein iPad. Was sich dabei ergeben hat wird im Folgenden ausgeführt.

6.2.1 Beobachtung

Die meisten Nutzer waren sehr fasziniert von der funktionierenden Erweiterung des Spielbrettes und der Spielfigur. Keiner schien große Vorbehalte gegenüber dem neuartigen Konzept zu haben. Sodass alle offen für die neue Erfahrung waren.
Bei der Beobachtung der Spiele, wurden jedoch verschiedene Schwachpunkte des aktuellen Konzeptes offenbart. Erstens hat sich gezeigt, dass Einigen die Erfassung der Spielfigur, doch recht schwer gefallen ist. Manchen Nutzern war nicht klar, was genau zu tun ist, um die Spielfigur zu erfassen. Dabei ist auch aufgefallen, dass die relativ großen iPads hinderlich sein konnten, wenn der Nutzer versucht hat die richtige Position zu finden. Außerdem war es, durch die großen Displays, schwer den, nur sich selbst gedachten, Hinweis wirklich geheim zu halten. Beziehungsweise schienen sich manche auch gar nicht darum zu bemühen, da nach erfolgter Erfassung ungeduldig der Hinweis aufgedeckt wurde. Auch wenn der Gegner grade das eigene Display einsehen konnte.
Ein weiteres Problem war, dass sich die Spielfiguren oft, auf dem relativ kleinen Spielbrett, gegenseitig verdeckt haben. Das hat dazu geführt, dass nach Möglichkeit oft auf Spielfelder, die sich in der Nähe der gegnerischen Spielfigur befanden, verzichtet wurde. Auch das Gewicht des Gerätes schien bei längeren Partien unangenehm zu werden. Da das iPad, wegen dem schnellen Wechsel zwischen den Spieler, kaum zwischendurch abgelegt wurde.
Es waren auch ein paar Nutzer dabei, denen die Erfassung sehr leicht gefallen ist, sodass flüssig gespielt werden konnte. Außerdem ist noch aufgefallen, dass die Spieler während der Partie tendenziell wenig kommuniziert haben. Was im Spiel leider auch nicht gefordert war.

6.2.2 Auswertung

Früh war klar, dass eindeutige Anweisungen gefehlt haben, die den Nutzern vermitteln, wie die Spielfigur erfasst werden muss. Es wäre wahrscheinlich auch besser gewesen, nicht einfach nur ein virtuelles Abbild der realen Spielfigur, als Orientierung für die Erfassung, zu verwenden. Passender wäre vielleicht, nur ein paar Kleidungsstücke oder Accessoires einzublenden, die der Spielfigur dann vom Spieler anlegt werden müssen.

Zweitens wären wohl Smartphones handlicher gewesen, als die relativ sperrigen Tablets. Die großen Geräte schienen am Ende doch recht unpraktisch und zu schwer, um sie in einer längeren Partie ständig in den Händen zu halten. Auch wenn der Vorteil, der sich durch das größere Display ergibt, noch ausbaufähig ist.
Dem Problem, der sich gegenseitig verdeckenden Spielfiguren, könnte man relativ leicht entgegen wirken, indem man Spielbretter mit größeren Spielfeldern erstellt. Wobei da auch immer abgewogen werden sollte, wie viele Spielfelder zu Verfügung stehen sollen und wie groß das gesamte Spielbrett werden darf. Denkbar wäre auch die kleineren Spielfelder beizubehalten und durch das Game Design zu verhindern, dass zwei Spielfiguren zu dicht beieinander stehen dürfen.
Außerdem wäre erstrebenswert die Kommunikation und unter den Spielern mehr zu fordern. Wofür das Spiel vielleicht umfangreicher konzeptioniert werden müsste. Das prototypische Spiel war im Allgemeinen noch zu unausgereift, um das ganze Potential von Augmented Reality für Brettspiele zu entfalten. Allerdings hat sich schon angedeutet, dass der grundlegende Ansatz nicht verworfen werden sollte. Vor allem wenn man die vielen neuen Möglichkeiten bedenkt, die sich durch die Kombination mit Augmented Reality ergeben können.

Literaturverzeichnis

[Andersen u. a. 2004] Andersen, Troels L. ; Kristensen, Sune ; Nielsen, Bjørn W. ; Grønbæk, Kaj: Designing an Augmented Reality Board Game with Children: The BattleBoard 3D Experience. In: *Proceedings of the 2004 conference on Interaction design and children: building a community* ACM (Veranst.), URL http://doi.acm.org/10.1145/1017833.1017858, 2004, S. 137–138

[Azuma 1997] Azuma, Ronald T.: A Survey of Augmented Reality. In: *Presence: Teleoperators & Virtual Environments* 6 (1997), Nr. 4, S. 355–385

[Bradski und Kaehler 2008] Bradski, Gary ; Kaehler, Adrian: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 2008

[Dörner u. a. 2014] Dörner, Ralf ; Broll, Wolfgang ; Grimm, Paul ; Jung, Bernhard: *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. Springer-Verlag, 2014

[Garrido-Jurado u. a. 2014] Garrido-Jurado, Sergio ; Muñoz-Salinas, Rafael ; Madrid-Cuevas, Francisco J. ; Marín-Jiménez, Manuel J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. In: *Pattern Recognition* 47 (2014), Nr. 6, S. 2280–2292. – URL http://www.sciencedirect.com/science/article/pii/S0031320314000235. – ISSN 0031-3203

[Garrido-Jurado u. a. 2016] Garrido-Jurado, Sergio ; Muñoz-Salinas, Rafael ; Madrid-Cuevas, Francisco J. ; Medina-Carnicer, Rafael: Generation of fiducial marker dictionaries using mixed integer linear programming. In: *Pattern Recognition* 51 (2016), S. 481–491. – URL http://www.sciencedirect.com/science/article/pii/S0031320315003544. – ISSN 0031-3203

[Giebler-Schubert u. a. 2013] Giebler-Schubert, Anke ; Zimmerer, Chris ; Wedler, Thomas ; Fischbach, Martin ; Latoschik, Marc E.: Ein digitales Tabletop-Rollenspiel für Mixed-Reality-Interaktionstechniken. In: *Virtuelle und Erweiterte Realität* 10 (2013), S. 181–184

[Huynh u. a. 2009] Huynh, Duy-Nguyen T. ; Raveendran, Karthik ; Xu, Yan ; Spreen, Kimberly ; MacIntyre, Blair: Art of Defense: A Collaborative Handheld Augmented Reality Board Game. In: *Proceedings of the 2009 ACM SIGGRAPH symposium on video games* ACM (Veranst.), URL http://doi.acm.org/10.1145/1581073.1581095, 2009, S. 135–142

[Lee u. a. 2005] Lee, Wonwoo ; Woo, Woontack ; Lee, Jongweon: TARBoard: Tangible Augmented Reality System for Table-top Game Environment. In: *2nd International Workshop on Pervasive Gaming Applications, PerGames* Bd. 5, URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.524.6142, 2005

[Milgram u. a. 1995] Milgram, Paul ; Takemura, Haruo ; Utsumi, Akira ; Kishino, Fumio: Augmented Reality: A class of displays on the reality-virtuality continuum. In: *Telemanipulator and telepresence technologies* Bd. 2351 International Society for Optics and Photonics (Veranst.), 1995, S. 282–293

[Molla und Lepetit 2010] Molla, Eray ; Lepetit, Vincent: Augmented Reality for Board Games. In: *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on* IEEE (Veranst.), URL https://doi.org/10.1109/ISMAR.2010.5643593, 2010, S. 253–254

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 30. Mai 2018 Ruben Christian Buhl