HAW HAMBURG

Project Report

Nadia Hintze

Entwicklung einer Pipeline zur Erweiterung eines Brettspiels um eine virtuelle Spieler-KI in AR

Betreuung durch: Prof. Dr. Kai von Luck
Eingereicht am: 07.02.2023

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Inhaltsverzeichnis

1 Einleitung

Videospiele erfreuen sich größter Beliebtheit. Neben dem boomenden Videospiel-Markt haben auch Brettspiele ihren Reiz nicht verloren und verzeichnen seit Jahren stabile Verkaufszahlen [9]. In der Zeit der Corona-Pandemie hat der gesamte Spielemarkt sogar ein zusätzliches Wachstum verzeichnet.

In der Welt der Brettspiele gibt es einen Trend dahin, Spiele, die ursprünglich für mehrere Spieler konzipiert wurden, auch für einzelne Personen spielbar zu machen. Dadurch wird aus der Sicht der einzelnen Person Flexibilität gewonnen. So muss sie bei der Auswahl des Spiels oder des Zeitpunktes keine Rücksicht auf andere Mitspieler nehmen. 2015 hatten rund 20% aller Brettspiele einen Solo-Modus mit einer seit 2006 stetig steigenden Tendenz. Davor waren es ca. 8-10%. [17]

Ein solcher Einzelspieler Modus kann u.a. realisiert werden, indem Mitspieler durch künstlich festgelegte Regeln imitiert werden. Der Spielablauf verändert sich in diesem Fall also nicht, da der Spieler nun mit künstlichen anstelle von realen Mitspielern interagiert. Ein solcher künstlicher Mitspieler wird auch *Automa* genannt. Das Verhalten eines Automas kann dem Spieler vermittelt werden, indem schriftliche Anweisungen auf physischen Spielkarten oder in einer App festgehalten werden. Ein Beispiel dafür bietet das Brettspiel „Scythe“ mit seiner Helper App „ScytheKick“ [4].

Im Rahmen der vorliegenden Arbeit und darauffolgender Projekte soll eine solche Automa-KI in Augmented Reality für den Spieler eingeblendet werden. Augmented Reality bezeichnet die vereinte Darstellung von virtuellen und realen Objekten, sodass sie wirken als befänden sie sich gemeinsam in der realen Umgebung (siehe Abb. 1) [15]. In dieser Arbeit sollen die virtuellen Objekte den künstlichen Mitspieler und seine Aktionen abbilden. Als Brettspiel wird hier Schach verwendet, da es einige verwandte Arbeiten gibt, die ebenfalls Schach als Anwendungsfall ausgewählt haben und so ein Vergleich damit stattfinden kann (siehe Kap. 4). Schach ist jedoch bloß ein Beispiel für ein Spiel in dessen Rahmen das Konzept benutzt werden kann.

Durch ein solches AR-Interface soll der Benutzer alle Vorteile eines Solo-Modus auskosten können, ohne dabei auf den unverfälschten Spieleablauf einer gängigen Partie verzichten zu müssen. Somit muss der Spieler sich nicht auf neue Regeln und Bedingungen einstellen, um den Einzelspieler Modus nutzen zu können. Der Game Designer Morten Monrad Pedersen hat sich auf die Konzipierung von Einzelspieler Modi spezialisiert und diese Eigenschaft als ein gutes Ziel für Solo-Modi identifiziert [17]. AR wurde als tragende

Abbildung 1: Lagerichtige Einblendung eines virtuellen Fisches auf einem realen Fossil, Quelle: Hintze [11]

Technologie für die folgende Arbeit ausgewählt, da sie mit ihrer Konzentration auf die Realität die Möglichkeit bieten könnte, sich besonders gut in das Spielgefühl einzufügen und es so wenig wie möglich zu verändern. So soll das physische, haptisch erlebbare Spiel, das von dem Spieler gewünscht wurde, weiterhin im Vordergrund stehen. Außerdem bietet ein technischer Automa einige weitere Vorteile einer gedruckten Version gegenüber. Beispielsweise ist eine Anwendung sehr flexibel in ihrer Funktionalität. So ist denkbar, den Schwierigkeitsgrad der KI dynamisch während des Spielverlaufs anzupassen. Mithilfe einer Anbindung an das Internet können zudem weitere Funktionen implementiert werden, wie z.B. ein weltweiter Highscore oder die online Einbindung menschlicher Mitspieler, die durch einen AR-Avatar vertreten werden.

Die vorliegende Arbeit konzentriert sich auf den Entwurf einer Pipeline eines solchen AR-Systems. Dazu gehören die Konstruktion einer zugrundeliegenden Architektur sowie die Implementierung eines ersten Grundgerüsts.

Um dieses Ziel zu erreichen, werden in Kapitel 2 zunächst die Anforderungen an das System genauer bestimmt und groben Komponenten zugeordnet. In Kapitel 3 wird anschließend abgewogen, welche Technologien und Methoden zur Realisierung der Komponenten geeignet wären. Kapitel 4 bietet einen Überblick darüber, wie diese Technologien in verwandten Arbeiten eingesetzt wurden und dient der konzeptionellen Abgrenzung der eigenen Arbeit. Nachdem eine Auswahl getroffen wurde, welche Technologien in der eigenen Architektur berücksichtigt werden sollen, wird in Kapitel 5 die grundlegende Architektur entworfen und prototypisch umgesetzt. Abschließend wird in Kapitel 6 eine

kurze Zusammenfassung der Ergebnisse und ein Ausblick auf die kommenden Projekte gegeben.

2 Technische Anforderungen

Aus dem umzusetzenden Anwendungsfall (siehe Kap. 1) erschließen sich mehrere Anforderungen an das System. Diese Anforderungen können groben Komponenten des Systems zugewiesen werden.

1. **Darstellung**
   Es wird ein Display benötigt, das die virtuellen Objekte darstellen kann. Dabei liegt die besondere Aufgabe darin, die Eindrücke der Realität und der virtuellen Objekte so miteinander zu kombinieren, dass sie als eine gemeinsame Umgebung wahrgenommen werden. Diese Darstellung kann sich auf jeden der fünf menschlichen Sinne beziehen, wobei die optische Erweiterung, gefolgt von der akustischen, am häufigsten in AR-Applikationen Anwendung findet [15] [2]. Somit sind diese Displays besonders weit entwickelt und den Entwicklern sowie potenziellen Benutzern zugänglicher [22]. In diesem Projekt wird sich zunächst auf die optische Erweiterung beschränkt, da ein Schachspieler in einem regulären Spiel über diesen Weg einen Großteil der Informationsermittlung durchführt. Ausnahmen kann es beispielsweise bei einer Sehbehinderung geben. Um die potenzielle Benutzergruppe zu erweitern, ist in Folgeprojekten denkbar, die KI zusätzlich über akustische Signale agieren zu lassen, indem sie beispielsweise ihre Spielzüge ankündigt.

2. **Tracking**
   Damit die virtuellen Elemente sich in die Realität einfügen, müssen sie lagerichtig eingeblendet werden. Beispielsweise müssen die dreidimensionale Position und Rotation eines realen Tisches im Raum bestimmt werden, damit ein virtuelles Objekt so angezeigt werden kann, als stünde es darauf. Diese Lagebestimmung muss zudem kontinuierlich stattfinden, damit die virtuelle Darstellung einem sich zur Laufzeit ändernden Standpunkt des Benutzers angepasst werden kann [22]. Damit die Ansicht der virtuellen Objekte möglichst realitätsnah wirkt, ist es wichtig, dass die Aktualisierung häufig genug stattfindet. Deshalb darf der Tracking-Prozess nicht zu langwierig sein. Der Anwendungsfall der Brettspiel Erweiterung setzt voraus, dass der aktuelle Zustand aller relevanter Brettspiel Elemente bekannt ist.

3. **Interaktion**
   Azuma [2] definiert AR Anwendungen darüber hinaus zusätzlich über Interaktionen in Echtzeit. In diesem Fall besteht sie einerseits aus den Bewegungen des Benutzers, aus denen optische Anpassungen der virtuellen Objekte folgen. Andererseits soll die Spiel-KI Rückmeldungen dazu liefern, welchen Spielzug sie als nächstes tätigen möchte.

4. **Spiellogik**
   Die letzte Komponente trifft die Entscheidungen der Spiel-KI. Sie beinhaltet eine Abbildung der Spiellogik und wählt aufgrund des im Tracking festgestellten Brettspiel-Zustands Aktionen für den nächsten Spielzug.

3 Technologien

Im diesem Kapitel werden Technologien vorgestellt, die sich zur Umsetzung der Komponenten eignen. Auf dieser Basis wird ausgewählt, welche davon in diesem Projekt eingesetzt werden.

3.1 Darstellung

Sowohl optische als auch akustische Erweiterungen lassen sich mithilfe von Head-Mounted-Displays („HMD“) oder mobilen Endgeräten, wie z.B. Smartphones oder Tablets, umsetzen. In beiden Fällen beinhalten die Geräte eigene Kameras, die die reale Umgebung filmen können, ein Display, das diese Umgebung gemeinsam mit virtuellen Objekten darstellen kann und Lautsprecher zur Wiedergabe akustischer Signale. [12] [22]

Ein HMD wird wie eine Brille getragen. Das Display befindet sich dabei direkt vor den Augen. Durch dieses Setup muss der Benutzer das Gerät nicht in den Händen halten, sondern kann ungehindert mit der realen Umgebung interagieren oder die Anwendung durch Gestern steuern. Das ist ein Vorteil gegenüber der Smartphone Anwendung. Hier muss das Smartphone so gehalten werden, dass die Kamera relevante Teile des Umfelds filmt. Allerdings besitzt ein Großteil der Menschen bereits ein Smartphone [3]. Somit ist die Gruppe der potenziellen Benutzer sehr groß.

3.2 Tracking

Die räumliche Lage eines Objektes kann mithilfe unterschiedlicher Sensoren festgestellt werden. Einige werden zu sogenannten „Inertial Measurement Units“ (oder kurz „IMUs“) zusammengesetzt, in zu trackenden Objekten verbaut und messen Positions- oder Drehbewegungen direkt. Alternativ können auch GPS-Sensoren verwendet werden, um einen Eindruck von der globalen Position zu gewinnen. Die am häufigsten verwendete Methode setzt auf eine Beobachtung der Umwelt durch Kamerabilder. Diese Bilder können beispielsweise aus herkömmlichen RGB-Werten, Grauwerten oder Infrarot Messungen bestehen. [22]

Es gibt verschiedene algorithmische Ansätze, die die Pose eines Objektes aus einem solchen Bild erschließen. Eine Übersicht dieser Methoden haben Fan u. a. [6] verfasst. Ein Ansatz, der aktuell häufig Verwendung findet, ist der Einsatz von Convolutional Neural Networks („CNNs“) im Rahmen von Machine Learning [8]. Diese Unterkategorie neuronaler Netzwerke zeichnet sich durch einen Aufbau aus, der insbesondere zur Verarbeitung von Daten genutzt wird, die in einem topologischen Kontext betrachtet werden sollen. Zu solchen Daten gehören auch die Pixel eines Bildes. Ein weiterer Ansatz ist der direktere Vergleich des Kamerabildes mit Referenzdaten im Voraus bekannter, gesuchter Objekte. Beispielsweise können für die Suche von zweidimensionalen Bildern im realen Raum Algorithmen verwendet werden, die prägnante Punkte in den Bildern suchen. Nachdem diese Algorithmen sowohl auf das Kamerabild als auch auf das Referenzbild angewendet wurden, kann ermittelt werden, wie diese Bildpunkte sich relativ zueinander verschoben haben. Dadurch kann man die Position und Rotation in der realen Umgebung feststellen. Um diese prägnanten Bildpunkte zu gelangen, kann eine zusätzliche Vorverarbeitung des Bildes mit Filtern sinnvoll sein. [6]

Je nach Anwendungsfall können die Messungen einzelner Sensoren nicht ausreichen, um das gewünschte Ergebnis zu erzielen. Zum Beispiel funktioniert die Ortung über ein Foto nur, solange das zu untersuchende Objekt nicht von anderen verdeckt wird. Ein weiteres Beispiel ist, dass GPS-Signale nur eine relativ grobe Auflösung haben und keine Aussage über die vollen sechs Degree of Freedom eines Objektes treffen. Daher kann es sich anbieten, unterschiedliche Sensordaten miteinander zu kombinieren, um so ein exaktes Ergebnis zu erhalten. [22]

3.3 Interaktion

Für eine AR-Anwendung sind natürliche Interaktionen mit der realen Umgebung im Gegensatz zu anderen digitalen Systemen besonders kennzeichnend (siehe Kap. 2). So passt sich die Ansicht der digitalen Inhalte einerseits der Position des Benutzers an, aber auch andere Teile der Umgebung, wie z.B. die Figuren eines Schachspiels, können dafür getrackt werden. Dementsprechend ist eine wichtige Design-Entscheidungen, welche realen Objekte und welche Interaktionen mit ihnen für die Anwendung relevant sind und digital erfasst werden müssen.

Für Interaktionen mit virtuellen Inhalten werden geeignete Schnittstellen benötigt. Hier ist unter anderem die Wahl des Displays relevant. Der Benutzer eines HMDs hat im Gegensatz zum Smartphone-Nutzer beide Hände frei. So hat er mehr Spielraum, die Anwendung über erkannte Gesten oder mithilfe eines zusätzlichen Controllers zu steuern.[1] Auch die haptische Interaktion mit realen Objekten wird so erleichtert. Im Gegensatz dazu ist bei der Verwendung von Smartphones der zusätzliche Gebrauch des integrierten Touch-Displays naheliegend. Dafür kann die große Anzahl der Smartphone-Besitzer [3] von Vorteil sein. Yusof u. a. [29] haben in ihrer Arbeit ein solches AR-Interface verwendet und in ihrer Evaluation festgestellt, dass die Benutzer problemlos intuitiv mit den virtuellen Objekten darüber interagieren konnten, ohne weitere Anweisungen zu benötigen.[2] Da Gestensteuerung oder ein dedizierter Controller u.U. nicht so häufig im Alltag der Benutzer benutzt wird, könnte die Bedienung hier nicht so intuitiv ablaufen. Das könnte in einem der auf diese Arbeit folgenden Projekte genauer untersucht werden (siehe Kap. 6.2).

3.4 Entwicklungsumgebung

Die virtuellen AR-Elemente werden gängigerweise in Entwicklungsumgebungen erstellt, die auch in der Spieleentwicklung eingesetzt werden, da die Anforderungen sich oft überschneiden (vgl. Kap. 4). Welche Tools genau zum Einsatz kommen, kann darauf abgestimmt werden, auf welcher Hardware die Anwendung laufen soll. Die Engine Unity3D[3] bietet an, die erzielte Plattform für den Bau-Prozess der Software flexibel zwischen Android- oder iOS-Endgeräten und HMDs wechseln zu können [28]. Aufgrund

[1] Ein Beispiel für ein solches System ist das AR-Interface „TiltFive“ [25]. Hier wird die Lage des Controllers bestimmt, sodass mit ihm auf virtuelle Objekte gezeigt und so interagiert werden kann.
[2] Das Projekt wird näher in Kap. 4.1 vorgestellt.
[3] Im Folgenden mit „Unity“ abgekürzt.

dieser Flexibilität wird ein beträchtlicher Teil des Systems in Kap. 5 in dieser Umgebung entwickelt.

3.5 Systemarchitektur

Wie bislang festgestellt wurde, bringen unterschiedliche Displays und damit verbunden unterschiedliche Formen der Interaktion sowie auch unterschiedliche Tracking-Methoden je ihre eigenen Vor- und Nachteile mit sich. In Folgeprojekten soll der Einsatz verschiedener Technologien getestet werden. Daher soll die hier entwickelte Architektur den flexiblen Einsatz der Technologien unterstützen. Zu diesem Zweck wird in dieser Arbeit besonderer Wert auf Modularität und eine lose Kopplung einzelner Komponenten untereinander gelegt, sodass diese austauschbar bleiben.
Zunächst muss daher überlegt werden, wie eine in Unity entwickelte Anwendung strukturiert werden kann. Innerhalb von Unity wird alles nativ entsprechend dem Entity-Component-System-Pattern aufgebaut. Jedes Objekt im virtuellen, dreidimensionalen Raum ist eine Entität und wird in der Engine „GameObject“ genannt. Jedem Objekt können Komponenten zugewiesen werden. Das sind hier C#-Skripts, die alle Daten und Funktionen beinhalten. Indem die Objekte in eine gemeinsame Hierarchie eingeordnet werden, können die Komponenten verschiedener Objekte aufeinander zugreifen. Da virtuelle Umgebungen sehr komplex werden können, wenn sie eine Vielzahl an Objekten beinhalten, können alle Querverweise zwischen ihnen und den Komponenten unübersichtlich werden. Dazu trägt auch bei, dass Objekt Referenzen sowohl über Skripts als auch in der grafischen Oberfläche per Drag-and-Drop definiert werden können. Zudem gibt es innerhalb von Unity Systeme, die die Eigenschaften der Objekte zusätzlich verändern können. Ein Beispiel dafür ist ein Physik-System, das z.B. Objekte so im Raum bewegen kann als beeinflusse sie die Schwerkraft. Auch das trägt zur Komplexität bei. [27]
Dieser Umstand kann es erschweren, einzelne Elemente auszutauschen. Das im Rahmen dieser Arbeit entwickelte System soll jedoch wie zuvor beschrieben genau dies unterstützen. Eine Möglichkeit, den Code strukturierter zu gestalten, ist die Orientierung am Model-View-Control Pattern. Dieses Pattern teilt eine Anwendung in die drei namensgebenden Komponenten auf. Das Model beinhaltet den Umgang mit den rohen Daten, der View implementiert das Benutzer Interface und der Controller trifft alle logischen Entscheidungen zur Laufzeit [7]. In Kap. 5 wird dieses Grundkonzept auf die mit Unity erstellte Software angewendet.

Wie vorab erläutert wurde, kann es sinnvoll sein, unterschiedliche Tracking-Methoden miteinander zu kombinieren. Jedoch ist schon die Auswertung eines Kamerabildes für sich genommen relativ aufwändig, da die Daten bei einer hohen Bildauflösung umfangreich werden. So sind beispielsweise CNNs auch unter den Neuronalen Netzen besonders aufwändig zu berechnen [16]. Um nicht auf die begrenzten Rechenkapazitäten des Endgeräts beschränkt zu sein, kann man Teile des Systems auf einen Server auslagern und die nötigen Daten mithilfe einer Netzwerkverbindung übermitteln. Auch hier gibt es verschiedene Wege, diese Kommunikation zu implementieren. Eine davon ist die Verbindung über TCP-Sockets. Dafür gibt es im .Net-Framework von Microsoft eine passende Funktion namens „TcpClient“ [14]. Alternativ unterstützt die Bibliothek „Networking“ von Unity auch die Nutzung einer REST-Api [26]. Da zu den Anforderungen jedoch eine ausreichende Performanz zählt (siehe Kap. 2) und REST-Kommunikation einen, wenn auch kleinen, Overhead hat, wurde sich für eine Kommunikation via Sockets entschieden. Es existieren auch Plugins, die bereits viele Funktionalitäten mitbringen, die insbesondere für große Multiplayer Spieler mit vielen Teilnehmern ausgelegt sind. Ein Beispiel dafür ist das Plugin „PUN“ [18]. Jedoch wäre hier der potenzielle Overhead des Plugins noch weitaus größer.

4 Verwandte Arbeiten

Anhand verwandter Arbeiten wird nun gezeigt, wie die beschriebenen Technologien für den hier verfolgten Anwendungsfall eingesetzt werden können. Die Evaluationen dieser Arbeiten sollen dabei helfen, den Aufbau des Systems in Kap. 5 weiter herauszuarbeiten. Zudem wird das hier verfolgte Konzept von den anderen Arbeiten in diesem Kapitel abgegrenzt. Als konzeptionell vergleichbar werden Arbeiten betrachtet, die Brettspiele durch Augmented Reality Anwendungen mit dem Ziel erweitern, damit das Verhalten eines Spielers zu ersetzen.

4.1 Vorstellung der Arbeiten

Yusof u. a. [29] haben ein System entwickelt, dass es Benutzern ermöglicht mit einem Schachspieler, der sich an einem anderen Ort befindet, gemeinsam aus der Ferne zu spielen. Hierfür haben sie mit Unity3D eine Smartphone-Anwendung konstruiert. Mithilfe der Bibliothek Vuforia wird ein im Voraus bekanntes, reales, extra für diese Anwendung

platziertes Bild („Marker“) getrackt und darauf lagerichtig ein rein virtuelles Schachspiel platziert. Das Spiel wird ausschließlich über das Touch-Display des Smartphones bedient. Die Synchronisierung der Spielfelder beider Spieler passiert durch einen zentralen Server, der mithilfe des Unity-Plugins PUN eingerichtet wurde.

Günther u. a. [10] setzen ebenfalls ein Schachspiel für zwei voneinander entfernte Spieler in Unity3D um und nutzen einen zentralen Server zur Synchronisation. Diese Arbeit unterscheidet sich von der von Yusof u. a. [29] in der Hinsicht, dass anstelle rein virtueller Spielfiguren auch reale zum Einsatz kommen. Das Schachbrett wird auf einem Tablet angezeigt, das eine Touch-Funktion mitbringt. Dadurch, dass die Spielfiguren des lokalen Spielers aus einem konduktiven Material bestehen, kann ihre Position auf dem Display und somit auf dem Spielbrett ermittelt werden. Das Schachbrett-Muster fungiert gleichzeitig als Marker für das Tracking durch Vuforia. Auf diese Weise können die gegnerischen Spielfiguren über ein HMD virtuell lagerichtig eingeblendet werden.

Auch bei Rayar u. a. [21] wird das Schachbrett auf einem zusätzlichen Display dargestellt und als Vuforia-Marker verwendet, während die Spielfiguren des lokalen Spielers real sind. Jedoch ist das Display hier Teil einer Tischoberfläche. Diese wurde außerdem mit Infrarot-Sensoren ausgestattet. Auf diese Weise können Marker, die unter den Spielfiguren befestigt werden, auf der Tischoberfläche lokalisiert werden. So kann auch hier der aktuelle Zustand des Spiels festgestellt werden, sodass die virtuell auf einem HMD angezeigten gegnerischen Spielfiguren darauf reagieren können. Im Gegensatz zu den vorherigen beiden Systemen ist hier der Gegenspieler eine künstliche Intelligenz.

„ScytheKick“ ist eine iOS-Anwendung aus dem kommerziellen Gebrauch, die zu dem Brettspiel „Scythe“ gehört. Auch hier soll der Benutzer gegen einen Automa spielen. Anders als bei den vorherigen Beispielen werden hier jedoch sowohl das reale, reguläre Spielbrett, als auch die Spielfiguren von der Tracking-Bibliothek ARKit erkannt. Anschließend werden die vom Automa gewünschten Spielzüge über das Smartphone mitgeteilt, sodass die realen Spieler diese stellvertretend durchführen können. Die Entwickler von ScytheKick haben die Entwicklungsumgebung XCode und zur Gestaltung der virtuellen Inhalte die dazugehörige API SceneKit verwendet. [23]

4.2 Abgrenzung der eigenen Arbeit

Hinsichtlich der einzelnen Technologien sind diese Arbeiten sehr ähnlich zu dem Ziel der vorliegenden Arbeit aufgebaut. Auch in diesem Projekt sollen zur Darstellung HMDs oder

mobile Endgeräte, zur Interaktion Touch-Displays oder das Tracking realer Aktionen und als Entwicklungsumgebung Unity3D verwendet werden. Einen wesentlichen Unterschied gibt es jedoch in der Vorgehensweise beim Tracking. Die verwandten Arbeiten haben dafür die Bibliotheken Vuforia und ARKit verwendet. Beide sind in ihrem funktionalen Umfang vergleichbar. Sie bieten ein Tracking von vorher bekannten, zwei- oder dreidimensionalen Objekten über ein Kamerabild an. Dabei ist insbesondere das Tracking von zweidimensionalen Bildern robust [29, 10], während das Tracking von dreidimensionalen Objekten schneller an seine Grenzen stößt [11]. Zudem sind beide Bibliotheken in der Anzahl gleichzeitig auf einem Kamerabild erkennbarer Objekte begrenzt. ARKit unterstützt weder das Tracking mehrerer gleich aussehender Objekte, noch das von mehr als vier Referenzobjekten im selben Kamerabild [1]. Auch Vuforia empfiehlt maximal fünf Objekte gleichzeitig zu tracken, da die Rechenkapazitäten auf dem Endgerät begrenzt sind [19]. Um diese Begrenzungen umgehen zu können, wird im Kap. 5 ein dedizierter Server eingerichtet, der das Tracking mithilfe aufwändigerer Methoden mit umfangreicheren Rechenkapazitäten durchführen kann.

Auf konzeptioneller Ebene kommt die Anwendung *ScytheKick* dem hier verfolgten Ziel am nächsten. Das liegt zum einen an dem Automa als Gegenspieler, aber auch daran, dass hier vornehmlich die realen, herkömmlichen Spiel-Objekte zur Interaktion verwendet werden. In den anderen Arbeiten wird entweder das komplette Spielfeld virtuell dargestellt [29], was aus unserer Sicht die Frage aufwirft, wo der Vorteil gegenüber einer rein virtuellen Anwendung ohne AR-Aspekt liegt, oder sie ersetzen ihr Spielfeld durch zusätzliche Displays und knüpfen ihre Spielfiguren ebenfalls an künstlich geschaffene Bedingungen, wie die Verwendung von Markern oder bestimmten Materialien. Insbesondere das in den Tisch eingelassene Display von Rayar u. a. [21] schafft damit eine starke, örtliche Abhängigkeit. Somit ist man weniger flexibel bei der Verwendung des Systems. Zudem ist hier das Tracking der Spielfiguren auf die zweidimensionale Position auf dem Schachbrett begrenzt. Das reicht im Fall von Schach zwar für die Auswertung des Spielzustands aus, allerdings ist eine virtuelle Erweiterung bei Bewegungen in der dreidimensionalen Ebene damit nicht möglich. Günther u. a. [10] haben in ihrer Evaluation außerdem festgestellt, dass die Verwendung realer Objekte stark zur Immersion beiträgt und sich die virtuellen Elemente somit in der Wahrnehmung des Benutzers besser in die Realität einfügen.
In unserer Anwendung, soll der möglichst unveränderte Ablauf des realen Brettspiels trotz Automa Einbindung im Vordergrund stehen. Aus den Quellen zu *ScytheKick* wird nicht ersichtlich, wie viel Input der Benutzer im AR-Modus parallel in der Smartphone-App und auf dem Spielbrett tätigen muss [23, 4]. Wenn sich das zu stark überschneidet,

könnte aus unserer Sicht der Spielablauf zu stark vom üblichen Vorgehen abgelenkt werden. Neben dem zuvor erläuterten, weitreichenderen Tracking könnte auch diese Priorität ein Unterschied sein.

5 Umsetzung

Dieses Kapitel widmet sich dem Entwurf einer Architektur, die den zuvor in Kap. 3.5 genannten Zielen gerecht werden kann und die die Einbindung der ausgewählten Technologien unterstützt. Außerdem findet eine prototypische Implementierung der dazugehörigen Komponenten statt.

5.1 Aufbau des Tracking Servers

Die erste Design-Entscheidung, die getroffen wird, betrifft den Umfang des Tracking-Servers. Zu diesem Zweck wird zunächst definiert, welche Vorgänge das System grundlegend durchlaufen muss, um alle gewünschten Funktionalitäten abzudecken. Dieser Workflow wird in Abb. 2 dargestellt.

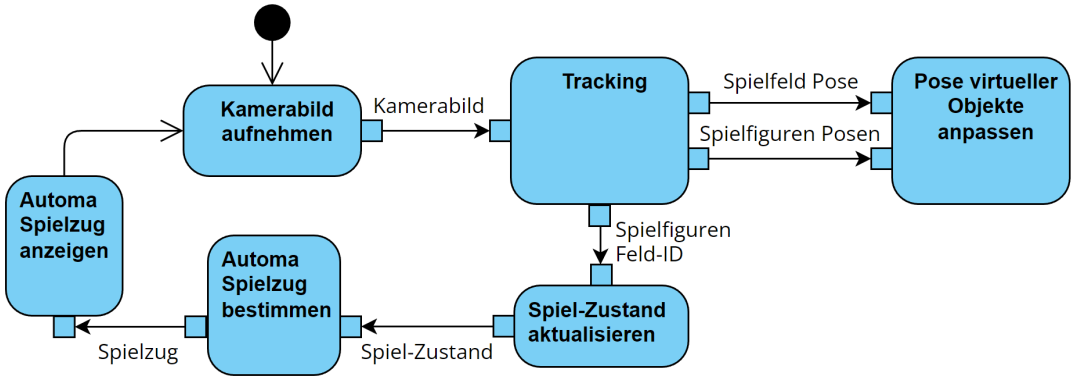

Abbildung 2: Workflow des gesamten Systems,
Quelle: Eigene Abbildung

Nun muss festgelegt werden, welche Teile dieser Vorgänge auf dem Server ausgeführt werden und welche Informationen dafür zwischen dem Server und der Unity Anwendung ausgetauscht werden müssen. Als mindeste Funktionalität soll der Server aus dem Kamerabild des Endgeräts die räumliche Lage des Spielbretts und der Spielfiguren relativ

zu dem Gerät bestimmen. Diese Funktionalität wurde hier erweitert, indem aus diesen Informationen auch das Feld auf dem Schachbrett ermittelt wird, auf dem eine Figur jeweils steht. Dadurch ergibt sich der in Abb. 3 dargestellt Informationsaustausch zwischen der Unity-Anwendung und dem Tracking-Server.

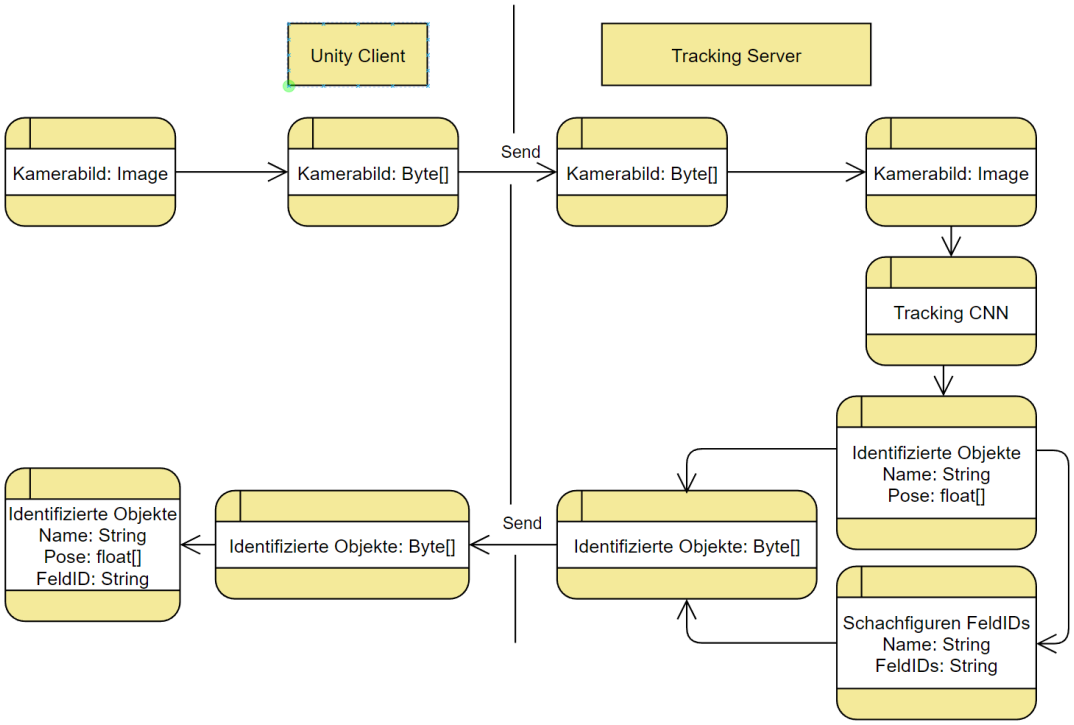

Abbildung 3: Datenaustausch zwischen Unity-Anwendung und Tracking-Server, Quelle: Eigene Abbildung

Um die Kommunikation zwischen den beiden Teilen des Systems zu etablieren, werden feste Nachrichtenformate benötigt, die beide verwenden. In diesem Projekt werden vier unterschiedliche Nachrichtenformate definiert. Die ersten zwei mit den Typen *InitUnity* und *InitServer* bestehen jeweils aus zwei Byte und signalisieren die erfolgreich aufgebaute Verbindung und die Bereitschaft der Kommunikationsteilnehmer zum Start der Anwendung. Die weiteren beiden Nachrichten dienen dem Austausch der für das Tracking benötigten Daten (siehe Abb. 4). Dieser Austausch findet einmal pro Frame des mit der Kamera aufgenommenen Videos statt.

Die Größe der Nachrichten kann je nach Anzahl erkannter Objekte variieren. Im Fall, dass alle Schachfiguren auf dem Feld stehen und von beiden Spielern getrackt werden, liegt die maximale Länge der Nachricht bei 1631 Byte. Die Länge der *CamImg* Nach-

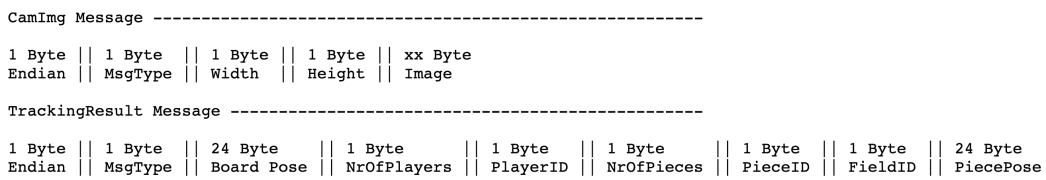

Abbildung 4: Format der verschickten Nachrichten, Quelle: Eigene Abbildung

richt hängt von der Größe des Kamerabildes ab. Für eine Kamera mit der Auflösung 4032x3024 Pixel, umfasst die Nachricht 36,56 Megabyte. Es bleibt zu Untersuchen, ob diese Nachrichtengröße mit der Frequenz einer Video-Aufnahme problematisch wird und die Performanz des Gesamtsystems einschränkt (siehe Ausblick in Kap. 6.2).

Der Server wurde mit der Programmiersprache Python verfasst und nutzt die Bibliothek Asyncio, um die TCP-Kommunikation zu handhaben. So wird ein asynchron wartender Server eingerichtet, der ausschließlich auf die Nachrichten des Unity-Clients reagiert [20]. Als Grundlage zur weiteren Entwicklung wurde ein Stub als Platzhalter für die Tracking Funktionen eingefügt, der zufällige Tracking-Ergebnisse liefert, sodass diese an die Unity-Anwendung geschickt werden können.

5.2 Aufbau der Unity3D Anwendung

Die zwei wesentlichsten Funktionen, die die Unity Anwendung aktuell mitbringt, sind die Netzwerkkommunikation, die Verwaltung des aktuellen Spielzustands und ein prototypisches User Interface. Das User Interface der Anwendung zeigt aktuell das Kamerabild der realen Umgebung an und blendet als Text darüber die aktuelle Instruktion der Automa K.I ein.

Wie in Kap. 3.5 erläutert wurde, soll das Model-View-Control-Pattern angewendet werden, um die Unity-Applikation zu strukturieren und einzelne Teile leichter austauschbar zu machen. Costa [5] hat eine leicht erweiterte Version des Patterns dafür vorgeschlagen, an der sich hier orientiert wird. Der Autor hat dafür zunächst alle anderen Unity-Objekte einem Root-Objekt namens *Application* untergeordnet. Dieses Objekt verwaltet alle Referenzen auf instanziierte Objekte. Da alle anderen Objekte von *Application* abgeleitet werden, haben sie darüber Zugriff auf die die Referenzen, ohne das die Instanzen unübersichtlich an unterschiedlichsten Stellen angelegt werden müssen. Eine weitere Funktion von *Application* ist das triggern von Methoden in dem Controller. Dafür werden Notifications genutzt, die die Methoden des Controllers mit einem identifizierenden String

triggern. Als Controller dient das Objekt *Control*. Es beinhaltet alle Vorgänge die, die Anwendungslogik betreffen. Dazu zählen hier beispielsweise die Schachspiel-Logik des Automas und die Durchführung der Netzwerkkommunikation. Diese dazugehörigen Methoden werden hier als Komponenten an das *Control* Objekt gehängt. Das Objekt *Model* beinhaltet dagegen Komponenten, die ausschließlich die permanenten Daten verwalten. Das ist hier beispielsweise der aktuelle Spiel-Zustand. Das letzte direkte Kind-Objekt von *Application* ist der *View*. Hier findet alles statt, dass sich direkt auf die virtuelle 3D-Welt bezieht. Daher hat *View* auch als einziges weitere Kind-Objekte, da hier Objekte in der Welt in der Hierarchie auftauchen müssen. Neben 2D- und 3D-Objekten zählt hier auch die virtuelle Kamera und das User Interface dazu. Die so entstandene Hierarchie an Objekten ist in Abb. 5 zu sehen.

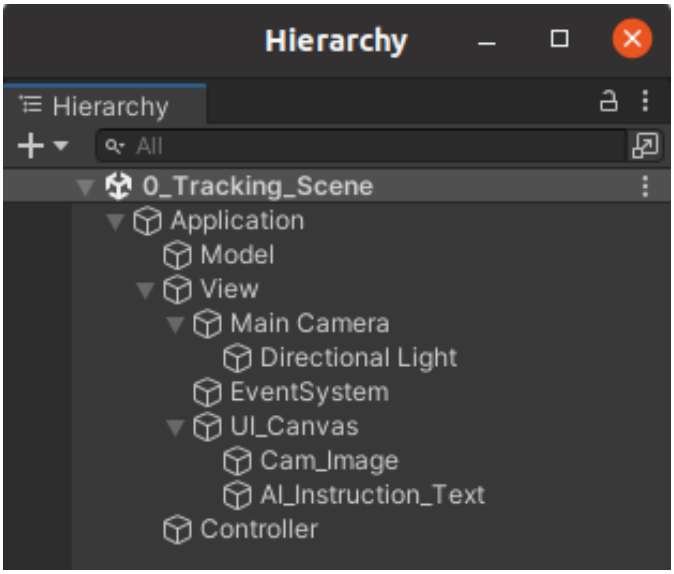

Abbildung 5: Hierarchie in Unity3D entsprechend dem Model-View-Control-Pattern, Quelle: Eigene Abbildung

6 Schluss

Zum Abschluss der Arbeit werden nun die Ergebnisse kurz zusammengefasst und ein kurzer Ausblick auf kommende Folgeprojekte gegeben.

6.1 Zusammenfassung

In dieser Arbeit wurde eine Architektur entwickelt, die den Einsatz verschiedener Technologien unterstützt. Durch die Verwendung der Entwicklungsumgebung Unity3D können zur Darstellung virtueller Elemente HMDs und mobile Endgeräte ohne große Anpassungen der Software eingesetzt werden. Die Gestaltung der Interaktion bleibt ebenfalls flexibel, da Unity3D Schnittstellen bietet, die sowohl die Gestenerkennung von HMDs [13] als auch die Bedienung über ein Touch-Display eines Smartphones [24] unterstützen. Für das Tracking realer Gegenstände wurde ein Server aufgesetzt, der größere Rechenkapazitäten als die Endgeräte dediziert zur Verfügung stellt. Auf diese Weise können auch aufwendige Methoden verwendet werden. Um auch in einem in Folgeprojekten zunehmend komplexen System diese Flexibilität aufrecht erhalten zu können, wurde bei dem Architekturentwurf das Model-View-Control-Pattern verwendet.

6.2 Ausblick

Zunächst wird sich auf die Tracking Komponente konzentriert werden. Hier können verschiedene Technologien unabhängig voneinander oder in Kombination in den Server integriert und anschließend die Ergebnisse damit evaluiert werden. In diesem Zuge kann auch erst sicher festgestellt werden, ob die hier implementierte Netzwerk-Verbindung mitsamt dem festgelegten Nachrichten-Austausch zu einem Flaschenhals in der Performanz des Gesamtsystems wird oder ob sie den in Kap. 2 formulierten Anforderungen standhalten kann. Im darauf folgenden Schritt wird sich mit der Erweiterung der Unity3D-Anwendung befasst und der virtuelle AR-Inhalt umgesetzt. Dieser wird stark von den Ergebnissen der Tracking Versuche abhängig sein. Eine mögliche zu untersuchende Frage ist die, wie der virtuelle Gegenspieler so repräsentiert werden kann, dass die gesamte Anwendung möglichst immersiv wirkt. Die Arbeit von Günther u. a. [10] wies darauf hin, dass es hier einen Zusammenhang geben könnte.

Literatur

[1] Apple Inc.: *Augmented Reality - More to Explore with ARKit 6.* – URL https://developer.apple.com/augmented-reality/arkit/. – Zugriffsdatum: 2023-02-01

[2] Azuma, Ronald T.: A Survey of Augmented Reality. In: *Presence: Teleoper. Virtual Environ.* 6 (1997), August, Nr. 4, S. 355–385. – URL http://dx.doi.org/10.1162/pres.1997.6.4.355. – ISSN 1054-7460

[3] Bitkom Research, VuMA;: *Anteil der Smartphone-Nutzer* in Deutschland in den Jahren 2012 bis 2021.* – URL https://de.statista.com/statistik/daten/studie/585883/umfrage/anteil-der-smartphone-nutzer-in-deutschland/. – Zugriffsdatum: 2021-11-30

[4] Cherna, Timothy: *ScytheKick: Scythe Companion.* – URL https://play.google.com/store/apps/details?id=com.cherna.scythekick&hl=de_AT&gl=US. – Zugriffsdatum: 2023-01-29

[5] Costa, Eduardo Dias d.: *Unity With MVC: How to Level Up Your Game Development.* – URL https://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development. – Zugriffsdatum: 2023-01-22

[6] Fan, Zhaoxin ; Zhu, Yazhi ; He, Yulin ; Sun, Qi ; Liu, Hongyan ; He, Jun: Deep Learning on Monocular Object Pose Detection and Tracking: A Comprehensive Overview. In: *ACM Comput. Surv.* (2022), mar. – URL https://doi.org/10.1145/3524496. – ISSN 0360-0300

[7] Fowler, Martin: *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional. – ISBN 0321127420

[8] Goodfellow, I. ; Bengio, Y. ; Courville, A. ; Safari, an O'Reilly Media C.: *Deep Learning – Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze.* mitp Verlag, 2018. – ISBN 978-3958457003

[9] Gräfe, Lena: *Umsatz mit Gesellschaftsspielen in Deutschland von 2005 bis 2021.* – URL https://de.statista.com/statistik/daten/studie/796288/umfrage/umsatz-mit-gesellschaftsspielen-in-deutschland/. – Zugriffsdatum: 2022-05-07

[10] Günther, Sebastian ; Müller, Florian ; Schmitz, Martin ; Riemann, Jan ; Dezfuli, Niloofar ; Funk, Markus ; Schön, Dominik ; Mühlhäuser, Max: CheckMate: Exploring a Tangible Augmented Reality Interface for Remote Interaction. New York, NY, USA : Association for Computing Machinery, 2018 (CHI EA '18), S. 1–6. – URL https://doi.org/10.1145/3170427.3188647. – ISBN 9781450356213

[11] Hintze, Nadia: *Entwicklung und Einsatzmöglichkeiten einer AR-Anwendung im Rahmen von Museumsausstellungen.* Dezember 2018. – URL https://users.informatik.haw-hamburg.de/~abo781/abschlussarbeiten/ba_hintze.pdf

[12] Mehler-Bicher, Anett ; Steiger, Lothar: *Augmented Reality.* Berlin, Boston : De Gruyter Oldenbourg, 2022. – URL https://doi.org/10.1515/9783110756500. – ISBN 9783110756500

[13] Microsoft: *Gesten in Unity.* – URL https://learn.microsoft.com/de-de/windows/mixed-reality/develop/unity/gestures-in-unity. – Zugriffsdatum: 2023-02-06

[14] Microsoft: *TcpClient Class.* – URL https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets.tcpclient?view=net-7.0. – Zugriffsdatum: 2023-02-06

[15] Milgram, Paul ; Kishino, Fumio: A Taxonomy of Mixed Reality Visual Displays. In: *IEICE Trans. Information Systems* vol. E77-D, no. 12 (1994), 12, S. 1321–1329

[16] Patel, Ankur A.: *Praxisbuch Unsupervised Learning - Machine-LEarning-Anwendungen für ungelabelte Daten mit Python programmieren.* O'Reilly Media, 2020. – ISBN 978-3-96009-127-1

[17] Pedersen, Stonemaier Games; Morten M.: *The Compelling Power of Solo Play for Tabletop Game Kickstarters.* – URL https://stonemaiergames.com/the-compelling-power-of-solo-play-for-tabletop-game-kickstarters/. – Zugriffsdatum: 2022-07-29

[18] Photon: *PUN.* – URL https://www.photonengine.com/pun#. – Zugriffsdatum: 2023-01-30

[19] PTC Inc.: *Detect and Track Multiple Targets Simultaneously.* – URL https://library.vuforia.com/objects/detect-and-track-multiple-targets-simultaneously. – Zugriffsdatum: 2023-02-01

[20] Python: *Streams*. – URL https://docs.python.org/3/library/asyncio-stream.html. – Zugriffsdatum: 2023-02-01

[21] Rayar, Frédéric ; Boas, David ; Patrizio, Rémi: ART-Chess: A Tangible Augmented Reality Chess on Tabletop. In: *Proceedings of the 2015 International Conference on Interactive Tabletops amp; Surfaces*. New York, NY, USA : Association for Computing Machinery, 2015 (ITS '15), S. 229–233. – URL https://doi.org/10.1145/2817721.2823485. – ISBN 9781450338998

[22] Schmalstieg, Tobias ; Lewin, Laura (Hrsg.): *Augmented Reality - Principles and Practice*. Taub, Mark L., 2016

[23] Stonemaier Games: *Augmented Reality for Scythe*. – URL https://scythekick.medium.com/augmented-reality-for-scythe-41c7a620530c. – Zugriffsdatum: 2023-01-21

[24] Technologies, Unity: *Input.GetTouch*. – URL https://docs.unity3d.com/ScriptReference/Input.GetTouch.html. – Zugriffsdatum: 2023-02-06

[25] Tilt Five Inc.: *t5 - TiltFive*. – URL https://www.tiltfive.com/lp/home. – Zugriffsdatum: 2023-01-21

[26] Unity: *UnityWebRequest*. – URL https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html. – Zugriffsdatum: 2023-01-30

[27] Unity Technologies: *Entity Component System*. – URL https://docs.unity3d.com/Packages/com.unity.entities@0.1/manual/index.html. – Zugriffsdatum: 2023-01-22

[28] Unity Technologies: *Unity*. – URL https://unity.com/. – Zugriffsdatum: 2022-11-30

[29] Yusof, Cik S. ; Low, Tian S. ; Ismail, Ajune W. ; Sunar, Mohd S.: Collaborative Augmented Reality for Chess Game in Handheld Devices. In: *2019 IEEE Conference on Graphics and Media (GAME)*, 2019, S. 32–37