Umsetzung und Bewertung verschiedener Mixed Reality-Compositing-Techniken

Studiengang Medieninformatik

Bachelorarbeit

vorgelegt von

Julius Hilbig
geb. in Lich

durchgeführt bei
qubic, Seligenstadt

Referent der Arbeit:	Prof. Dr. Cornelius Malerczyk
Korreferent der Arbeit:	M.Sc. Hans Christian Arlt
Betreuer bei qubic:	Christof Frank

Friedberg, 2018

Für meine Familie

Danksagung

Zunächst möchte ich Prof. Dr. Cornelius Malerczyk danken, der mir für diese Bachelorarbeit als Betreuer zur Verfügung stand und darüber hinaus auch ein großes Interesse für die 3D-Computergrafik im mir weckte. Dafür möchte ich auch Hans Christian Arlt und dem übrigen Team des GDV-Lab danken.

Außerdem möchte ich Christof Frank danken, der mir ein sehr gutes Praktikum bei *qubic* ermöglichte, viel Vertrauen in mich hatte und mir die technische Ausstattung für diese Arbeit zu Verfügung stellte und stets hilfsbereit war. An dieser Stelle möchte ich auch Lucas Müller danken, der mir während seines Praktikums bei *qubic*, bei den Aufnahmen für diese Arbeit als „Schauspieler“ half.

Auch danken möchte ich meinen Eltern, die mich stets unterstützen und durch Korrekturlesen und Teilnehmen an der Evaluation geholfen haben. Außerdem danke ich allen anderen Freunden und Bekannten, die an der Evaluation teilgenommen haben und als Ablenkung helfen konnten.

Und ein besonderer Dank gilt meiner Schwester Johanna Hilbig, die trotz des schreibens an ihrer Masterarbeit genug Zeit fand, meine Bachelorarbeit Korrektur zu lesen.

Selbstständigkeitserklärung

Ich erkläre, dass ich die eingereichte Bachelorarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Friedberg, März 2018

Julius Hilbig

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In dieser Arbeit werden vier verschiedene Mixed Reality-Compositing-Techniken umgesetzt, um sie miteinander zu vergleichen und zu bewerten. Schließlich sollen dadurch die besten Einsatzbereiche der einzelnen Techniken gefunden werden.

1.1 Motivation

Virtual Reality (auf Deutsch virtuelle Realität, kurz: VR) ermöglicht Benutzern ein immersives Erlebnis in einer virtuellen Umgebung, mit der durch verschiedene technische Geräte interagiert werden kann. Viele Computerspiele nutzen die Vorteile der Virtual Reality schon, um Spielern eine immersive, interaktive Erfahrung zu bieten. VR muss allerdings nicht nur zur Unterhaltung von Privatanwendern dienen. Für Unternehmen kann die virtuelle Realität viele Vorteile bieten[1]. Für das Entwickeln von Produkten kann VR verwendet werden, um das Produkt vor der Produktion virtuell zu betrachten, zu gestalten oder zu testen. Beim Vermarkten von Produkten könnten potenzielle Kunden das Produkt vor einem Kauf mit Hilfe eines VR-Headsets ausprobieren. Studien des Unternehmns *Deloitte* zufolge, ist die Nutzung von VR neben den bereits erwähnten Punkten auch für Schulungen oder virtuelle Meetings sinnvoll und wird dort zum Teil bereits genutzt [DFB16]. So können zum Beispiel Umgebungen für Schulungen in der virtuellen Realität simuliert werden, wenn es nicht möglich ist die Umgebung in der Realität zu besuchen, sie nachzustellen oder um die Kosten für Schulungen zu reduzieren. Durch virtuelle Meetings mit VR-Systemen ist für Teilnehmer nicht zwangsläufig eine Reise nötig und sie ermöglichen das kollaborative Arbeiten an virtuellen Prototypen. Auf diese Art und Weise wird die Technik *Deloitte* zufolge auch schon von Ford eingesetzt, sodass Mitarbeiter aus der ganzen Welt gemeinsam an dem Design neuer Fahrzeuge oder der Entwicklung von autonomen Fahren arbeiten können [DFB16, S. 7].

[1] https://www.lead-digital.de/wofuer-marken-ar-und-vr-nutzen-koennen/ – Zuletzt geprüft am 14.02.18

Abbildung 1.1: Ein Screenshot aus einem von *Valve* produzierten Demonstrationsvideo der *HTC Vive*. Es ist zu sehen, wie die VR-Anwenderin mit einem virtuellen Roboterhund interagiert.
Quelle: https://www.youtube.com/watch?v=qYfNzhLXYGc – Valve 2016

Die Anzahl der Virtual Reality-Nutzern steigt und soll Prognosen der Deutschen Bank zufolge auch weiter steigen [Deu16]. Aus diesem Grund kann es für Unternehmen sinnvoll sein VR zu nutzen, um weitere Kunden auf einem neuen Weg zu erreichen. Aber auch unternehmensintern bietet die virtuelle Realität, wie bereits erwähnt, neue Möglichkeiten für die Entwicklung und Vermarktung von Produkten, Schulungen und für Meetings. Jedoch können sich viele potenzielle private oder kommerzielle Nutzer von Virtual Reality die Vor- oder auch Nachteile von VR nicht ohne das Benutzen eines VR-Headsets vorstellen. Das Ausprobieren eines VR-Headsets ist jedoch nicht überall für jeden möglich. Denn zumindest für die Verwendung eines VR-Systems, das freie Bewegung in einem bestimmten Bereich ermöglicht, sind ein Computer mit hoher Leistung sowie ausreichend Platz nötig. Damit potenziellen Nutzern die Vor- und Nachteile der virtuellen Realität demonstriert werden können, sollten diese also in einem Format präsentiert werden, das möglichst unabhängig von Ort und technischer Ausstattung gezeigt werden kann und auch für Personen nachvollziehbar ist, die noch kein VR-Headset genutzt haben und möglicherweise nicht einmal wissen was VR ist. Es sind also bereits etablierte Formate nötig, wie Text, Bilder oder Video. Von diesen kann aber letztlich nur ein Video die Möglichkeiten von VR aussagekräftig wiedergeben.

Abbildung 1.2: Links ist die *Oculus Rift* zu sehen, mit den zugehörigen Tracking-Kameras rechts und links vom Headset und davor sind die *Oculus Touch Controller* abgebildet. Rechts ist die *HTC Vive* mit den zugehörigen Controller abgebildet.
Quellen: Links: https://goo.gl/HVdthV – Oculus 2017
Rechts: https://images-na.ssl-images-amazon.com/images/I/71oPa3WRXXL._AC_.jpg – HTC 2016

Das Aufnehmen eines Videos, welches die Möglichkeiten von Virtual Reality darstellt, sollte einerseits die Aktionen eines VR-Anwenders in der Realität zeigen und die Folgen dieser Aktionen in der virtuellen Realität darstellen und somit die Interaktionen des Anwenders mit virtuellen Objekten zeigen. Ein Beispiel dazu ist in Abbildung 1.1 zu sehen. Bei der Produktion eines solchen Videos muss eine reale Aufnahme des Anwenders mit einer Aufnahme dessen virtueller Umgebung zusammengesetzt werden, um den VR-Benutzer in der virtuellen Umgebung zu zeigen. Es wird also ein Compositing durchgeführt. Dadurch entsteht eine Vermischung der Virtual Reality und der Realität – die Mixed Reality.

Für ein solches Mixed Reality-Compositing können verschiedenen Techniken verwendet werden. Das Verwenden der Game Controller eines VR-Headsets, wie die der *HTC Vive* [2] oder der *Oculus Rift* [3] (beide sind in Abbildung 1.2 abgebildet), kann für ein Compositing hilfreich sein. Sowohl das VR-Headset als auch die Controller werden in Echtzeit getrackt. Ein Controller könnte verwendet werden, um eine reale Kamera zu tracken und deren Position dann für die Positionierung einer virtuellen Kamera zu verwenden.

[2] https://www.vive.com/
[3] https://www.oculus.com/rift/

1.2 Problemstellung

Für die Produktion eines Mixed Reality-Videos zu einer Virtual Reality-Anwendung, muss die Anwendung die Möglichkeit bieten eine virtuelle Kamera anhand der Position einer realen Kamera zu positionieren. Es gibt verschiedene Compositing-Techniken, um dies zu erreichen. Für Entwickler von VR-Anwendungen oder Spielen, die ihre Anwendung in einem Video präsentieren wollen, stellen sich jedoch folgende Fragen: Welche Mixed Reality-Compositing-Techniken gibt es? Wie müssen die Compositing-Techniken in die Anwendung implementiert werden und wie umfangreich ist diese Implementierung? Welche technische Ausstattung und welche Software ist für die verschiedenen Compositing-Techniken nötig? Wie lange kann die Produktion für ein Mixed Reality-Video abhängig von der Compositing-Technik dauern? Welche Kamerabewegungen sind mit der Compositing-Technik möglich? Gibt es sichtbare Qualitätsunterschiede der verschiedenen Compositing-Techniken, im Bezug auf die Kamerabewegungen?

Diese Fragen sind bisher nur zum Teil beantwortet und eine klare Bewertung der verschiedenen möglichen Compositing-Techniken gibt es noch nicht. Für die Entwicklungsumgebung *Unreal Engine 4* [4], welche in dieser Bachelorarbeit verwendet werden soll, gibt es zu Mixed Reality-Aufnahmen außerdem kaum Dokumentationen oder Plugins zum Thema, was in Kapitel 3 Stand der Technik, Unterkapitel 3.3 Mixed Reality-Compositing weiter behandelt wird.

1.3 Zielsetzung

In dieser Bachelorarbeit sollen zunächst Prototypen für vier verschiedene Mixed Reality-Compositing-Techniken umgesetzt werden, die eine reale Kamera tracken, um eine virtuelle Kamera zu positionieren. Dazu wird eine virtuelle Testumgebung in der *Unreal Engine* erstellt, in der die vier Compositing-Techniken implementiert und getestet werden sollen. Die umzusetzenden Techniken umfassen das Verwenden einer statischen Kamera, das Festlegen der Kamerabewegung vor der Aufnahme, das Rekonstruieren der Kamerabewegung nach der Aufnahme mit Hilfe von Software und die Positionsbestimmung der realen Kamera mit Hilfe eines VR Motion Controllers sowie die gleichzeitige Positionierung der virtuellen Kamera anhand der Controller-Position. Letzteres ist in Abbildung 1.3 zu sehen.

Für jede Technik soll ein kurzes Video erstellt werden, in dem ein VR-Anwender mit seiner virtuellen Umgebung interagiert und die Möglichkeiten der Compositing-Technik demonstriert werden. Die vier Compositing-Techniken sollen im Anschluss ausführlich verglichen und bewertet werden. Kriterien sind dabei Zeitaufwand, Komplexität, technische Beschränkungen und unter Umständen weitere, im Laufe der Bachelorarbeit auftretende Vor- oder Nachteile bei der Umsetzung, Aufnahme, Post-Produktion sowie das finale Video. Das finale Video soll mit Hilfe einer Evaluation darauf bewertet werden, wie ansprechend die Kamerabewegung ist und ob Diskrepanzen in dieser wahrgenommen wurden. Außerdem

[4] https://www.unrealengine.com/

Abbildung 1.3: Eine Kamera wird für eine Mixed Reality-Aufnahme mit einem Controller der *HTC Vive* getrackt. Der Controller ist oberhalb der Kamera montiert.
Quelle: https://www.youtube.com/watch?v=qYfNzhLXYGc – Valve 2016

soll aus den Ergebnissen herausgefiltert werden, für welche Anwendungszwecke sich welche Mixed Reality-Compositing-Technik am besten eignet.

1.4 Aufbau der Arbeit

Diese Arbeit ist in sieben Kapitel unterteilt. Das erste Kapitel „Einleitung“ geht auf die Hintergründe zur Wahl des Themas dieser Arbeit ein, zeigt die zu lösenden Probleme der Thematik auf und gibt einen Ausblick auf die Ziele dieser Arbeit. Das Zweite Kapitel „Grundlagen“ gibt das für den Leser dieser Arbeit nötige Basiswissen in den Bereichen Virtual Reality, Unreal Engine und Compositing wieder. Das dritte Kapitel „Stand der Technik“ zeigt die aktuelle technischen Rahmenbedingungen des Mixed Reality-Compositings und welche Probleme es hat.

Das vierte Kapitel „Vorstellung der Mixed Reality-Compositing-Techniken“ stellt die vier, in dieser Arbeit umzusetzenden, Compositing-Techniken vor. Das fünfte Kapitel „Umsetzung der Mixed Reality-Compositing-Techniken“ zeigt zunächst, wie eine Testumgebung für das Mixed Reality-Compositing in der Unreal Engine erstellt wird und wie der allgemeine Studioaufbau für die Aufnahmen aussieht. Anschließend wird in Unterkapiteln für jede der vier Compositing-Techniken das Vorgehen Implementierung, Aufnahme und Post-Produktion aufgezeigt. Dabei wird zunächst auf die nötige Implementierungsarbeit in der *Unreal Engine* eingegangen. Danach wird gezeigt, wie der Studioaufbau für die verschiedenen Compositing-Techniken angepasst werden muss und darauf folgende der Ablauf des Aufnahmeprozesses der jeweiligen Compositing-Technik präsentiert. Schließlich wird der Post-Produktionsprozess der einzelnen Compositing-Technik dargestellt.

Das sechste Kapitel „Evaluation und Ergebnisse“ vergleicht und bewertet die Compositing Techniken anhand ihrer bei der Umsetzung wahrnehmbaren Eigenarten, messbarer Parameter und einer Evaluation. Im siebten Kapitel „Zusammenfassung und Ausblick“ wird

die Arbeit abschließend noch einmal zusammengefasst und ein Ausblick auf zukünftige Forschungen am Thema dieser Arbeit gegeben.

1.5 Zusammenfassung

Diese Arbeit befasst sich mit vier verschiedenen Compositing-Techniken die für Mixed Reality-Videos verwendet werden können. Bevor die Umsetzung der Compositing-Techniken behandelt wird, werden die Grundlagen der verwendeten Techniken und Programme gezeigt. Dies umfasst Virtual Reality, die *Unreal Engine*, *After Effects* und die *Open Broadcaster Software*. Anschließend wird der aktuelle Stand der Technik von Mixed Reality-Aufnahmen und den dafür nötigen Technologien und Programmen behandelt. Dabei wird unter anderem auf die Tracking-Methoden von Room-Scale VR-Headsets eingegangen und gezeigt, wie und womit aktuell Mixed Reality-Aufnahmen möglich sind.

Nachdem die Grundlagen aufgezeigt wurden, werden die vier Mixed Reality-Compositing-Techniken, die umgesetzt werden sollen, zunächst detailliert erklärt. Die vier Compositing-Techniken sind das Verwenden einer statischen Kamera, das Festlegen der Kamerabewegung vor der Aufnahme, die Rekonstruktion einer aufgenommenen realen Kamerabewegung in der virtuellen Umgebung und das Tracken der Kamera mit Hilfe eines VR-Controllers.
Die vier Compositing-Techniken werden in Form von Prototypen umgesetzt. Für alle vier Techniken ist eine identische Grundimplementierung in der *Unreal Engine* und ein Grundaufbau für die Aufnahme nötig. Nachdem diese gezeigt wurden, wird auf die einzelnen Techniken eingegangen. Beim Erläutern der Umsetzung der vier Techniken werden für jede Technik die jeweils nötigen Arbeiten, die den Prozessen Implementierung, Aufnahme und Post-Produktion zuzuordnen sind, separat gezeigt.

Im Rahmen einer Umfrage werden die vier Techniken anhand der mit ihnen erstellten Videos evaluiert. Die Videos der Techniken „statische Kamera“ und „Kameratracking mit VR-Controller“ wurden in der Evaluation besser bewertet, als die anderen beiden Techniken. Auch bei Betrachtung der bei der Umsetzung deutlich gewordenen Besonderheiten und Probleme sowie des nötigen Zeitaufwands der vier Compositing-Techniken zeigen die beiden Mixed Reality-Compositing-Techniken „statische Kamera“ und „Kameratracking mit VR-Controller“ mehr Vorteile. Beide Techniken benötigen in der Regel weniger Zeit für die Umsetzung und darüber hinaus sind sie weniger fehleranfällig.

Kapitel 2

Grundlagen

2.1 Einleitung

In diesem Kapitel werden die für die folgenden Kapitel nötigen Grundlagen behandelt. Zunächst wird auf die existierenden Virtual Reality-Systeme eingegangen und wie das für die Mixed Reality-Aufnahmen verwendete *HTC Vive* VR-System eingerichtet und verwendet wird. Außerdem wird auf die verwendeten Programme *Unreal Engine*, *After Effects* und *OBS Studio*, sowie deren Funktionen eingegangen.

Diese Programme wurden für diese Arbeit aus verschiedenen Gründen ausgewählt:
Zur Entwicklung von VR-Anwendungen können verschiedene Game-Engines verwendet werden. Weit verbreitet sind dafür *Unity* und *Unreal Engine* aufgrund ihrer kostenlosen Verfügbarkeit und auch da ihr Quellcode offen zugänglich ist (*Unity's* Quellcode ist erst seit dem 26. März 2018 verfügbar[1]). Die Wahl fiel auf die *Unreal Engine*. Grund dafür waren einerseits bereits vorhandene Vorkenntnisse und andererseits soll in dieser Arbeit auch gezeigt werden, dass ohne spezielle Plugins für Mixed Reality-Aufnahmen, wie es sie für *Unity* gibt, möglich ist Mixed Reality-Compositings durchzuführen.
Für das Compositing der Videos gibt es verschiedene Programme die in Frage kommen, wie zum Beispiel *Nuke*[2], *After Effects*[3] oder *Blackmagic Fusion*[4]. Wichtig ist vor allen, dass diese Programme Color-Keying unterstützen. Darüber hinaus sind die genannten Programme speziell für Compositing ausgelegt und bieten dafür weitere Möglichkeiten. Die Wahl fiel hier auf *After Effects*, da es bei *qubic* bereits zur Verfügung stand und einige Vorkenntnisse vorhanden waren. Außerdem steht *After Effects* als Teil von *Adobe's Creative Cloud* auch vielen Entwicklern von VR-Anwendungen bereits zur Verfügung, da sie andere Programme der *Creative Cloud* bereits nutzen.
Für die Live-Produktion mit VR-Anwendungen oder Videospielen kommen die Programme

[1] https://blogs.unity3d.com/2018/03/26/releasing-the-unity-c-source-code/ – Zuletzt geprüft am 27.03.18
[2] https://www.foundry.com/products/nuke
[3] https://www.adobe.com/de/products/aftereffects.html
[4] https://www.blackmagicdesign.com/de/products/fusion/

XSplit Broadcaster[5] und *OBS Studio*[6] in Frage. Beide ermöglichen auch Color-Keying in Echtzeit. Die Wahl fiel hier auf das *OBS Studio*, da es im Gegensatz zu *XSplit* komplett kostenlos ist.

2.2 Virtual Reality

„Virtuelle Realität (Virtual Reality, VR) ist eine computergenerierte Wirklichkeit mit Bild (3D) und in vielen Fällen auch Ton. Sie wird über Großbildleinwände, in speziellen Räumen (Cave Automatic Virtual Environment, kurz CAVE) oder über ein Head-Mounted-Display (Video- bzw. VR-Brille) übertragen.“, so definiert Prof. Dr. Oliver Bendel Virtual Reality in [Ben13]. Heute wird Virtual Reality fast ausschließlich mit Head-Mounted-Displays assoziiert, die es erlauben, in den meisten Umgebungen VR nutzen zu können. Die VR-Headsets zeigen dem Anwender ein in Echtzeit generiertes 3D-Bild, das sich abhängig von den Aktionen und Blickrichtung des Anwenders verändert. Ziel der virtuellen Realität ist es dem Anwender ein möglichst immersives Gefühl zu geben und ihm eine alternative Realität zu simulieren, in der er sich frei umsehen kann.
Für VR werden verschiedene Techniken verwendet, um den Nutzer in die virtuelle Realität einzubinden. Unter anderem werden Sensoren oder Kameras in VR-Headsets verwendet, um deren Bewegungen zu tracken oder externe Sensoren und Kameras tracken die Bewegungen des VR-Anwenders. Bei einigen VR-Systemen ist es dem Anwender möglich mit Hilfe seiner Hände mit der virtuellen Umgebung zu interagieren. Dazu werden in der Regel sogenannte Motion Controller verwendet, deren Bewegungen ebenfalls getrackt werden und durch Tasten an den Controllern können Aktionen ausgeführt werden.

[5] https://www.xsplit.com/de/
[6] https://obsproject.com/de

Abbildung 2.1: Verschiedene VR-Headsets für Smartphones. Abgebildet sind rechts die *Samsung Gear VR*, in der Mitte das *Google Cardboard*, links oben ein *Homido VR* Headset und links unten die *Zeiss VR One*
Quelle: https://www.androidpit.de/htc-setzt-auf-eine-echte-vr-brille – Androidpit 2017

Aktuell gibt es verschiedene Arten von VR-Headsets. Einerseits existieren Vorrichtungen, in die Smartphones eingesetzt werden können, von denen einige in Abbildung 2.1 zu sehen sind. Die Sensoren des Smartphones können dann verwendet werden, um die Bewegungen des Anwenders in der VR-Anwendung zu wiederholen. Diese Systeme ermöglichen es, das VR-Erlebnis immer dabei zu haben und können nahezu überall verwendet werden, da keine externen Geräte notwendig sind. Es gibt einige VR-Headsets, die dieselbe Funktionalität der Smartphone VR-Geräte bieten, aber als geschlossenes System angeboten werden. Ein Beispiel dafür ist das *BOBOVR AIO*[7]. Der Nachteil dieser Virtual Reality-Implementierungen ist jedoch eine Einschränkung des Benutzers in seiner Bewegung, denn er kann sich in virtuellen Umgebung nur frei umsehen, aber nicht frei bewegen, da die räumliche Bewegung nicht komplett getrackt wird.

Eine andere Art von VR-Headsets sind solche, die an einen Computer oder ein vergleichbares Gerät gebunden sind und ein Room-Scale Tracking unterstützen. Dadurch werden die Bewegungen des VR-Anwenders auch in der virtuellen Realität durchgeführt. Somit kann sich der Anwender frei in der virtuellen Realität bewegen, zumindest soweit wie der reale Raum Platz bietet. Außerdem gibt es für diese VR-Systeme Controller, mit denen die Hände des Benutzers getrackt werden können und er mit der virtuellen Umgebung interagieren kann.

[7] http://www.bobovr.com/product/x1/

Abbildung 2.2: Die vier Room-Scale VR-Headsets mit den zugehörigen Controllern. Oben links *PlayStation VR*, oben rechts *Oculus Rift*, unten links *HTC Vive* und unten rechts ein *Windows Mixed Reality*-Headset vom Hersteller *Acer*
Quellen: https://www.vrnerds.de/wp-content/uploads/2016/06/psvr-cam-ps-move.jpg – Sony 2016
https://www.overclockers.co.uk/media/image/thumbnail/VR005OR_169286_800x800.jpg – Oculus 2016
https://images-na.ssl-images-amazon.com/images/I/71oPa3WRXXL._AC_.jpg – HTC 2016
https://goo.gl/bGLeq1 – Acer 2017

Aktuell existieren vier VR-Systeme die das Room-Scale Tracking unterstützen, die in Abbildung 2.2 zu sehen sind:
Sony's 2016 erschienenes *PlayStation VR*[8] ist ein VR-System für die Videospielkonsole *PlayStation 4*. Es verwendet eine stereoskopische Kamera um Controller und Headset anhand von LED-Leuchten, die sich an diesen befinden, zu tracken.
Die bereits seit 2013 als Entwicklerversion erhältlich und 2016 für Endnutzer veröffentlichte *Oculus Rift*[9] wird an einen PC angeschlossen und verwendet zwei bis drei Infrarot-Kameras, die ebenfalls an den PC angeschlossen werden, zum tracken des Headsets und der Controller.

[8] https://www.playstation.com/de-de/explore/playstation-vr/
[9] https://www.oculus.com/rift/

Die ebenfalls 2016 veröffentlichte *HTC Vive*[10] für den PC benötigt zwei Basistationen zum tracken des Headsets und der Controller. Die Basisstationen benötigen aber nur einen Stromanschluss und müssen nicht an den PC angeschlossen werden.
Ende 2017 wurden die *Windows Mixed Reality*-Headsets[11] veröffentlicht, die aktuell von *Lenovo*, *HP*, *Samsung*, *Acer* und *Dell* angeboten werden. Der Unterschied der *Windows Mixed Reality* Headsets im Vergleich zu den anderen drei Systemen ist, dass keine externen Kameras oder Basistationen zum Tracken des Headsets oder der Controller nötig sind, wodurch der Aufbau für dieses VR-System vereinfacht wurde. Stattdessen werden Kameras im Headset zum Tracken von diesem und der Controller verwendet. Dazu befimden sich an den Controllern LED-Leuchten.
In Kapitel 3 Stand der Technik, im Unterkapitel 3.2 Virtual Reality wird noch genauer auf die Tracking-Techniken der drei VR-Systeme für den PC eingegangen.

Die VR-Systeme für den PC benötigen ihre eigene Software, um korrekt eingerichtet und verwendet zu werden. Die *HTC Vive* baut dabei komplett auf *Steam VR* auf. *Steam VR*[12] wurde vom Unternehmen *Valve*[13] entwickelt, das auch die *Vive* zusammen mit *HTC* entwickelte. *Valve* ist ein 1996 gegründetes Softwareentwicklungsunternehmen, das ursprünglich lediglich Videospiele entwickelte, zum Vertrieb und Update dieser Spiele die Vertriebsplattform *Steam*[14] entwickelte und sich seitdem in weitere Bereiche, die häufig etwas mit Videospielen zu tun haben, ausbreitete. *Steam VR* dient als zentrale Software zum Einrichten und Verwenden eines VR-Headset und ermöglicht das Ausführen von VR-Anwendungen, die über *Steam* vertrieben werden. *Steam VR* ist dabei nicht auf eine Marke von VR-Headsets beschränkt, sondern unterstützt neben der *HTC Vive* auch die *Oculus Rift* und in der Beta-Version auch *Windows Mixed Reality*-Headsets (Stand 20.02.2018).

Zur Inbetriebnahme der *Vive* empfiehlt es sich zunächst die Basistationen, die den Namen *Lighthouses* tragen, montiert werden. Diese sollten in gegenüberliegenden Ecken des Raums über Kopfhöhe und leicht nach unten geneigt montiert werden, damit das Tracking gut funktioniert und sie nicht zu leicht blockiert werden. Ist die *Vive* mit einem Computer verbunden, auf dem *Steam VR* installiert ist, kann das System eingerichtet werden. Dazu sind die dargestellten Einrichtungsschritte zu befolgen. Zum Kalibrieren des Bodens, sollen die Controller auf den Boden gelegt werden. Dabei ist zu beachten, dass die Controller gerade liegen und sich während des Kalibrierungsprozesses nicht bewegen. Ansonsten ist die Bodenebene schief oder falsch platziert, was zu Tracking-Ungenauigkeiten führt.

[10] https://www.vive.com/de/
[11] https://www.microsoft.com/de-de/windows/windows-mixed-reality
[12] http://store.steampowered.com/steamvr
[13] http://www.valvesoftware.com/
[14] http://store.steampowered.com

2.3 Unreal Engine

Die *Unreal Engine*[15] ist eine von *Epic Games*[16] entwickelte Spiel-Engine, die in der ersten Version 1998 veröffentlicht wurde. Mittlerweile befindet sich die *Unreal Engine* in der Version 4.18.3 (Stand 21.02.18). Eine Spiel-Engine bietet im Vergleich zu anderen 3D-Anwendungen den Vorteil Bilder von 3D-Szenen mit Beleuchtung, Reflexionen, Physik und dergleichen in Echtzeit zu berechnen. Aktuell haben Spiel-Engine allerdings viele weitere Anwendungsbereiche, als nur Videospiele, denn die optische Qualität von Spiel-Engines entspricht heutzutage in einigen Anwendungsfällen derselben Qualität, wie „Offline Renderings„,auch „Pre-Calculated Renderings„genannt (zu Deutsch „vorberechnete Renderings“), sie erreichen.
Für VR-Anwendungen sind Spiel-Engines unumgänglich, denn die Blickrichtung des Anwenders ändert sich durchgehend und so muss auch das Bild für ihn durchgehend neu berechnet werden.

Die *Unreal Engine* fasst verschiedene Programm-Elemente in sich zusammen, um Entwicklern möglichst viele Möglichkeiten zentralisiert zu bieten. Zu diesen Elementen gehören:
Ein Level-Editor in dem Objekte in einem Level (vergleichbar mit Szenen in anderen 3D-Programmen) platziert und angepasst werden können.
Ein Node-basierter Material-Editor in dem Materialien sehr frei angepasst werden können. Die Unreal Engine ermöglicht auch einige Funktionen in Materialien einzusetzen, mit denen Materialien auch während die Anwendung läuft angepasst und verändert werden können.
Der sogenannte *Level Sequencer*, sowie der vom *Level Sequencer* ersetzte Matinee-Editor, ermöglichen 3D-Kameras im Level zusammenzuschneiden und Level-Objekte zu animieren, um so eine Videosequenz in Echtzeit abspielen zu lassen.
Zum Programmieren von Anwendungen besitzt die *Unreal Engine* eine objektorientierte Node-basierte Skriptsprache, die *Blueprint Visual Scripting*[17] genannt wird.
Ein Beispiel für das *Blueprint Visual Scripting* ist in Abbildung 2.3 zu sehen. Bei dem Beispiel wird das Event „Actor BeginOverlap“ verwendet. Das Event wird ausgelöst, wenn der Actor mit dieser Funktion, von einem anderen Actor im 3D-Raum überlagert wird. Beim Auslösen des Events, wird der andere Actor („Other Actor“) mit dem in der Variable „Player Actor“ gespeicherten Actor verglichen. In einer Branch (eine If-Abfrage im *Blueprint Visual Scripting*) wird das Ergebnis des Vergleichs verwendet und wenn es „True“ ist wird die Variable „Counter“ um eins erhöht.
Das *Blueprint Visual Scripting* ist aber nicht die einzige Möglichkeit in der *Unreal Engine* zu programmieren. Es kann stattdessen auch mit der Programmiersprache C++ programmiert werden. Das Verwenden von C++ bietet auch mehr Möglichkeiten als Blueprints. Denn das *Blueprint Visual Scripting* ist in einigen Bereichen eingeschränkt. Darüber hinaus können Funktionen auch in C++ programmiert und in Blueprints verwendet werden.

[15] https://www.unrealengine.com/en-US/what-is-unreal-engine-4
[16] https://www.epicgames.com/de
[17] https://docs.unrealengine.com/latest/INT/Engine/Blueprints/

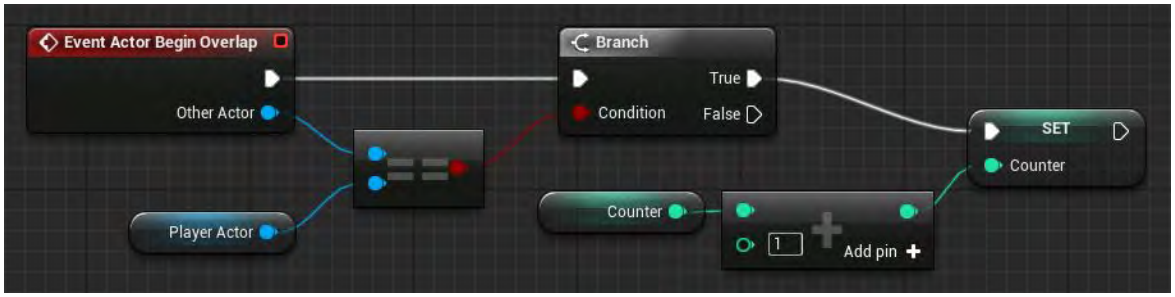

Abbildung 2.3: Abgebildet ist ein Beispiel für das *Blueprint Visual Scripting*, in dem eine Aktion ausgeführt wird, nachdem das Event "Actor BeginOverlap„ ausgelöst wurde.
Quelle: https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/Events/index.html – Epic Games

Nachfolgend werden einige Begriffe erklärt, die speziell für die *Unreal Engine* und das *Blueprint Visual Scripting* sind und die in dieser Arbeit verwendet werden.

Actor Ein Actor ist jedes Objekt, das in einem Level platziert werden kann. Actors unterstützen 3D Transformationen[18]

Component Eine Component (zu deutsch Komponente) ist ein Teil eines Actors, der eine bestimmte Art von Objekt beinhaltet[19]. Das Objekt kann selbst ein Actor sein.

Pawn Ein Pawn ist ein Actor, über den der Anwender die Kontrolle übernehmen und ihn steuern kann.

Character Ist ein Pawn, der in der Lage ist zu gehen, zu laufen, zu springen und weiteres.

PlayerController Ist ein Actor der Eingaben des Anwenders an Pawns weiterleitet, damit der Anwender diese steuern kann.

Game Mode Definiert was für ein Spiel gespielt wird und welche Regeln gelten. Dazu zählt zum Beispiel auch welcher Pawn standardmäßig vom Anwender/Spieler kontrolliert wird.

[18] https://docs.unrealengine.com/latest/INT/Programming/UnrealArchitecture/Actors/index.html,
[19] https://docs.unrealengine.com/latest/INT/Engine/Components/

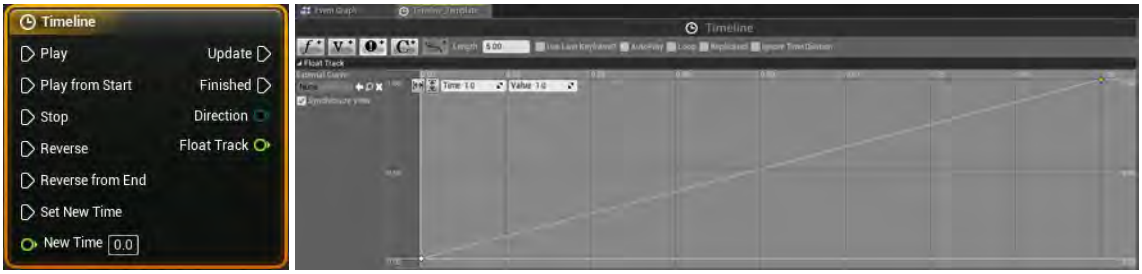

Abbildung 2.4: Links ist die Node einer Timeline im Blueprint-Editor der *Unreal Engine* zu sehen. Rechts ist abgebildet, wie diese Timeline eine Float innerhalb einer Sekunde von null zu eins ändert.

Blueprint Blueprints ermöglichen es Entwicklern Programmlogik existierenden Gameplay Klassen mit Hilfe des *Blueprint Visual Scripting* hinzuzufügen. Gameplay Klassen sind alle Klassen, die bei Laufzeit der Anwendung für den Anwender relevant sind und manipuliert werden können. Beim Erstellen einer Blueprint muss eine Elternklasse ausgewählt werden, auf der die Blueprint basiert. Die Elternklassen sind Actor, Pawn, Character, PlayerController und Game Mode[20].

Timelines Eine Timeline ist eine verfügbare Node im *Blueprint Visual Scripting*. Eine Timeline erlaubt es, über einen festgelegten Zeitraum Zahlen, Vektoren oder Farben zu verändern und auszugeben oder Events zu vorbestimmten Zeitpunkten auszulösen. Dies geschieht durch das Erstellen verschiedener Arten von Tracks in der Timeline und das setzen von Keys in diesen[21]. Ein Beispiel für eine Timeline zeigt Abbildung 2.4.

2.3.1 Unreal Engine für VR-Anwendungen

Die *Unreal Engine 4* ist als Entwicklungsumgebung für VR-Anwendungen aufgrund der kostenlosen Nutzung und umfangreichen Features bereits etabliert. Plugins zur Entwicklung von VR-Anwendungen für *SteamVR*, *Oculus Rift*, *Samsung's Gear VR* und *Google VR* sind standardmäßig in der *Unreal Engine* enthalten.

[20] https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/Types/ClassBlueprint/index.html

[21] https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/Timelines/

Abbildung 2.5: Der Play Button im Unreal Editor startet standardmäßig eine Preview der Anwendung. Diese Preview lässt sich zu einer VR Preview ändern, wie auf dem Screenshot zu sehen ist.
Quelle: https://docs.unrealengine.com/latest/INT/Platforms/VR/CheatSheet/index.html – Epic Games

Um ein Level in VR zu betreten, benötigt es zunächst nicht mehr als einen Pawn mit einer Kamera-Komponente. Wird eine VR Preview gestartet (Siehe Abbildung 2.5), wird die getrackte Position des VR-Headsets automatisch genutzt, um die Kamera-Komponente zu positionieren. In einer fertigen Anwendung (Packaged Build) der *Unreal Engine* kann mit dem Konsolenbefehl „Stereo On“ die Unterstützung für VR-Headsets aktiviert werden. Um dies zu automatisieren, ist es üblich, im „Begin Play“ Event der Level Blueprint diesen Konsolenbefehl mit der Node „Execute Console Command“ auszuführen. Die Level Blueprint ist die Blueprint des Levels, sobald ein Objekt geladen wird, wird das „Begin Play“ Event von diesem ausgeführt. Somit wird beim laden des Levels die Darstellung umgestellt.

Um die Motion Controller der *HTC Vive* oder der *Oculus Rift* zu verwenden, gibt es in der *Unreal Engine* „Motion Controller Components“. Diese können zum Beispiel als Komponenten in einem VR Pawn verwendet werden. Damit ein Controller Mesh in der virtuellen Umgebung angezeigt wird, muss eine Mesh-Komponente an die Motion Controller Komponente geparented werden und ein Mesh für die Komponente ausgewählt werden. Die Motion Controller Komponenten sind selbst nur zur Translations-Wiedergabe der realen Motion Controller gedacht.

Abbildung 2.6: Ein Screenshot aus dem Projekt *Here VR*.

Für die Mixed Reality-Aufnahmen dieser Arbeit dient ein Level aus dem *Here VR* Projekt von *Qubic*. Im *Here VR* Projekt existiert ein Level, von dem aus der VR-Benutzer andere VR-Anwendungen von *Qubic* laden kann. Ein Teil dieses Level ist in Abbildung 2.6 zu sehen. Dieses Level dient als Grundlage für das *Unreal Engine* Projekt dieser Arbeit. Das *Here VR* Projekt selbst baut auf den Blueprints des VR Templates der *Unreal Engine* auf. Zwei Implementierungen des VR Templates und damit auch des *Here VR* Projektes, die für das Verständnis der folgenden Kapitel relevant sind, werden nachfolgend erklärt.
Die Erste ist die Implementierung der Motion Controller im VR Template: Die Motion Controller Komponenten sind nicht Teil des VR Pawns. Stattdessen werden zwei Instanzen eines MotionController Actors mit dem VR Pawn verbunden. Dies geschieht mit einer „Attach to“-Blueprint-Node, wodurch der VR Pawn zum „Parent“ der Controller wird. Die MotionController Actor haben schließlich die Motion Controller Komponenten als eigene Komponente. Dadurch existiert jedoch keine direkte Verbindung zwischen dem VR Pawn und den Motion Controller Komponenten. Von der Programmierer-Perspektive erleichtert diese Implementierung einige Dinge, da die Motion Controller für jede Hand dieselben Funktionen ermöglichen.
Die zweite relevante Implementierung ermöglicht dem Benutzer das Aufheben von Gegenständen. Dies funktioniert folgendermaßen: Der MotionController Actor besitzt eine Sphere Komponente. Wenn diese Sphere Komponente ein Objekt überlagert, das das Interface „Pickup“ implementiert, vibriert der Motion Controller des VR-Systems. Das Vibrieren signalisiert dem Anwender, dass er das Objekt aufheben kann. Der Benutzer muss zum „Festhalten“ des Objektes die Trigger-Taste des Motion Controllers, die sich auf der Rückseite befindet, gedrückt halten. Das zu haltende Objekt wird mit der Node „Attach to Component“ an einer festgelegten Komponente des MotionController Actors befestigt. Zusätzlich wird die Physik-Simulation des aufgehobenen Actors deaktiviert, ansonsten würde er trotzdem fallen. Nach dem Loslassen der Trigger-Taste wird der aufgehoben Actor vom MotionController Actor getrennt und seine Physik-Simulation wieder aktiviert.

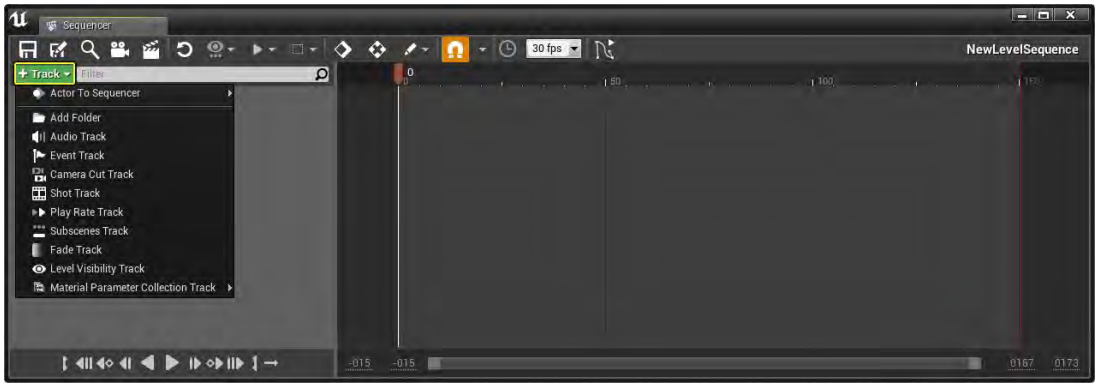

Abbildung 2.7: Der Sequencer der Unreal Engine während ein Track hinzufgefügt wird.
Quelle: https://docs.unrealengine.com/latest/INT/Engine/Sequencer/Overview/ – Epic Games

Unabhängig von der verwendeten Engine gilt es beim Entwickeln von VR-Anwendungen besonders die Performance zu beachten. Eine zu niedrige Bildwiederholrate kann bei VR-Anwendern schnell Übelkeit auslösen und auch, wenn die Bildwiederholrate nicht konstant genug ist, passiert dies. Die VR-Headsets *HTC Vive* und *Oculus Rift* haben eine maximale Bildwiederholrate von 90 Hz, diese sollten VR-Anwendungen versuchen zu erreichen. Die *Unreal Engine* limitiert die Framerate einer VR-Anwendung automatisch auf 90 FPS (Abkürzung für Frames per Second, zu deutsch Bilder pro Sekunde). Wenn die Anwendung nicht konstant 90 FPS erreicht, wird die Framerate automatisch auf 45 FPS beschränkt, damit sie nicht zu stark schwankt. Für ruhigere VR-Erlebnisse können 45 FPS ausreichen, wenn das Geschehen in der virtuellen Umgebung jedoch hektischer ist, sollten 90 FPS erreicht werden, damit dem Anwender nicht übel wird.

2.3.2 Unreal Engine für Videoaufnahmen

Mit Hilfe des Sequencers[22] der *Unreal Engine* lassen sich sogenannte Level Sequences, also Videosequenzen, die Echtzeit abgespielt werden können, direkt im Unreal Editor erstellen. Der Sequencer erlaubt das Mischen von Kamerasequenzen, aber auch das Keyframen vieler Eigenschaften von im Level vorhandenen Actorn, wie zum Beispiel der Translation. Abbildung 2.7 zeigt den Sequencer der *Unreal Engine* und welche Arten an Tracks erstellt werden können. Mit dem Sequencer können in der *Unreal Engine* Videos erstellt werden und der er erlaubt es diese auch direkt als Videodatei oder Bildseqeunz auszuspielen. Dabei handelt es sich um ein Offline Rendering, dessen Qualtiät auch bestimmt werden kann.

[22] https://docs.unrealengine.com/latest/INT/Engine/Sequencer/Overview/

Um das Echtzeitgeschehen der *Unreal Engine* in fertigen Anwendungen oder Previews als Video aufzunehmen, muss jedoch ein zusätzliches Programm verwendet werden. Es kommen verschiedene Programme für Bildschirm- oder Videospielaufnahmen in Frage. Einige Beispiele dafür sind *Fraps*[23], *Dxtory*[24] oder *Bandicam*[25]. Die drei genannten Anwendungen sind die vor allen für die Aufnahme von Videospielen ausgelegt, belasten dafür allerdings den Prozesser des Computers teilweise sehr stark. Ein weiters Beispiel ist *NVidia's Shadowplay*[26], das bei der Verwendung einer unterstützten NVidia Grafikkarte direkt von dieser Videos aufzeichnen kann, ohne die Performance der laufenden Anwendung merkbar zu beeinflussen. Auch möglich ist es, Anwendungen wie *OBS Studio*[27] oder *XSplit Broadcaster*[28] zu verwenden. Beide Anwendungen sind für das Live-Broadcasting ausgelegt, ermöglichen aber auch das Aufzeichnen von Videos.
Zu beachten ist beim Aufnehmen mit solchen Anwendungen die Framerate. Üblich bei der Aufnahme von Videospielen ist es, 30 oder 60 FPS als Framerate zu verwendet. Einige Programme, wie zum Beispiel *Nvidia's Shadowplay* unterstützen keine anderen Framerates.

Ein anderes Feature der *Unreal Engine*, das für Aufnahmen nützlich ist, ist das Replay System[29]. Das Replay System ermöglicht das Aufzeichnen des Geschehens einer Anwendung, um es später wiederzugeben. Dabei werden die Bewegungen von Actors im 3D-Raum, Variablenwerte, das Auslösen von Events und mehr gespeichert. Vor dem Aufzeichnen eines Replays muss jedoch für Actor und deren Komponenten, für die Bewegungen und Aktionen gespeichert werden sollen, Replication (zu deutsch Wiederholung oder Replizierung) aktiviert werden. Die Replication dient eigentlich in Mehrspieler-Spielen dazu, zu entscheiden, welche Actor-Aktionen an den Server gesendet werden und von diesem an die Teilnehmer des Spiels weiterzuleiten, damit die Aktionen dort repliziert werden können. Anstatt des *NetDriver* der diese Daten bei einem Server ausliest, wird der *DemoNetDriver* verwendet, um die zu replizierenden Daten auszulesen und zu speichern. Abbildung 2.8 zeigt die Replication-Optionen eines Actors. Die Option „Replicates" aktiviert die Replictation, „Replicate Movement" aktiviert Replication auch für Bewegungen und mit „Net Update Frequency" lässt sich die Frequenz einstellen, in der die Attribute und Position des Actors an den *NetDriver* oder *DemoNetDriver* gesendet werden.

Die Befehle zum Verwenden des Replay Systems sind nicht direkt im *Blueprint Visual Scripting* enthalten, sondern nur als C++-Funktionen oder Konsolenbefehle. Letztere können mit der Blueprint-Node „Execute Console Command" in Blueprints ausgeführt werden. Folgende Konsolenbefehle sind für das Verwenden des Replay Systems relevant:

[23] http://www.fraps.com/
[24] http://exkode.com/dxtory-features-en.html
[25] https://www.bandicam.com/de/
[26] https://www.nvidia.de/geforce/geforce-experience/shadowplay/
[27] https://obsproject.com/de
[28] https://www.xsplit.com/de/
[29] https://docs.unrealengine.com/latest/INT/Engine/Replay/

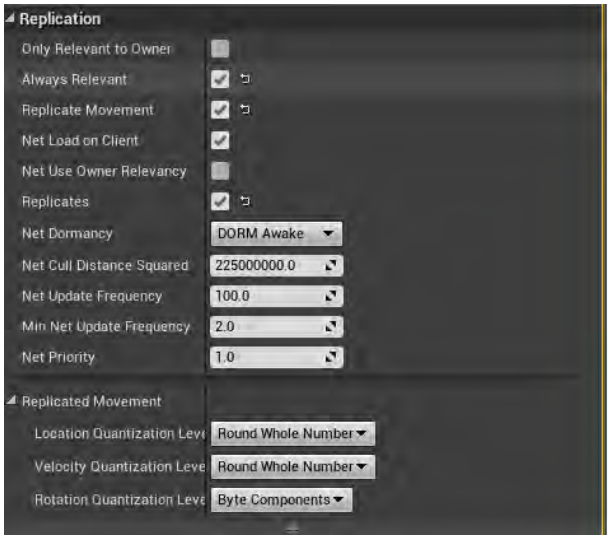

Abbildung 2.8: Die Replication-Einstellungen eines Actors in der *Unreal Engine*.

demorec (ReplayName)	Startet die Aufnahme eines Replays und speichert es unter dem angegebenen Namen. Wenn kein Name angegeben wird, wird er automatisch generiert.
demostop	Stoppt eine laufende Replay-Aufnahme
demoplay (ReplayName)	Gibt das Replay mit dem angegebenen Name wieder.
demoscrub (ReplayZeit)	Springt zu der in Sekunden angegebenen Stelle im Replay, das aktuell wiedergegeben wird.
demopause	Beim ersten Ausführen des Befehls wird die laufende Wiedergabe eines Replays pausiert. Beim zweiten Ausführen wird die Wiedergabe fortgesetzt.
demospeed (ReplayGeschwindigkeit)	Legt die Wiedergabegeschwindigkeit des Replays fest. Ein Wert von 1.0 ist die normale Geschwindigkeit.
demo.RecordHZ (ReplayFrequenz)	Legt die Frequenz fest in der die Daten des Replays gespeichert werden. Der Standardwert ist 8 Hz und reicht nicht aus, um ein flüssiges Replay zu erhalten. Dieser Wert sollte vor dem Aufnehmen eines Replays auf eine höhere Frequenz gesetzt werden.

Abbildung 2.9: Die Einstellungsmöglichkeiten für den *After Effects*-Effekt „Linearer Color-Key“.

2.4 Compositing mit After Effects

Adobe After Effects[30] ist eine Software zur Videoproduktion, die für Videobearbeitung, Compositing, Animation und Effekte ausgelegt ist. Die Software kann als *Adobe Photoshop* für Videos beschrieben werden. Es wird, wie auch in *Photoshop* mit Ebenen gearbeitet, die zum Beispiel Video-Clips, Effektebenen, Masken oder statische Bilder sein können. Ein Vorteil von *After Effects* ist, dass sehr viele Plugins existieren, die das Programm um viele weitere Effekte erweitern können.

Beim Compositing können die verschiedenen, zu kombinierenden Ebenen in *After Effects* auf verschiedene Arten übereinander gelegt und maskiert werden, um ein einheitliches Bild zu erhalten. Ein einfaches Beispiel zum Maskieren, ist das „Zeichenstift-Werkzeug“, mit dem die Maske für eine Ebene frei gezeichnet oder mit Punkten festgelegt werden kann. Es kann dann entschieden werden, ob der „gezeichnete“ Bereich in der Ebene ausgeblendet oder eingeblendet werden soll.
Üblich ist es bei Videoaufnahmen, die später in einem Compositing verwendet werden, einen Green- oder Bluescreen zu benutzen. *After Effects* besitzt zum Ausblenden von Green- oder Bluescreens standardmäßig zwei Effekte und einige weitere zum Verbessern des Ergebnisses. Das Ersetzen eine Farbe durch Transparenz wird als Keying bezeichnet.
Der „Lineare Color-Key“-Effekt ist der einfachere Effekt. Es kann die auszublendende Key-Farbe ausgewählt werden, in welchem Farbraum nach der Farbe gesucht werden soll (RGB-Wert, Farbton oder Chrominanz), mit welcher Toleranz die Farbe ausgeblendet werden soll, also wie ähnlich eine Farbe der Key-Farbe sein muss, um ausgeblendet zu werden, und ob an den Kanten der ausgeblendeten Farbbereiche ein Glättung stattfinden soll. Diese Optionen sind in Abbildung 2.9 zu sehen.

[30] https://www.adobe.com/de/products/aftereffects.html

Abbildung 2.10: Die einfachen Einstellungsmöglichkeiten für den *After Effects*-Effekt „Keylight“.

Der Effekt „Keylight“, aktuell in der Version 1.2, bietet einige weitere Features, im Vergleich zum Linearen Color-Key. Es muss für Keylight auch eine auszublendende Farbe ausgewählt werden. Mit Hilfe der Option „Screen Gain“ wird die Empfindlichkeit beim Ausblenden der Farbe eingestellt. Auch bei diesem Effekt kann eine Glättung für die Ränder des Green- oder Bluescreens eingestellt werden. Darüber hinaus bietet Keylight noch Einstellungsmöglichkeiten um den sogenannten „Spill“ bei Green- oder Bluescreenaufnahmen zu optimieren. Der Spill ist der grüne oder blaue Schimmer der vor allen auf reflektierenden Objekten vor dem Green- oder Bluescreen auftritt. Dieser Spill wird häufig mit ausgeblendet – dies versucht Keylight zu verhindern. Auch lassen sich Masken im Keylight-Effekt festlegen, die bestimmen in welchen Bereichen der Aufnahme der Effekt angewandt werden soll. Außerdem bietet Keylight Farbkorrekturoptionen, um grüne oder blaue Schimmer an Kanten oder im Vordergrund zu minimieren. Die einfachen Einstellungsmöglichkeiten von Keylight sind in Abbildung 2.10 abgebildet.

Ein anderes Feature von *After Effects*, das für Compositing interessant ist, ist die Möglichkeit die Kamerabewegung in einer Aufnahme dreidimensional zu tracken. Der Kameratracker von *After Effects* analysiert dafür zunächst die Aufnahme, um Track-Punkte zu finden. Ein Besipiel dafür ist in Abbildung 2.11 zu sehen. Anschließend berechnet *After Effects* anhand der Track-Punkte eine dreidimensionale Kamerabewegung. Zuvor muss aber eingestellt werden, ob bei der Aufnahme das Sichtfeld der Kamera verändert wurde. Der 3D Kameratracker von *After Effects* bietet allerdings nicht sehr viele Möglichkeiten, um selbst Anpassungen zu machen. So lassen sich die Track-Punkte nicht selbst festlegen. Aber es ist möglich Track-Punkte zu löschen, wodurch die Kamerabewegung neu berechnet

Abbildung 2.11: Abgebildet ist ein Screenshot aus *After Effects*, in dem die generierten Track-Punkte des 3D Kameratrackers als farbige Kreuze zu sehen sind.

wird. Es ist also sinnvoll Track-Punkte zu löschen, die sich auf Objekten befinden, die sich während des Videos bewegen. Ansonsten wird die Kamerabewegung nicht akkurat sein. Bei der Aufnahme vor einem Green- oder Bluescreen, die später getrackt werden soll, entsteht das Problem, dass zunächst keine Track-Punkte existieren, da nur eine grüne oder blaue Fläche als statische Objekte vorhanden sind. Es sollten also Markierungen auf dem Green- oder Bluescreen verwendet werden, damit *After Effects* Track-Punkte finden kann. Die Markierungen sollten dabei nicht nur auf einer Ebene sein, sondern auch im Raum verteilt sein. Nur dann kann die Kamera gut dreidimensional getrackt werden. Trotzdem kann es Aufnahmen geben, die der *After Effects* Kameratracker nicht erfolgreich oder akkurat tracken kann.

2.5 Kameraaufnahme für Compositing

Bei Aufnahmen für Compositing wird in der Regel ein Green- oder Bluescreen verwendet, wenn Reales in eine virtuelle Umgebung integriert werden soll. Dieser muss gut ausgeleuchtet werden, damit er in der Post-Produktion gut ausgekeyt werden kann. Dazu werden ausreichend Scheinwerfer benötigt, die die Ausleuchtung der virtuellen Szene, die für das Compositing verwendet wird, nachahmen sollten. Um die Beleuchtung der realen und virtuellen Umgebung weiter aneinander anzupassen, sollten die verschiedenen Parameter der Kamera entsprechend eingestellt werden. Dabei darf der ISO-Wert nicht zu hoch sein, da die resultierende Körnung im Bild zu Problemen beim Color-Keying führen kann. Bei der Verschlusszeit ist zu beachten, dass lange Öffnungszeiten zu stärkerer Bewegungsunschärfe führen, die beim Color-Keying ebenfalls Probleme hervorrufen kann, da Vorder- und Hin-

tergrund nicht mehr klar getrennt werden können. Empfohlen wird eine Verschlusszeit von 1/125 oder 1/250 Sekunde[31]. Eine niedrige Brennweite, wie 24mm, ist sinnvoll für Green- oder Bluescreenaufnahmen, damit der Vordergrund stets scharf ist und klar vom Hintergrund getrennt werden kann. Bei einer niedrigen Brennweite kann der Blendenwert relativ niedrig eingestellt werden, um eine gute Belichtung zu erreichen. Allerdings führt ein zu niedriger Blendewert zu einem größeren Unschärfebereich. Die Blendenöffnung sollte also trotzdem klein genug sein und der Blendenwert damit groß genug, um starke Unschärfe zu vermeiden.

Wenn die Kameraposition vermessen werden soll, um sie zum Beispiel in einer 3D-Anwendung zu replizieren, sollte der Abstand bis zum Kamerasensor und nicht nur bis zum Objektiv gemessen werden. Denn auf dem Sensor wird das Bild bei einer Kamera abgebildet und in 3D-Anwendungen entsteht das aufzunehmende Bild einer virtuellen Kamera an der exakten Koordinate dieser Kamera.

2.6 Open Broadcaster Software

Das *OBS Studio* (OBS ist die Abkürzung für Open Broadcaster Software)[32] ist eine Open-Source-Software, die für das Live-Broadcasting entwickelt wurde und sich aktuell in der Version 21.0.1 befindet. Die Software ermöglicht die Bildmischung von verschiedenen Videoquellen an einem Computer und das Ausstrahlen eines Live-Broadcasts auf Broadcasting-Webseiten, wie zum Beispiel *Twitch*[33]. Außerdem ermöglicht es das Aufnehmen des Broadcasts, auch wenn dieser nicht ausgestrahlt wird.
Die Bildzusammenstellungen von Quellen können als Szenen gespeichert werden, um sie später zu verwenden. *OBS* unterstützt als Quellen Audioein- oder Audioausgänge, Bilddateien, Videodateien, Fensteraufnahmen, Videospiele, Text oder mit dem Computer verbundene Videoaufnahmegeräte, wie zum Beispiel Webcams. In Abbildung 2.12 ist ein Screenshot des Anwendungsfensters des *OBS Studios* zu sehen.

Auf Quellen können auch einige Filter angewandt werden. So können zum Beispiel Bilddateien als Masken verwendet werden oder ein Chroma oder Color Key Filter auf Videoquellen angewandt werden, um Green- oder Bluescreens auszublenden.

[31] http://thebullseyemedia.com/basic-camera-settings-shooting-green-screen/
[32] https://obsproject.com/de
[33] https://www.twitch.tv/

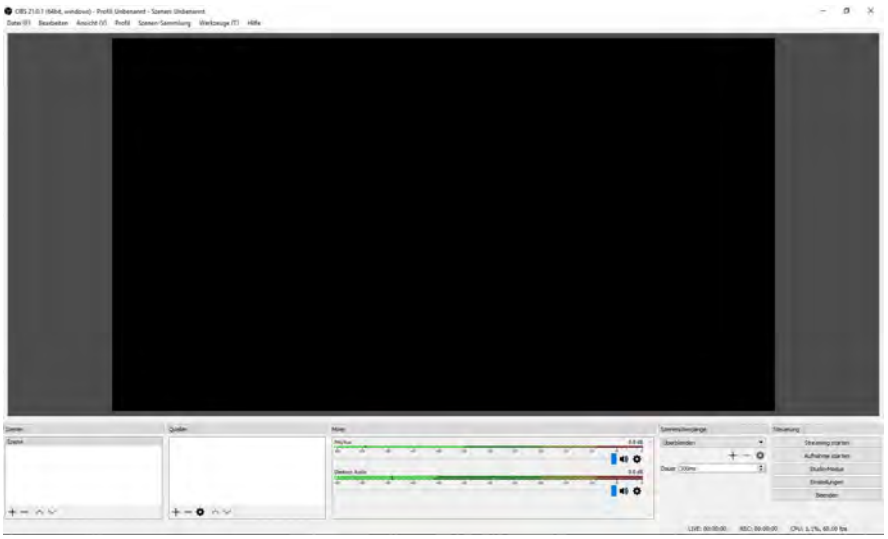

Abbildung 2.12: Ein Screenshot des *OBS Studios*. In der Mite wird das Kontrollbild angezeigt. Unten links können Szenen erstellt und gelöscht werden. Rechts davon werden Quellen ausgewählt und eingestellt.

Kapitel 3

Stand der Technik

3.1 Einleitung

Dieses Kapitel wird den aktuellen Entwicklungs- und Forschungsstand von Virtual Reality und Mixed Reality-Compositing aufzeigen. Dabei wird darauf eingegangen welche technischen Möglichkeiten der existierenden VR-Systeme das Mixed Reality-Compositing beeinflussen, welche Software für Mixed Reality-Aufnahmen existiert, welche Möglichkeiten spezifisch die *Unreal Engine* bietet, die in dieser Arbeit verwendet wird, welche Hardware für Mixed Reality-Aufnahmen vonnöten ist und was bei dieser zu beachten ist.

3.2 Virtual Reality

In diesem Unterkapitel wird der Entwicklungsstand von VR-Systemen behandelt und wie sich deren Entwicklungsstand auf das Mixed Reality-Compositing auswirkt.
Es gibt zwei verschiedene Arten von Virtual Reality-Headsets, die es zu unterscheiden gilt:

- VR-Headsets zur statischen Verwendung
- VR-Headsets mit Room-Scale Tracking

Bei ersterem handelt es sich zum Beispiel um VR-Headsets, in denen ein Smartphone befestigt wird, dessen Sensoren benutzt werden, um die Blickrichtung des Anwenders zu bestimmen. Ein Beispiel für ein solches Headset ist die *Samsung Gear VR-Brille*[1], welche in Abbildung 3.1 zu sehen ist. Diese VR-Brillen bieten dem Anwender aber nicht die Möglichkeit dazu sich in der virtuellen Umgebung zu bewegen, sondern nur die Möglichkeit sich frei umzuschauen. Denn das Headset wird nicht räumlich getrackt, sondern Bewegungen werden des Headsets werden vom Smartphone gemessen und daraus die Blickrichtung des Anwenders bestimmt und in der Anwendung umgesetzt.
VR-Headsets die Room-Scale Tracking unterstüzen, bieten hingegen die Möglichkeit, dass der Benutzer sich in einem begrenzten Raum frei bewegen kann und seine Bewegungen in der Virtual Reality wiedergegeben werden. Um dies zu ermöglichen können unterschiedliche

[1] http://www.samsung.com/de/wearables/gear-vr-r323/ – Zuletzt geprüft am 15.01.2018

Abbildung 3.1: Ein *Samsung* Smartphone wird in die *Gear VR* eingelegt.
Quelle: https://www.youtube.com/watch?v=vfbhL6hKox8 – Samsung 2016

Techniken verwendet werden. Eines der aktuell am meisten verbreiteten Room-Scale VR-Headsets, die *Oculus Rift*[2], benutzt optisches Tracking, um die Positionen von Headset und Eingabegeräten zu bestimmen.
Paul Grimm, Rigo Herold, Johannes Hummel und Wolfgang Broll schreiben in [DBG13, S. 104], dass beim optischen Tracking Objekte mit einer Kamera aufgenommen werden, die im Videostrom leicht zu erkennen sind, um deren Position relativ zur Kamera zu bestimmen. Die Oculus Rift verwendet hierfür zwei bis drei Infrarot-Kameras, die im Raum positioniert werden und Infrarot-LEDs, die im Headset eingebaut sind, aufnehmen. Die Infrarot-Kameras und das VR-Headset mit sichtbaren LEDs sind in Abbildung 3.2 zu sehen. Somit handelt es sich, Paul Grimm und weiteren zufolge, um das *Outside-In-Verfahren*, da die Kameras „von außerhalb des Interaktionsbereichs, die Szene aufnehmen“ [DBG13, S. 107]. Zusätzlich verwendet die Oculus Rift noch Gyroskop- und Beschleunigungssensor sowie einen Kompass, um die Orientierung des Headsets genauer zu bestimmen[3,4].
Die Präzision liegt Michael Abrash (Chief Scientist, Oculus VR) und Dov Katz (Senior Vision Engineer, Oculus VR) zu Folge bei 5 mm [5]. Ein Problem des Tracking Systems der *Oculus Rift* ist jedoch, dass drei Kameras für ein 360° Tracking des VR-Anwenders benötigt werden und jede dieser Kameras mit einem USB-Kabel an den Computer angeschlossen werden muss, wodurch der Aufbau etwas aufwändiger ist, da die Kabel der Kameras nicht im Weg des Anwenders sein sollten. Wenn außer dem VR-Anwender eine weitere Person im Tracking-Bereich der *Oculus Rift* ist, wie zum Beispiel der Kameramann für eine Mixed Reality-Aufnahme, kann eine Kamera schnell von dieser blockiert werden oder ein Kabel im Weg sein.

[2] https://www.oculus.com/rift/ – Zuletzt geprüft am 13.02.1018
[3] https://www.popsci.com/oculus-rift-how-it-works – Zuletzt geprüft am 15.01.2018
[4] https://www.wareable.com/oculus-rift/how-oculus-rift-works – Zuletzt geprüft am 15.01.2018
[5] https://youtu.be/dxbh-TM5yNc?t=42m57s – Zuletzt geprüft am 26.03.18

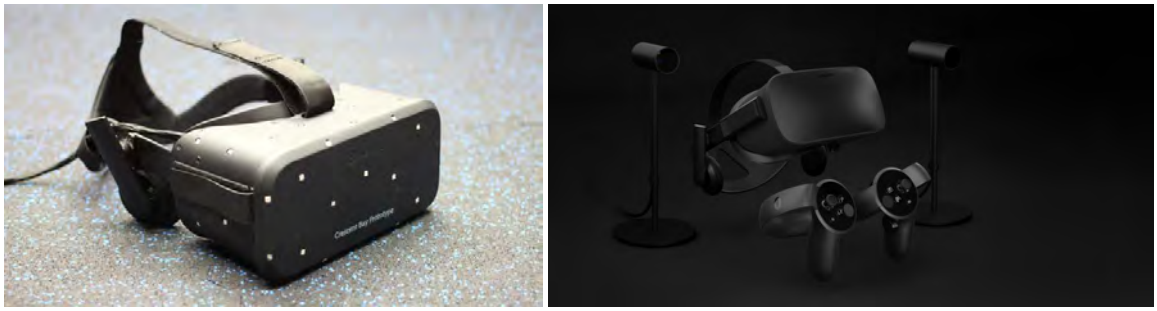

Abbildung 3.2: Links ist ein *Oculus Rift*-Prototyp mit sichtbaren Infrarot-LEDs abgebildet. Rechts ist die *Oculus Rift* mit den Infrarot-Kameras (hinten) und den zugehörigen Controllern (vorne) zu sehen.
Quellen: Links: https://www.flickr.com/photos/pestoverde/16568187562 – Maurizio Pesce 2015
Rechts: https://goo.gl/HVdthV – Oculus 2017

Abbildung 3.3: Abgebildet ist das *HTC Vive* VR-Headset mit den zu gehörigen Motion Controllern.
Quelle: https://images-na.ssl-images-amazon.com/images/I/71oPa3WRXXL._AC_.jpg – HTC 2016

Abbildung 3.4: Links ist ein eingeschaltes *Lighthouse* zu sehen, dessen LEDs und Laser mit Hilfe einer Kamera sichtbar werden. Rechts ist ein geöffneter *Lighthouse*-Prototyp abgebildet. Rechts und Unten im Lighthouse befinden sich die rotierenden Laser und in der Mitte die LEDs.
Quellen: Links: https://roadtovrlive-5ea0.kxcdn.com/wp-content/uploads/2017/04/new-vive-base-station-steamvr-tracking-lighthouse-2.jpg – Craig Albert 2017
Rechts: https://roadtovrlive-5ea0.kxcdn.com/wp-content/uploads/2017/04/lighthouse-base-station-3.jpg – Road To VR 2017

Abbildung 3.5: Zu sehen ist ein Prototyp des *HTC Vive* VR-Headsets bei dem die Sensoren nicht abgedeckt sind.
Quelle: https://i.kinja-img.com/gawker-media/image/upload/s--hQd37FGl--/c_fit,fl_progressive,q_80,w_636/yeou9y7s50wjgjce0cmy.jpg – Carlos Rebato 2015

Die *HTC Vive*[6](Abbildung 3.3) benötigt ebenfalls externe Stationen, die im Raum platziert werden müssen. Allerdings handelt es sich dabei nicht um Kameras. Die sogenannten *Lighthouses* besitzen einige Infrarot-LEDs und zwei rotierende Infrarot-Laser, welche in Abbildung 3.4 zu sehen sind.

[6] https://www.vive.com/de/ – Zuletzt geprüft am 13.02.1018

Die Infrarot-LEDs der *Lighthouses* leuchten den Raum 60 Mal pro Sekunde aus. Nach jedem Aufleuchten der LEDs dreht sich einer der beiden Laser und sendet einen Lichtstrahl horizontal oder vertikal durch den Raum. Nachdem nächsten Aufleuchten der LEDs geht der Strahl des anderen Lasers durch den Raum.
In das *HTC Vive* VR-Headset und in die Motion Controller sind Photodioden eingebaut, die das Infrarot-Licht der *Lighthouses* in elektrische Signale wandeln. Die Positionen der Photodioden am Headset sind in Abbildung 3.5 zu erkennen. Die Photodioden befinden sich unter den Einbuchtungen an Headset und Controllern, die bereits in Abbildung 3.3 zu erkennen waren. Wenn die Photodioden das Infrarot-Blitzlicht der *Lighthouses* erkannt haben, stoppt der Computer die Zeit, bis das Licht des rotierenden Lasers auf die Photodioden fällt. Anhand des Zeitversatzes und der Reihenfolge, in der die Dioden beleuchtet wurden, kann der Computer berechnen, wo sich das VR-Headset oder die Controller relativ zu den *Lighthouses* befinden[7,8].
Diese Tracking-Technik ermöglicht ein sehr genaues Tracking für das VR-Headset und Controller und gegenüber der Oculus Rift ist die Genauigkeit des Trackings nicht aufgrund der Auflösung von Kameras beschränkt. Oliver Kreylos zufolge kann eine Präzision von 2 mm beim Tracking des HTC Vive Headsets und der Controller erwartet werden [Kre16].
Zu dem Ergebnis, dass die Abweichungen von Position stets unter 2 mm und zusätzlich die Abweichung der Ausrichtung stets unter 0.02° liegt, kamen auch Diederick C. Niehorster, Li Li und Markus Lappe in [NLL17]. Allerdings bemerken sie, dass die Referenzebene des Vive VR-Systems nicht immer der Ebene des Bodens entspricht, wodurch das System nicht geeignet für exakte Messungen ist, außer wenn die Ausrichtung der Referenzebene stets gemessen wird.

Die letzten momentan verfügbare Room-Scale VR-System für Computer sind die Windows Mixed Reality-Headsets. Die *Windows Mixed Reality* Headsets[9] benutzen zum Tracking zwei im Headset integrierte Kameras, die die Umgebung zur Positionsbestimmung benutzen. Da die Kameras mit dem zu trackenden Objekt verbunden sind, handelt es sich hierbei nach Paul Grimm und weiteren um ein *Inside-Out-Verfahren* [DBG13, S. 109]. Zusätzlich werden die Kameras verwendet, um die Controller zu tracken. Dies funktioniert letztlich genauso, wie bei der Oculus Rift: An den Controller befinden sich LEDs (Siehe Abbildung 3.6), deren Licht die Kameras aufnehmen. Anhand der Positionierung der LEDs am Controller, kann der Computer basierend auf dem Bild der Kameras, die Position und Ausrichtung der Controller bestimmen. Allerdings können die Controller aufgrund der Kamerapositionen nicht getrackt werden, wenn der Anwender sie hinter dem Rücken oder an einer anderen Stelle außerhalb des Blickfeldes der Kameras hält. Für den Anwender selbst ist dies in vielen Fällen kein Problem. Für Mixed Reality-Aufnahmen ist dies jedoch ein Problem, denn virtuelle Darstellungen der Controller können nur solange die realen Controller getrackt werden, an den

[7] https://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768 – Zuletzt geprüft am 12.02.2018

[8] https://hackaday.com/2016/12/21/alan-yates-why-valves-lighthouse-cant-work/ – Zuletzt geprüft am 12.02.2018

[9] https://www.microsoft.com/de-de/windows/windows-mixed-reality – Zuletzt geprüft am 13.02.1018

Abbildung 3.6: Ein *Windows Mixed Reality*-Headset mit Controllern vom Hersteller *Acer*. An den Controllern lassen sich die zum Tracking benötigten LEDs erkennen.
Quelle: https://pisces.bbystatic.com/image2/BestBuy_US/images/products/6115/6115522_rd.jpg – Acer 2017

richtigen Positionen und mit der korrekten Rotation angezeigt werden. Werden die Controller nicht mehr getrackt sind unnatürliche Bewegungen der virtuellen Controller zu erwarten. Außerdem scheint die Präzision des Trackings der *Windows Mixed Reality*-Headsets nicht so gut zu sein, wie die Präzision von *Oculus Rift* oder *HTC Vive*[10]. Dazu gibt es aber keine offiziellen Angaben oder wissenschaftliche Messungen.

Wenn ein von einem VR-System getracktes Gerät, wie zum Beispiel ein Controller, verwendet werden soll, um die reale Kamera zu tracken, ist es relevant, wie präzise das Tracking des jeweiligen VR-Systems ist. Denn mit Hilfe der Tracking-Daten soll eine virtuelle Kamera genauso wie die reale Kamera bewegt werden. Bei einem nicht präzisen Tracking ist davon auszugehen, dass wahrnehmbare Differenzen entstehen, wenn Aufnahmen der realen und der virtuellen Kamera vermischt werden. Somit ist ein möglichst präzises Tracking in diesem Anwendungsfall nötig.

Die Entwickler des *Oculus Rift* Virtual Reality-Systems haben für Mixed Reality-Aufnahmen Software entwickelt, die es Endnutzern relativ einfach machen soll ihre eigenen Mixed Reality-Aufnahmen zu tätigen. Dazu steht ein ausführlicher Leitfaden zu Verfügung, der Einrichtung und Kalibrierung für Mixed Reality-Aufnahmen beschreibt und auch eine Datei für den 3D-Druck zur Verfügung stellt, mit der sich eine Vorrichtung drucken lässt, um *Oculus Touch Controller* an einer Kamera zu befestigen[11]. Die aufzunehmende Anwendung muss die, für die Mixed Reality-Aufnahmen, nötigen Funktionen allerdings auch unterstützen. *Oculus VR* stellt dafür Plugins für *Unreal Engine* 4[12] und *Unity*[13] zur Verfügung.

[10] https://goo.gl/DDw1rN – Zuletzt geprüft am 12.02.18
[11] https://support.oculus.com/guides/rift/latest/concepts/mr-intro/ – Zuletzt geprüft am 12.02.2018
[12] https://developer.oculus.com/documentation/unreal/latest/concepts/unreal-mrc/ – Zuletzt geprüft am 12.02.2018
[13] https://developer.oculus.com/documentation/unity/latest/concepts/unity-mrc/ – Zuletzt geprüft am 12.02.2018

Für *Windows Mixed Reality* Headsets oder die *HTC Vive* an sich existieren solche Plugins nicht. Allerdings ermöglicht das SteamVR Plugin für Unity Mixed Reality-Aufnahmen[14]. Für die Unreal Engine gibt es kein VR-System unabhängiges Plugin für Mixed Reality-Aufnahmen, somit ist es den VR-Anwendungsentwicklern selbst überlassen, dies zu entwickeln.

3.3 Mixed Reality-Compositing

Mixed Reality, zu deutsch vermischte Realität, ist Paul Milgram und anderen zu Folge der zusammenfassende Begriff für alle Techniken, die Virtualität mit der Realität vermischen und Teil des „Virtualitäts Kontinuums“ sind [MK94]. Der Begriff Mixed Reality wird aber häufig für spezifische Teile des „Virtualitäts Kontinuums“ verwendet, wie die in **3.2 Virtual Reality** erwähnten *Windows Mixed Reality* Headsets zeigen. Mixed Reality-Compositing, oft auch lediglich Mixed Reality Captures (Aufnahmen) genannt, bezeichnet das Aufnehmen eines VR-Benutzers und das Kombinieren dieser Aufnahme mit einer Aufnahme aus der virtuellen Umgebung in der sich der VR-Benutzer befindet.

3.3.1 Aktuell verwendete Mixed Reality-Compositing-Techniken

Für Mixed Reality-Aufnahmen werden aktuell in der Regel zwei Techniken zur Kamerapositionsbestimmung genutzt, damit ein korrektes Compositing der realen und der virtuellen Aufnahmen geschehen kann.
Die erste Technik ist die Verwendung einer statischen Kamera. Zunächst wird die reale Kamera im Aufnahmeraum aufgebaut. Anschließend wird dieselbe Kameraposition für die virtuelle Kamera eingestellt. Damit die virtuelle Kamera korrekt platziert wird, kann die Position der realen Kamera vermessen werden oder ein vom verwendeten VR-System getrackter Controller kann verwendet werden, um die Position zu bestimmen. Ein Greenscreen wird verwendet, um den VR-Benutzer freistellen zu können. Da die Kamera statisch ist, reicht es, wenn der Greenscreen nur auf einer Ebene ist und muss keinen Raum bilden. Ein Video unter Verwendung dieser Technik erstellte zum Beispiel das Animationsstudio *Stormy Studio*[15].
Für die zweite Technik wird ein Controller oder ein anderes vom VR-System getracktes Gerät, wie zum Beispiel der *Vive Tracker*[16], verwendet, um die reale Kamera permanent zu Tracken. Der *Vive Tracker* ist in Abbildung 3.7 zu sehen. Das getrackte Gerät wird an der Kamera befestigt (Siehe Abbildung 3.7), sodass jede Bewegung der Kamera getrackt wird. Die Position des getrackten VR-Gerätes wird in der VR-Anwendung schließlich benutzt, um die virtuelle Kamera zu positionieren. *Oculus VR* erstellte ein auch ein kurzes Video, um das Mixed Reality-Compositing beispielhaft zu demonstrieren[17].

[14] https://medium.com/@dariony/about-mixed-reality-and-a-how-to-part-1-28387e792a4 – Zuletzt geprüft am 12.02.18
[15] https://www.youtube.com/watch?v=maODJkAwqMI – Zuletzt geprüft am 26.03.18
[16] https://www.vive.com/de/vive-tracker/ – Zuletzt geprüft am 13.02.1028
[17] https://www.youtube.com/watch?v=YDzLFAF07-U – Zuletzt geprüft am 26.03.18

Abbildung 3.7: Links ist der *Vive Tracker* zu sehen. Er besitzt, so wie das *Vive* Headset und die Controller, mehrere Photodioden, die zur Positionsbestimmung verwendet werden. Rechts ist der *Vive Tracker* auf einer Kamera befestigt abgebildet.
Quelle: https://goo.gl/jNTgAV – Road To VR 2017

Abbildung 3.8: Ein Screenshot aus *OBS Studio*, der ein Bild aus dem Spiel *Rick and Morty: Virtual Rick-ality* zeigt. Für das Mixed Reality-Compositing zeigt die Anwendung vier verschieden Bilder in vier Quadranten an: Oben links den Vordergrund, oben rechts die Maske für den Vordergrund, unten links den Hintergrund und unten rechts die Perspektive des VR-Anwenders. Abbildung aus [TBM17].

Um die virtuellen und realen Bilder zusammenzusetzen wird häufig für diese beiden Techniken eine Software wie *XSplit Broadcaster*[18] oder *OBS* (Abkürzung für *Open Broadcaster Software*)[19] verwendet. Diese beiden Softwares sind für die Bildmischung von Live-Produktionen ausgelegt. Sie ermöglichen für das Mixed Reality-Compositing reales und virtuelles Bild in Echtzeit zusammenzusetzen.
Einige VR-Anwendungen können für das Compositing Vordergrund und Hintergrund, sowie eine Maske zum Trennen dieser in verschiedenen Quadranten des Anwendungsfensters ausgeben. In Abbildung 3.8 ist diese Darstellung aus dem Videospiel *Rick and Morty: Virtual Rick-ality*[20] zu sehen. Der Hintergrund kann dann hinter der Aufnahme des VR-Benutzers dem Greenscreen angezeigt werden und der Vordergrund kann mit Hilfe der Maske vor dem Benutzer angezeigt werden, sodass der Anwender in die Anwendung integriert wird.
Für die *Unity* Game Engine existieren, wie in **3.2 Virtual Reality** bereits erwähnt, Plugins, damit Entwickler dieses Feature in ihre Anwendung implementieren können. Das *Oculus Rift* Plugin für die *Unreal Engine* ermöglicht dies auch, zumindest für Anwendungen für die *Oculus Rift*. Anwendungen für die *HTC Vive*, für *Windows Mixed Reality* oder für andere zukünftige *SteamVR* Geräte, die mit der *Unreal Engine* entwickelt werden, können diese Features also nicht ohne weiteres ermöglichen. Dafür ist ein Umschreiben des Quellcodes der Engine notwendig, damit die verschiedenen Renderpasses in einem oder mehreren Fenstern gleichzeitig ausgegeben werden können.

3.3.2 Compositing in der Unreal Engine

Seit der Version 4.17 enthält die *Unreal Engine* das Tool *Composure*[21], das Compositing direkt im *Unreal Editor* ermöglicht. *Composure* nutzt den bereits existierenden *Level Sequencer*, mit dem Level Objekte animiert oder Kamerasequenzen geschnitten werden können. Mit diesem Tool ist es somit möglich Compositings in finaler oder nahezu finaler Qualität in Echtzeit anzusehen, während an diese bearbeitet werden. Abbildung 3.9 zeigt ein Beispiel für Compositing mit der *Unreal Engine*, bei dem Auto-Modelle aus der *Unreal Engine* in ein reales Video eingefügt wurden.
Allerdings ermöglicht *Composure* nicht verschiedene Renderpasses in einer fertigen Anwendung gleichzeitig wiederzugeben, wie es für eine Mixed Reality-Aufnahme nötig wäre. Denn das *Composure* Tool ist nicht für Mixed Reality-Aufnahmen gedacht, sondern für die Post-Produktion in Film-Produktionen oder die Produktion von Zwischensequenzen von Videospielen, die in der *Unreal Engine* erstellt wurde, in denen das Compositing dann in Echtzeit abgespielt werden kann. So kann zum Beispiel ein vom Spieler erstellter Charakter in eine reale Umgebung eingefügt werden.
Das in 3.3.1 Aktuell verwendete Mixed Reality-Compositing-Techniken bereits erwähnte

[18] https://www.xsplit.com/de/broadcaster – Zuletzt geprüft am 13.02.18
[19] https://obsproject.com/ – Zuletzt geprüft am 13.02.18
[20] http://www.adultswim.com/games/pc-console/rick-and-morty-simulator-virtual-rick-ality/ – Zuletzt geprüft am 13.02.18
[21] https://www.unrealengine.com/en-US/blog/composure-compositing-tool-sample-released – Zuletzt geprüft am 14.02.18

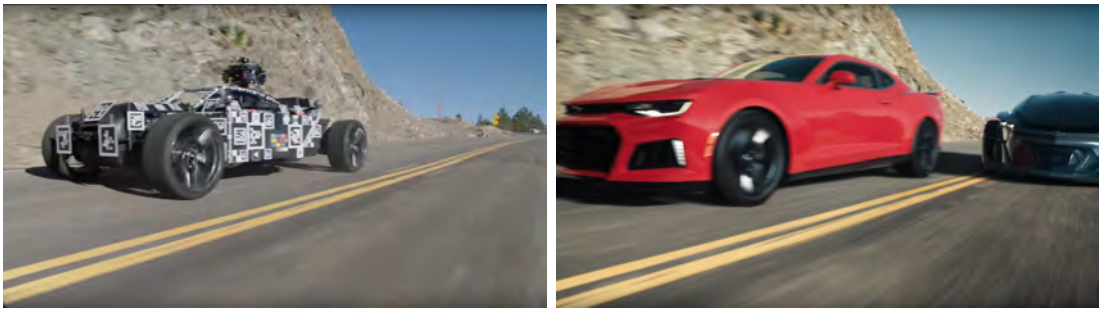

Abbildung 3.9: Zwei Screenshots aus einem Video von *Epic Games*, das als Beispiel für Compositing mit dem *Composure* Tool in der *Unreal Engine* dient.
Quelle: https://www.youtube.com/watch?v=eQjdWKVKRS8 – Epic Games

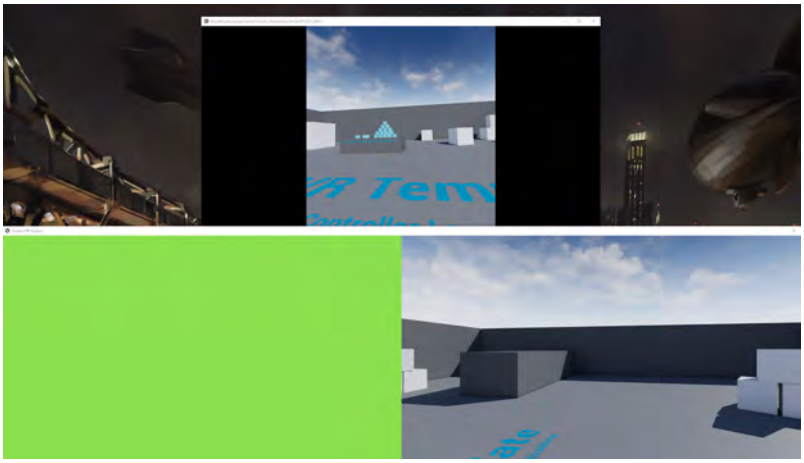

Abbildung 3.10: Ein Screenshot der vom *Oculus OVRPlugin* für das Mixed Reality-Compositing erzeugten Fenster. Das obere Fenster gibt die Perspektive des VR-Anwenders wieder. Im unteren wird rechts der zu Anwender relative Hintergrund und links der Vordergrund vor einer grünen Fläche angezeigt.
Quelle: https://developer.oculus.com/documentation/unreal/latest/concepts/unreal-mrc/ – Oculus VR

Abbildung 3.11: Zwei Screenshots einer Demo von *Epic Games*für den VR Spectator Screen. Links ist die Perspektive des VR-Benutzers zusehen, rechts die externe Kameraperspektive, in der der Charakter des Benutzers zu sehen ist.
Quelle: https://docs.unrealengine.com/latest/INT/Platforms/VR/VRSpectatorScreen/ – Epic Games 2017

Oculus OVRPlugin für die *Unreal Engine*, ist hingegen auch für Mixed Reality-Compositing ausgelegt, aber funktioniert nur bei Verwendung der *Oculus Rift*. Ähnlich wie in Abbildung 3.8 erzeugt dieses Plugin ein zusätzliches Fenster in dem auf einer Seite der zum VR-Anwender relative Hintergrund und auf der anderen Seite die Vordergrund-Objekte vor einer grünen Fläche angezeigt werden, wie in Abbildung 3.10 zu sehen. Diese Darstellung kann in *OBS Studio* oder *XSplit Broadcaster* verwendet werden, um den Vordergrund mit Hilfe eines Chroma Keys freizustellen und vor dem Hintergrund anzuzeigen.

Um das Mixed Reality-Compositing in der *Unreal Engine* mit einer *HTC Vive* oder anderen VR-Headsets zu ermöglichen, für die kein Plugin für Mixed Reality-Aufnahmen vorhanden ist, können andere Features der *Unreal Engine* verwendet werden. Seit Version 4.17 besitzt die *Unreal Engine* einen *Virtual Reality Spectator Screen*[22], also einen Zuschauermodus für VR. Damit kann, anstatt der standardmäßigen Anzeige der Perspektive des VR-Headsets, im Anwendungsfenster eine externe Kameraperspektive angezeigt werden, in der der virtuelle Charakter des VR-Anwenders angezeigt wird (Siehe Abbildung 3.11). Dieses Feature kann verwendet werden, um eine reale Kamera in der *Unreal Engine* nachzustellen und die Aufnahmen von virtueller und realer Kamera können in einem Compositing zusammengesetzt werden.

Ein anderes Beispiel für Compositing unter Verwendung der *Unreal Engine* zeigt Abbildung 3.12, in der Screenshot aus einem Testvideo des Unternehmens *On-set Facilities* zu sehen ist. Zum Rendern beziehungsweise Baking der virtuellen Umgebung wurde die *Unreal Engine* verwendet, für das Compositing von Vorder- und Hintergrund wurde dann *NUKE*[23] verwendet. Asa Bailey, ein Mitarbeiter von *On-set Facilities*, schrieb zu dem Echtzeit-

[22] https://docs.unrealengine.com/latest/INT/Platforms/VR/VRSpectatorScreen/ – Zuletzt geprüft am 16.02.18
[23] https://www.foundry.com/products/nuke

Abbildung 3.12: Ein Screenshot aus einem Testvideo von On-Set Facilities, der einen Echtzeit-Compositing Test zeigt. Auf den Bildschirmen im Vordergrund ist auf dem linken Bildschirm der virtuelle Hintergrund aus der Unreal Engine und auf dem zentralen Bildschirm das Compositing von virtuellen Hintergrund und realem Vordergrund zu sehen.
Quelle: https://www.youtube.com/watch?v=5gZWna6RnCA – On-Set Facilities 2018

Compositing-Test in [Asa18], dass die Kamera Tracking-Informationen der Kamera, in denen auch Fokus und Zoom der Kamera enthalten sind, im FBX-Format gespeichert wurden. Die FBX-Daten konnten dann in *NUKE* verwendet werden, um auf dem Set das Compositing durchzuführen. Die Szene aus der *Unreal Engine* stammende Szene wurde anscheinend vor dem Dreh in einem Baking-Prozess vorberechnet und dann in *NUKE* verwendet, wo die virtuelle Kamerabewegung durchgeführt wurde. Jedoch lässt sich dieser Prozess nicht eindeutig aus dem Artikel erschließen und auch die Methode zum Tracken der Kamera wird nicht weiter beschrieben. Zusammenfassend wurde für dieses Testvideo von *On-set Facilities* die Unreal Engine zwar verwendet, aber nur für das Baking der virtuellen Umgebung. Die Kamerabewegung wurde in *NUKE* rekreiert und mit diesem wurde dann auch das Compositing durchgeführt.

3.3.3 Benötigte Hardware für Mixed Reality-Aufnahmen

Um Mixed Reality-Aufnahmen zu machen, sind zunächst ein für VR geeigneter PC, ein VR-System, eine Kamera sowie ein Green- oder Bluescreen mit ausreichender Beleuchtung nötig. Der Aufbau und die Größe des Greenscreens ist dabei abhängig von der gewünschten Kamerabewegung. Eine grüne Wand kann reichen, wenn die Kamera statisch ist, oder sich nur so bewegt, dass der VR-Benutzer stets vor dem Greenscreen ist. Für mehr Freiheit bei der Kamerabewegung ist ein Raum mit zwei bis vier grünen Wänden und Boden allerdings nötig. Die Art der Kamera ist nicht sehr wichtig, solange sie in der Lage ist ein digitales Video aufzuzeichnen. Eine Webcam kann ausreichend sein und bietet den Vorteil, sich leicht an einem VR-Controller befestigen zu lassen, wenn ein solcher zum Tracken der Kamera

Abbildung 3.13: Die Abbildung zeigt eine normale Webcam, die mit Kabelbindern an einen Vive Controller befestigt wurde
Quelle: http://www.ureality.de/blog/mixed-reality-in-steamvr-htc-vive/ – UReality 2017

verwendet werden soll, was in Abbildung 3.13 zu sehen ist. Es kann aber auch eine DSLR Kamera verwendet werden, an die ein VR-Controller mit einem 3D gedruckten Objekt befestigt wird, um sie zu tracken. Die Datei für ein solches Objekt stellt *Oculus VR* in ihrem Leitfaden zum Mixed Reality Capturing zur Verfügung[24] und ist in Abbildung 3.14 zu sehen. Sonstiges Kameraequipment, wie zum Beispiel ein Stativ oder ein Kamera-Rig, sind abhängig von der gewünschten Kamerabewegung und nicht zwingend notwendig.

[24] https://support.oculus.com/guides/rift/latest/concepts/mr-vrobject/ – Zuletzt geprüft am 16.02.18

Abbildung 3.14: Es ist ein *Oculus Touch Controller* zu sehen, der mit Hilfe einer 3D gedruckten Befestigung an einer DSLR-Kamera befestigt wurde. Die 3D-Datei für die Befestigung stellt *Oculus VR* frei zur Verfügung.
Quelle: http://www.ureality.de/blog/mixed-reality-in-steamvr-htc-vive/ – UReality 2017

Wenn eine Live-Produktion durchgeführt werden soll oder eine Live-Vorschau des Compositings sichtbar sein soll, ist es notwendig, dass das Live-Bild der Kamera auf dem Computer zur Verfügung steht. Eine Webcam ermöglicht das ohne weiteres, mit andere Kameras ist eine Verbindung über über USB, HDMI vergleichbaren Anschlüssen nötig, sofern die Kamera dies unterstützt. Um die Kamera per HDMI am Computer anzuschließen, benötigt dieser eine Capture Card oder ein vergleichbares Gerät mit HDMI Eingang, wie zum Beispiel ein *Elgato Cam Link*[25] oder ein *Blackmagic Intensity*[26]. Beim Mixed Reality-Compositing für Live-Produktionen ist zu beachten, dass ein noch leistungsfähiger Computer nötig ist, als für VR nötig war. *Oculus VR* empfiehlt „ein System mit einem Hochleistungs-Motherboard mit mindestens 16 GB RAM, SSDs für Speicherplatz und einem GTX 1080 oder höher“ [Ocu17].
Wenn eine vorbestimmte Kamerafahrt von realer und virtueller Kamera durchgeführt werden soll, ist für die reale Kamerafahrt ein motorisierter Kameraslider, motorisierter Gimbal oder auch ein kompletter, automatisierbarer Kameraarm notwendig, um identische Kamerabewegungen zu erreichen.

[25] https://www.elgato.com/de/gaming/cam-link – Zuletzt geprüft am 16.02.18
[26] https://www.blackmagicdesign.com/de/products/intensity – Zuletzt Geprüft am 16.02.18

Kapitel 4

Vorstellung der Mixed Reality-Compositing-Techniken

Nachfolgend werden die vier Compositing-Techniken erklärt, die in dieser Arbeit umgesetzt werden. Außerdem wird erwähnt, ob die Möglichkeit einer Live-Produktion zu erwarten ist und welche technische Ausstattung, neben der Grundausstattung bestehend aus einem VR-System mit VR-fähigen Computer, Kamera und Greenscreen mit ausreichender Beleuchtung, nötig sein wird.
Bei den Aufnahmen wird das *HTC Vive* VR-System verwendet. Grund dafür sind die in Kapitel 3 Stand der Technik, im Unterkapitel 3.2 Virtual Reality genannte Tracking-Präzision der *HTC Vive*, sowie der einfachere Aufbau der externen Stationen der *Vive*. Vorteilhaft ist, dass die *Lighthouse*-Stationen der *Vive*, im Gegensatz zu den Kameras der *Oculus Rift*, keine Kabelverbindung zum Computer benötigen. Dies reduziert die Kabel die bei den Mixed Reality-Aufnahme im Weg sein könnten.
Bei den Compositing-Techniken, bei denen eine Live-Produktion möglich ist, wird das Programm *OBS Studio* verwendet, um eine solche zu simulieren.

4.1 Statische Kamera

Die erste Compositing-Technik ist die Verwendung einer statischen Kamera. Die Position der Kamera kann vermessen werden und in der virtuellen Umgebung kann die virtuelle Kamera an derselben Position mit gleicher Ausrichtung platziert werden, wie die reale Kamera. In der Post-Produktion müssen die Aufnahmen dann übereinandergelegt und synchronisiert werden, nachdem der Greenscreen maskiert wurde.
Bei dieser Technik ist zu erwarten, dass eine Live-Produktion möglich ist, da lediglich das Keying des Greenscreen für ein Compositing nötig ist und *OBS Studio* dazu in der Lage ist. Voraussichtlich wird außer der Grundausstattung nur ein Stativ oder etwas vergleichbares benötigt, mit dem die Kamera statisch platziert werden kann.

4.2 Festgelegte Kamerabewegung

Für die zweite Compositing-Technik wird die Kamerabewegung zuvor festgelegt. Die Bewegung wird in der virtuellen Umgebung vorprogrammiert und in der Realität wird idealerweise ein programmierbares Gerät verwendet, das eine Kamerabewegung auf Abruf ausführen kann. Die reale und virtuelle Kamerabewegungen können dann gleichzeitig ausgeführt werden, um die Aktionen des VR-Anwenders in der realen und virtuellen Umgebung aufzuzeichnen.
Es kann erwartet werden, dass eine Live-Produktion mit dieser Compositing-Technik möglich ist. Allerdings müssen alle Kamerabewegungen vor ihrer Ausführung festgelegt werden, wodurch wenig Flexibilität bei der Produktion möglich ist.
Neben der Grundausstattung wird für diese Technik ein Gerät benötigt, mit dem eine Kamera eine zuvor festgelegten Bewegung korrekt ausführen kann. Optimal sind dafür automatisierte Kamerakräne, Kameraslider oder andere Geräte zur automatisierten Kamerabewegung. Aber auch einfache nicht-automatisierte Kameraslider können benutzt werden, wenn die festgelegte Bewegung relativ präzise ausgeführt werden kann.

4.3 Rekonstruieren der Kamerabewegung

Die dritte Compositing-Technik rekonstruiert die reale Kamerabewegung im Nachhinein für die virtuelle Aufnahme. Während der Aufnahme wird eine freie Kamerabewegung durchgeführt. Die Kamerabewegung wird nach der Aufnahme mittels Software, wie zum Beispiel mit dem 3D Camera Tracker von *After Effects*, getrackt. Für das Kameratracking gibt es auch einige andere Optionen, wie zum Beispiel *Mocha Pro*[1] oder auch *Nuke*. *After Effects* stand allerdings bereits bei *qubic* zur Verfügung, während andere Software zusätzlich hätte erworben werden müssen. Außerdem haben viele Entwickler von VR-Anwendungen, die eine Mixed Reality-Video erstellen möchten, bereits Zugriff auf *After Effects*, da sehr viele die *Adobe Creative Cloud* bereits benutzen.

Die aus der jeweiligen Software erhaltenen Trackingdaten werden genutzt, um in der VR-Anwendung die virtuelle Kamerafahrt zu erstellen, welche dann aufgenommen wird und mit der realen Aufnahme kombiniert werden kann. Es stellt sich jedoch die Frage, wie die Aktionen des VR-Anwenders in der virtuellen Umgebung stattfinden sollen, wenn die Aufnahme der Virtualität nicht gleichzeitig mit der Aufnahme der Realität stattfindet. Dafür kann das Replay System der *Unreal Engine*, welches in Kapitel 2 Grundlagen, im Unterkapitel 2.3.2 Unreal Engine für Videoaufnahmen erklärt wurde, oder etwas vergleichbares möglicherweise das Problem lösen.
Da die Aufnahme der virtuellen Umgebung erst stattfinden kann, nachdem die reale Aufnahme stattgefunden hat und die Kamera getrackt wurde, kann mit dieser Compositing-Technik keine Live-Produktion durchgeführt werden.
Für diese Compositing-Technik ist nicht zwingend eine von der Grundausstattung abweichende technische Ausstattung nötig. Allerdings ist es sinnvoll ein Kamera-Rig zu verwenden,

[1] https://borisfx.com/products/mocha/

damit die Aufnahmen ein stabiles Bild haben und nicht verwackelt sind. Denn die Kamera kann bei einer flüssigen Bewegung besser getrackt werden.

4.4 Kameratracking mit einem VR-Controller

Bei der vierten Compositing-Technik geht es darum die reale Kamera mit der Hilfe eine VR-Controllers des verwendeten VR-Systems zu tracken. Die Position des VR-Controllers wird von der VR-Anwendung erfasst und kann verwendet werden, um eine virtuelle Kamera zu positionieren. Da die Position des VR-Controllers in Echtzeit getrackt wird, können freie Kamerabewegungen gemacht werden, die sowohl von der realen als auch der virtuellen Kamera gleichzeitig durchgeführt werden. Beide Kamerabilder werden dann zugleich aufgenommen und in der Post-Produktion zusammengesetzt.
Da die reale Kamera in Echtzeit getrackt und mit den Tracking-Daten die virtuelle Kamera zeitgleich positioniert wird, ist eine Live-Produktion mit dieser Compositing-Technik möglich. Dabei ist aber ein ausreichend großer Greenscreen nötig, damit die Kamera möglichst frei ausgerichtet werden kann, denn ohne eine Post-Produktion können Bereiche außerhalb des Greenscreens nicht maskiert werden.
Neben der Grundausstattung wird für diese Compositing-Technik ein Kamera-Rig benötigt, an dem ein VR-Controller befestigt werden kann. Außerdem wird ein dritter VR-Controller benötigt, wenn der VR-Anwender zwei Controller in der Aufnahme verwenden soll, denn standardmäßig gehören nur zwei Controller zu einem VR-System.

Kapitel 5

Umsetzung der Mixed Reality-Compositing-Techniken

Dieses Kapitel unterteilt sich in fünf Unterkapitel. Im ersten Unterkapitel wird die Entwicklung der Testumgebung behandelt, die für alle Compositing-Techniken benötigt wird. Die folgenden vier Unterkapitel zeigen die Umsetzung der vier Compositing-Techniken mit jeweils einer Compositing-Technik pro Unterkapitel. Diese vier Unterkapitel besitzen je drei weitere Unterkapitel in denen die Implementierung, der Aufnahmeprozess und der Post-Produktionsprozess der jeweiligen Compositing-Technik behandelt werden.
Für die Umsetzung der Mixed Reality-Compositing-Techniken wurde an Hardware die *HTC Vive*, eine *Canon EOS 5D Mark III* sowie ein Greenscreen mit Beleuchtung verwendet. An Software wurde die *Unreal Engine*, das *OBS Studio*, *Adobe After Effects* und Blender benutzt.

5.1 Entwicklung der Testumgebung

Bevor mit der Umsetzung der einzelnen Compositing-Techniken begonnen werden konnte, musste die grundlegende Testumgebung entwickelt werden. Diese Testumgebung basiert auf dem *Here VR* Projekt von *Qubic*, das wiederum die VR-Features des VR Templates der *Unreal Engine* benutzt. Die *Here VR* Anwendung bietet für VR dem Anwender die Möglichkeiten virtuell zu laufen, sich zu teleportieren und einige Objekte aufzuheben.
Für die Mixed-Reality Aufnahmen wurde die Anwendung um den VR Spectator Screen Mode erweitert, der in Kapitel 3 Stand der Technik, im Unterkapitel 3.3.2 Compositing in der Unreal Engine erwähnt wurde. Der Spectator Screen besitzt mehrere Modi, die verschiedene Ansichtsarten bieten, wie zum Beispiel das Anzeigen der VR-Anwenderperspektive, einer UTexture (Unreal Engine Texturenformat) oder das Anzeigen einer Textur und der VR-Anwenderperspektive gleichzeitig. Um auf einer UTexture ein Echtzeitbild anzuzeigen, wird der Scene Capture 2D Actor verwendet, der ähnlich wie ein Kamera-Actor funktioniert, aber sein Bild direkt auf einer Textur ausgeben kann.

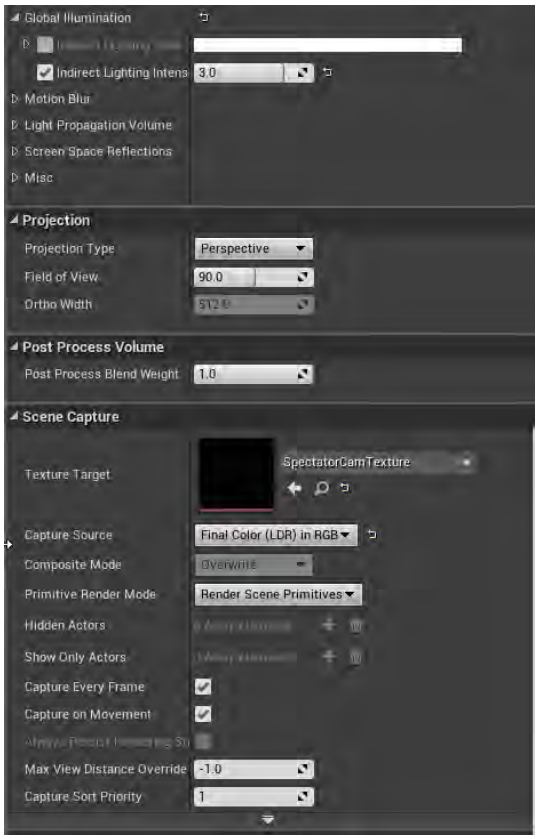

Abbildung 5.1: Screenshot aus dem Unreal Editor mit einigen Einstellungen des Scene Capture 2D Actors. Der gelbe Pfeil markiert dabei die Änderungen an Standardeinstellungen.

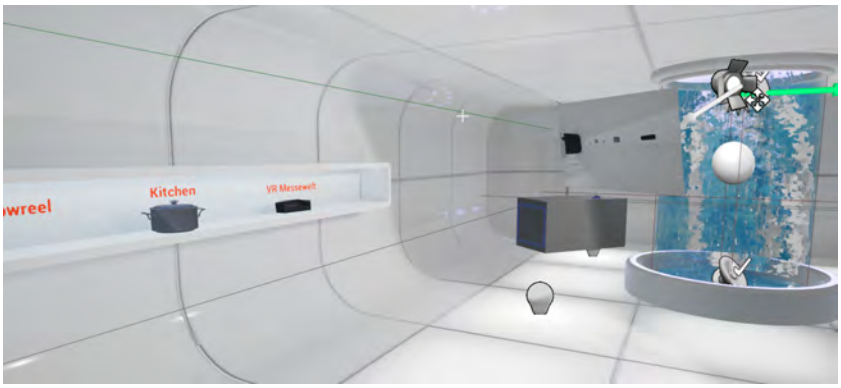

Abbildung 5.2: Dargestellt ist, wie der VR-Anwender die virtuelle Kamera in der *Unreal Engine* wahrnimmt. In dieser Abbildung wird auf einer Fläche oberhalb der Scene Capture Component noch das Bild angezeigt, dass diese aufnimmt.

Zunächst wurde also ein Scene Capture 2D Actor im Level platziert und eine UTexture mit dem Namen „SpectatorCamTexture“ erstellt. Am Scene Capture 2D Actor wurden die in Abbildung 5.1 zu sehenden Einstellungen vorgenommen. Die Zieltextur wurde auf die erstellte UTexture geändert und die Capture Source, die die auf der Textur auszugebenden Kanäle festlegt, wurde auf „Final Color (LDR) in RGB“ geändert. Dadurch sind die Farben akkurat und entsprechen den Farben die der VR-Anwender sieht, ohne ein Post-Processing auszuführen. Außerdem wurde die Global Illumination Einstellung „Indirect Lighting Intensity“ von 1.0 auf 3.0 erhöht, denn mit der Standardeinstellung entsprach die Belichtung des Bildes des Scene Capture 2D Actors nicht der Belichtung des Level Editors oder der virtuellen Kamera für das VR-Headset. Nachdem dies getestet wurde, wurde der Scene Capture 2D Actor durch eine Scene Capture Component ersetzt, die dem VR Pawn hinzugefügt wurde. Dadurch hat die virtuelle Kamera stets dieselbe Position relativ zum Tracking-Ursprung des VR-Systems, der der Koordinate des VR Pawns entspricht. Zusätzlich wurde ein simpler Quader als Kind-Komponenten der Scene Capture Component hinzugefügt. Dieser dient dazu, dem VR-Anwender die Position der realen Kamera mitzuteilen, während er sich in der virtuellen Realität befindet. Dies wird in Abbildung 5.2 gezeigt.

Damit der Spectator Screen im Anwendungsfenster angezeigt wird, wurden die in Abbildung 5.3 abgebildeten Nodes der LevelBlueprint hinzugefügt, die beim Starten der Anwendung ausgeführt werden. Die „Set Spectator Screen Texture“ legt die erstellte UTexture „SpecatorCamTexture“ als Textur für den Spectator Screen fest. In der Node „Set Spectator Screen Mode Texture Plus Eye Layout“ wird festgelegt, wo im Anwendungsfenster sich die Bereiche zur Darstellung der Textur und der VR-Headset-Perspektive befinden. Dafür werden die Eckpunkte mit relativen Koordinaten des Fensters angegeben. Mit der Option „Draw Eye First“ wird festgelegt, ob die VR-Headset-Perspektive zuerst gerendert und damit im Hintergrund angezeigt werden soll. Die Textur wird für die Aufnahmen im ganzen Fenster im Vordergrund angezeigt. Die Node „Set Spectator Screen Mode“ legt den Spectator Screen Modus fest, der für die Mixed Reality-Aufnahmen „Texture“ ist.

Es wurde außerdem in der LevelBlueprint ein Feature programmiert, mit dem sich der Modus des Spectator Screens ändern lässt, um die Perspektive des VR-Anwenders zu überprüfen oder zusammen mit der externen Kameraperspektive anzuzeigen.

Bevor mit den Compositing-Techniken begonnen wurde, wurde getestet, ob ohne ein Aufwändiges umprogrammieren des Renderers der *Unreal Engine*, Vordergrundobjekte für Mixed Reality-Aufnahmen separat gerendert werden können. Ähnlich wie das in Kapitel 3 Stand der Technik, im Unterkapitel 3.3.2 Compositing in der Unreal Engine erwähnte *OVR-Plugin* es ermöglicht.
Dazu wurde getestet, ob das Split Screen Feature der *Unreal Engine* dies ermöglichen könnte. Allerdings funktioniert VR zusammen mit der Split Screen Darstellung nicht, wie Abbildung 5.4 zeigt. Für jeweils ein Auge werden beide Perspektiven angezeigt und zusätzlich eine weitere Perspektive, die vom Split Screen Feature erstellt wurde, aber nur im oberen Drittel angezeigt wird. Normalerweise sollte das Split Screen Feature jeweils die Hälfte des Fenster für je eine Perspektive nutzen.

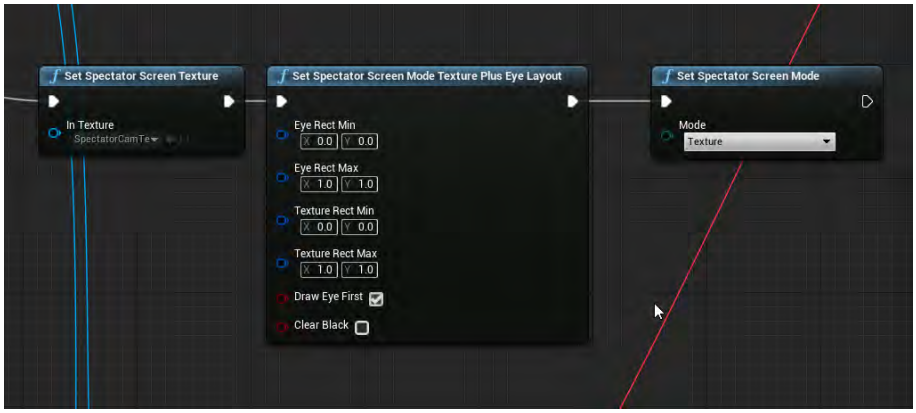

Abbildung 5.3: Nodes zum Anzeigen des VR Spectator Screens, die in der LevelBlueprint ausgeführt werden.

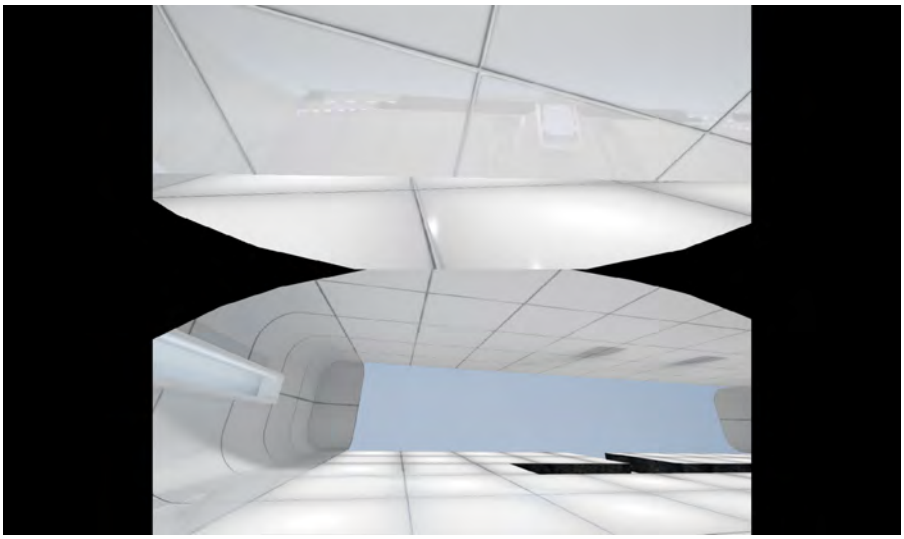

Abbildung 5.4: Abgebildet ist ein Screenshot einer Test-VR-Anwendung in der Split Screen aktiviert wurde. Dieses Bild wird für nur ein Auge angezeigt, enthält aber die Darstellung für beide Augen übereinander.

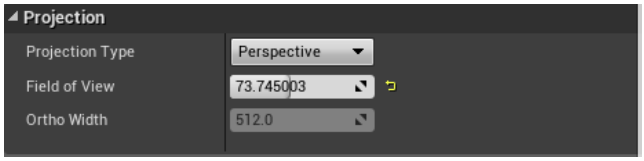

Abbildung 5.5: Die Einstellung des Field of Views der Scene Capture Component ist abgebildet. Diese wurde an das Sichtfeld der realen Kamera angepasst.

Außerdem ermöglicht der Scene Capture 2D Actor im Gegensatz zu einem Camera Actor nicht andere Render Passes darzustellen. Anderenfalls wäre es denkbar ausgewählte Objekte in einem eigenen Render Pass zu rendern. Somit ist dies aber nicht bei der Verwendung des VR Spectator Screens möglich.

Für alle Compositing-Techniken musste das Sichtfeld der virtuellen Kamera, also der Scene Capture Component, an die Brennweite der realen Kamera angepasst werden. Für den Scene Capture 2D Actor kann die Brennweite selbst nicht eingestellt werden, sondern nur das Field of View. Es wurde also zunächst das Field of View einer *Canon EOS 5D Mark III* mit einer Brennweite von 24 mm, die für alle Aufnahmen benutzt wurde, bestimmt. Dafür wurde der *Field Of View Calculator* von Howard Edin[1] benutzt. Das Ergebnis war ein Horizontales Field of View von 73,745°. In der *Unreal Engine* wurde das Sichtfeld der Scene Capture Component entsprechend angepasst, was in Abbildung 5.5 zu sehen ist.

[1] https://www.howardedin.com/articles/fov.html

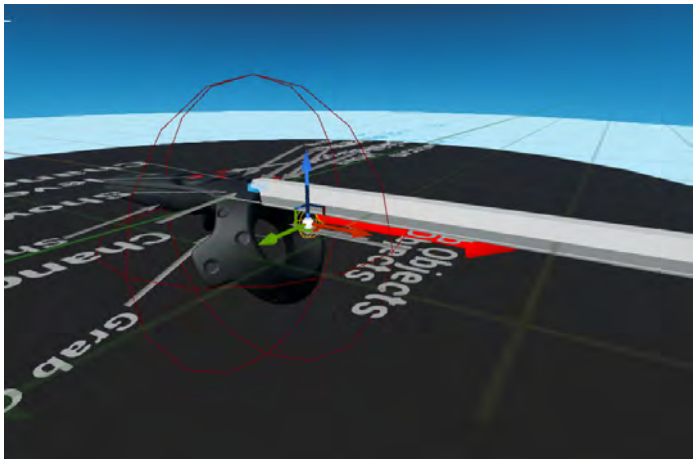

Abbildung 5.6: Zu sehen ist die 3D-Ansicht des Motion Controller Actors. Ausgewählt ist die Sphere Component, die zur Positionsbestimmung der Kamera dienen soll.

5.2 Statische Kamera

5.2.1 Implementierung

Zunächst wurde für die statische Kameraaufnahme die virtuelle Kamera im VR Pawn zumindest ungefähr so platziert, dass die Position ähnlich der realen Kameraposition ist. Für die einfache Bestimmung der Kameraposition wurde einer der Motion Controller verwendet. Dem Motion Controller Actor wurde dafür eine Sphere Component hinzugefügt, die sich vor dem Controller Mesh befindet und damit die Position der realen Kamera bestimmen soll (Siehe Abbildung 5.6). Diese Komponente trägt den Variablen-Namen „Locator“. Im Motion Controller Pawn wurde eine Funktion programmiert, die beim Drücken einer festgelegten Taste am rechten Controller, die Scene Capture Component an der absoluten Position des Locators im Level positioniert. Diese Funktion ist in Abbildung 5.7 zu sehen. Dabei wird die Ausrichtung nicht verändert, da die Kamera senkrecht zum Greenscreen stehen soll.

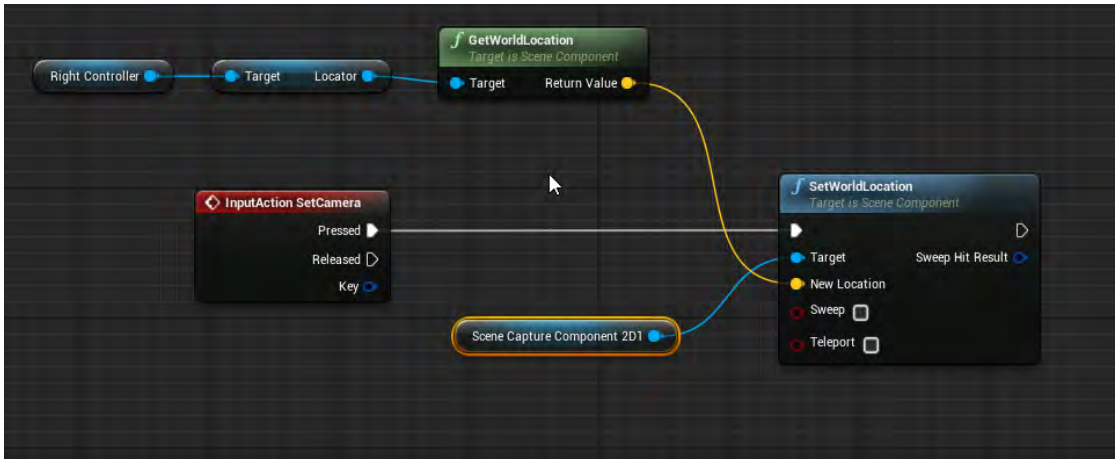

Abbildung 5.7: Abgebildet ist die Blueprint-Funktion, die die Scene Capture Component abhängig von der Position des rechten Motion Controllers platziert. Diese wird ausgeführt wenn die Taste gedrückt wurde, die dem Input-Event „SetCamera“ zugeordnet wurde.

Abbildung 5.8: Zu sehen ist, wie eine Motion Controller der Vive verwendet wird, um die Position der realen Kamera zu bestimmen.

5.2.2 Aufnahme

Zunächst wurde die Kamera auf einem Stativ senkrecht zum Greenscreen aufgebaut. Anschließend wurde die VR-Anwendung beziehungsweise die Preview im *Unreal Editor* gestartet. Dann konnte der rechte Controller verwendet werden, um die virtuelle Kamera zu positionieren, wie in Abbildung 5.8 dargestellt ist.

Abbildung 5.9: Links ist der Unterschied zwischen realer und virtueller Controllerposition zu sehen. Rechts wurde die Kamera neu ausgerichtet, damit die Controller im Bild an derselben Position sind.

Das Programm *OBS Studio* wurde verwendet, um eine Live-Produktion zu simulieren und ein Live-Kontrollbild zu haben. In diesem wurde erkenntlich, dass die Positionen von realen und virtuellen Motion Controllern im Bild nicht übereinstimmten. Dies war aufgrund der bisher nur ungefähren Platzierung der virtuellen Kamera auch zu erwarten gewesen. Um diese Diskrepanz auszugleichen wurde die reale Kamera so ausgerichtet, bis die Positionen übereinstimmten. Siehe dazu Abbildung 5.9 in der ein Motion Controller in einem Stuhl geklemmt wurde, um die Kamera neu auszurichten und den Positionsunterschied auszugleichen.

In *OBS Studio* wurden zwei Quellen eingerichtet. Die erste ist eine Fensteraufnahme der VR-Anwendung, die im Hintergrund angezeigt wird. Die zweite ist eine Fensteraufnahme der *Remote Live View* der Canon-Kamera. Dafür wurde die Kamera über USB mit dem Computer verbunden, um ein Live-Bild der Kamera zu erhalten. Das Fenster der *Remote Live View* ist in Abbildung 5.10 zu sehen. Zunächst wurde probiert mit Hilfe des *Elgato CamLinks* die Kamera über HDMI mit dem Computer zu verbinden. Dies funktionierte allerdings nicht korrekt, da die Kamera nicht komplett vom *CamLink* unterstützt wird.
Die Fensteraufnahme der Live View der realen Kamera benutzt drei Filter zur Anpassung des Bildes: „Zuschneiden/Pad“ schneidet das Bild des Fensters auf den relevanten Teil zu, denn nur das Bild der Kamera wird benötigt. „Chroma Key“ stellt die angegebene Farbe frei, hier also den Greenscreen. „Bild Maske/Blend“ benutzt eine angelegte Bilddatei als Maske, um die nicht vom Greenscreen abgedeckten Bereiche freizustellen. Das Fenster zur Filtereinstellung von OBS ist in Abbildung 5.11 zu sehen.
Aufgenommen wurden letztlich die VR-Anwendung mit Hilfe von *NVidia Shadowplay*, das Kamerabild auf der Kamera selbst und zusätzlich das von *OBS Studio* erstellte Bild, welches in der Post-Produktion als Referenz genutzt werden konnte.

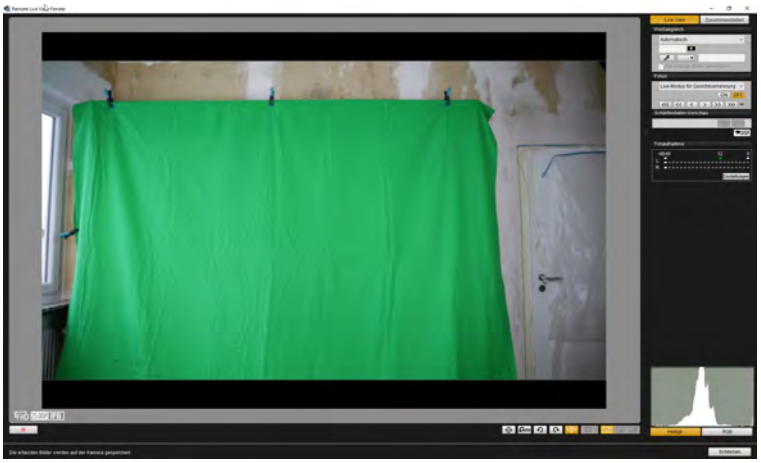

Abbildung 5.10: Abgebildet ist ein Screenshot der *Canon Remote Live View* Anwendung.

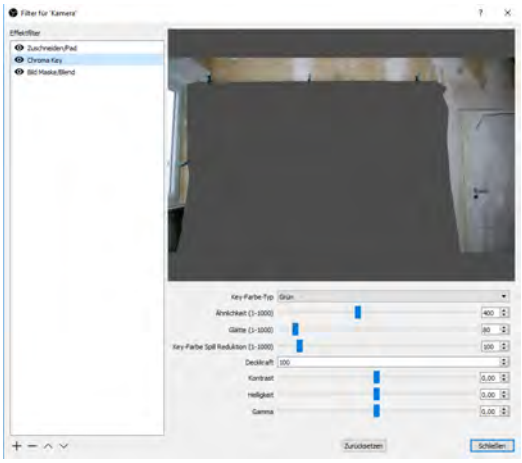

Abbildung 5.11: Abgebildet ist das Filtereinstellungsfenster von OBS mit den Filtern der Fensteraufnahme des *Canon Remote Live View* Fensters. Der „Chroma Key“ Filter ist gerade ausgewählt und die zugehörigen Einstellungen sind unten zu sehen.

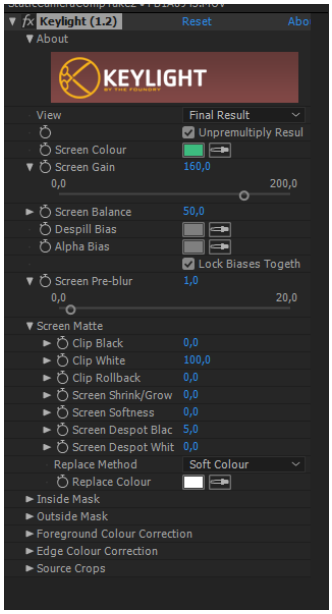

Abbildung 5.12: Ein Screenshot aus *After Effects* in dem die Einstellungen des Keylight-Effektes, die für das Compositing der statischen Kamera verwendet wurden, abgebildet sind.

5.2.3 Post-Produktion

Das Zusammensetzen der Aufnahmen für die statische Kamera war nicht sonderlich aufwendig. In After Effects wurden die beiden Videodateien importiert und in einer neuen Komposition platziert. Die bereits für OBS verwendete Maske wurde ebenfalls importiert und in der Komposition verwendet. Für die Komposition ist wichtig, dass sie die niedrigste Framerate der vorhandenen Videodateien benutzt. Dabei handelte es sich um die Aufnahme der Canon EOS 5D Mark III, die mit 25 Bildern pro Sekunde aufnimmt. In *After Effects* ist die Aufnahme der *Unreal Engine*-Anwendung die unterste Ebene. Die Ebene darüber ist die Real-Aufnahme und darüber ist die Maske angeordnet. Die Maske für die Real-Aufnahme wurde als Luma Matte in der Spalte Track Matte (abgekürzt: TrkMat) des Videos der realen Kamera gesetzt. Um den Greenscreen freizustellen, wird der Effekt Keylight auf die Ebene der Real-Aufnahme angewandt. In den Effekteinstellungen musste die „Screen Colour“ auf die Farbe des Greenscreens geändert werden und der „Screen Gain“ soweit hochgestellt werden, bis der Greenscreen nicht mehr zu sehen war. Die verwendeten Einstellungen sind in Abbildung 5.12 zu sehen.
Außerdem war es nötig die Ebene der virtuellen Aufnahme vertikal geringfügig zu verschieben, damit virtuelle und reale Controller im Bild an derselben Position zu sehen sind. Dies lag daran, dass die Aufnahme der VR-Anwendung eine Auflösung von 1920x1200 Pixel aufgrund der Bildschirmgröße hatte, während alle anderen Aufnahmen eine Auflösung von 1920x1080 Pixel haben.

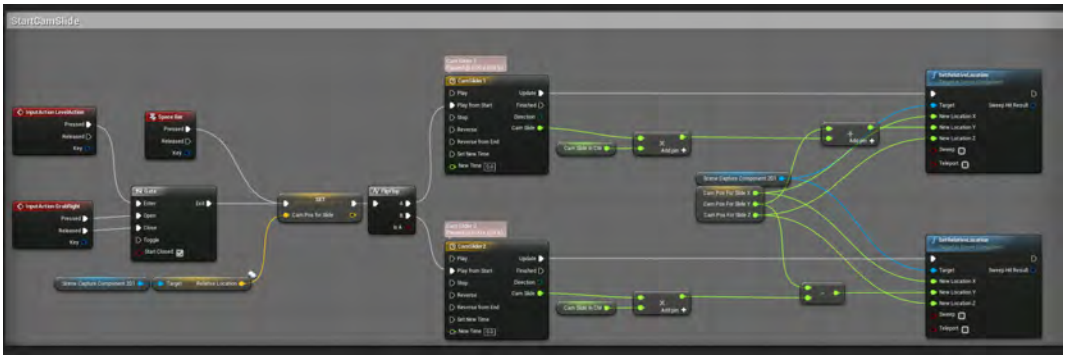

Abbildung 5.13: Screenshot aus der Blueprint des VR Pawns für die Aufnahme der Kamerafahrt. Abgebildet ist die Funktion zum Starten der Kamerafahrt.

5.3 Festgelegte Kamerabewegung

5.3.1 Implementierung

Für die virtuelle Kamerafahrt wurde eine Funktion in der Blueprint des VR Pawns programmiert, die nach dem Drücken einer Taste an der Tastatur oder an einem der Motion Controller, die virtuelle Kamera in einer geradlinigen Kamerafahrt bewegt.
In Abbildung 5.13 ist diese Funktion zu sehen. Ausgeführt wird sie entweder nach dem Drücken der Leertaste oder dem gleichzeitigen Drücken von zwei Tasten am rechten Motion Controller. Es wurden zwei Timelines benutzt. Die erste Timeline („CamSlider1“) interpoliert über 10 Sekunden die Variable „CamSlide“ von 0 auf 1. „CamSlider2“ ändert „CamSlide“ von 1 auf 0. Diese Variable wird mit der Variable „Cam Slide in CM“ multipliziert, die die Länge des Kamerasliders angibt. Das Ergebnis gibt die zurückgelegte Strecke der Kamera an und wird zur Y-Koordinate der Ausgangsposition der Kamera hinzuaddiert. Abschließend wird die Y-Koordinate der Kamera neu gesetzt. Anstatt eine Timeline vorwärts und rückwärts ablaufen zu lassen, wurden zwei Timelines verwendet. Anderenfalls könnte die Timeline, während sie läuft, in ihrer Richtung geändert werden, wodurch sich die Start- und Endpositionen der Kamerafahrt verändern würden. Wird die Taste zum Auslösen dieser Funktion erneut betätigt, läuft die Kamerafahrt rückwärts ab.
Die übrige Implementierung für die Kamerafahrt entspricht der Implementierung der statischen Kamera.

Abbildung 5.14: Der verwendete Kameraslider auf einem Stativ montiert mit befestigter Kamera, bevor mit der Aufnahme begonnen wurde.

5.3.2 Aufnahme

Für die Kamerafahrt wurde der in Abbildung 5.14 zu sehende Kameraslider auf einem Stativ verwendet und von Hand bewegt.

Vor der Aufnahme mussten, ebenso wie für die statische Kamera, die Kamerapositionen und -parameter abgestimmt werden. Die Kameraparameter waren dieselben, wie für die Aufnahme mit der statischen Kamera.

Zum Abstimmen der Kamerapositionen, wurde dieselbe Vorgehensweise, wie für die statische Kamera angewandt, allerdings war zu beachten, dass virtuelle und reale Kamera sowohl am Anfang als auch am Ende der Kamerafahrt die gleiche Position haben. Dazu musste die Länge der Kamerafahrt in der Unreal Engine auf dieselbe Länge, wie die real mögliche Länge der Kamerafahrt, gestellt werden. Wichtig ist hierbei, dass die Fahrtstrecke des Sensors der Kamera relevant ist und nicht die Länge des Kamerasliders. Die Kamerafahrtlänge betrug 106 cm und wurde in der entsprechenden Variable in der *Unreal Engine* eingetragen.

Bei der Aufnahme mussten virtuelle und reale Kamerafahrt möglichst synchron ablaufen, da aber kein automatisierter Kameraslider zur Verfügung stand, musste die Kamera per Hand bewegt werden. Mit Hilfe der Vorschau von OBS konnte die Kamerafahrt aber trotzdem relativ synchron durchgeführt werden, da der Kameramann direkt sehen konnte, ob er zu schnell oder zu langsam war und dies ausgleichen konnte.

Abbildung 5.15: Abgebildet ist die in *After Effects* erstellte und animierte Maske zum Entfernen der Umgebung des Greenscreens.

5.3.3 Post-Produktion

Der Post-Produktionsprozess läuft bei der Kamerafahrt ähnlich ab, wie bei der statischen Kamera. Nachdem der Greenscreen herausgekeyed war, musste allerdings noch zusätzlich eine Maske animiert werden. Denn die Kamera bewegt sich und somit ist unterschiedlich viel Freiraum neben dem Greenscreen zu sehen. Dieser wird mit der Maske entfernt, sodass auch an diesen Stellen des Bildes die virtuelle Umgebung zu sehen ist. Siehe dazu Abbildung 5.15. Es wurde also jeweils zu Beginn und Ende einer Kamerafahrt ein Keyframe für die Maske gesetzt und wenn nötig weitere Keyframes dazwischen, um auftauchende Ränder des Greenscreens zu entfernen.

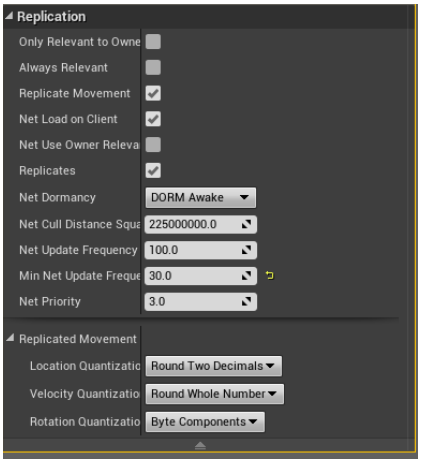

Abbildung 5.16: Die verwendeten Replication Einstellungen des VR Pawns.

5.4 Rekonstruieren der Kamerabewegung

Bei dieser Compositing-Technik fand die Aufnahme der realen und der virtuellen Footage versetzt statt. Zunächst wurde mit der realen Kamera der VR-Benutzer aufgenommen, wie er mit der virtuellen Umgebung interagiert. Danach wurde die Kamera mit Hilfe von After Effects getrackt. Die Kamerabewegung wurde schließlich in die Unreal Engine importiert und erst dann konnte die Footage der virtuellen Umgebung aufgenommen werden. Damit aber die Aktionen des VR-Benutzers bei der realen Kameraaufnahme in der virtuellen Umgebung mit aufgenommen werden konnten, wurde das Replay-System der Unreal Engine verwendet.

5.4.1 Implementierung

Das Replay-System der Unreal Engine speichert nur die Daten von Actorn, die ihre Aktionen replizieren. Ein replizierender Actor sendet die zu replizierenden Daten in einem Netzwerk-Spiel an den Server, damit dieser die Aktionen an andere Clients weiterleiten kann. Damit ein Replay mit den Aktionen des VR-Benutzers aufgezeichnet werden kann, muss der vom Benutzer kontrollierte Pawn, sowie die Actor mit denen er interagiert (zum Beispiel aufhebbare Objekte) replizieren. Dazu muss lediglich die Einstellungsoption „Replicates“ für die Actor und Komponenten der Actor aktiviert werden, die für das Replay relevant sind. Außerdem wurde bei den Actorn, für die „Replication“ eingeschaltet wurde, die „Min Net Update Frequency“ hochgestellt, damit die Objekte im Replay nie mit einer zu niedrigen Frequenz abgespielt werden. Die verwendeten Einstellungen des VR Pawns sind in Abbildung 5.16 zu sehen.

Außerdem ist es wichtig, dass die allgemeine Aufnahmefrequenz für Replays höher gestellt wird. Standardmäßig befindet sich diese nur bei 8 Hz. Für die Aufnahme wurde sie mit Konsolenbefehl „demo.RecordHz 60“ auf 60 Hz gestellt. Dies kann entweder in einer laufenden Anwendung der Unreal Engine in die geöffnete Konsole eingegeben werden oder auch mit der

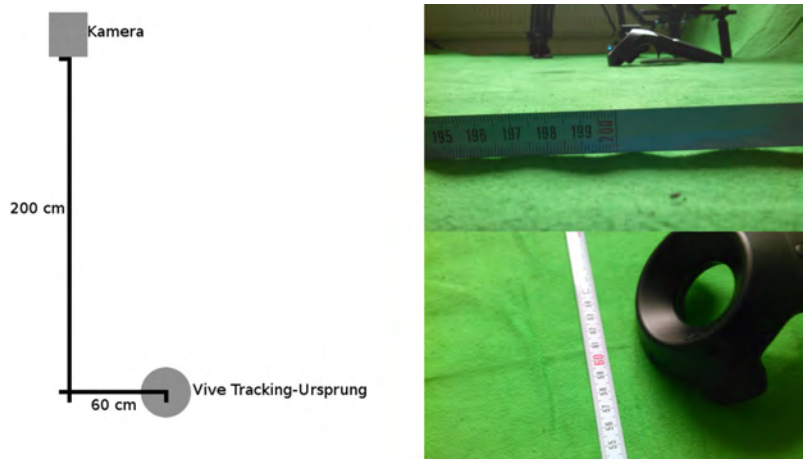

Abbildung 5.17: Abgebildet ist die Vermessung der Kameraposition sowie eine Skizze die diese darstellt.

Node „execute console command“ beim Starten der Anwendung ausgeführt werden. Siehe auch Kapitel 2 Grundlagen, Unterkapitel 2.3.2 Unreal Engine für Videoaufnahmen.
Allerdings war das Verwenden des Replay-Systems mit dem zuerst verwendeten Pawn nicht möglich, denn bei diesem waren die Motion Controller als separate Actor mit der „AttachTo“-Node in diesen integriert worden. Beim Wiedergeben eines Replays führte dies dazu, dass die Controller an einer anderen Stelle waren als der Pawn. Dies betraf auch Objekte, die mit den Controllern aufgehoben wurden. Es wurde also ein neuer VR Pawn erstellt, in dem die MotionController Component direkt im VR Pawn für die beiden Controller implementiert wurde. Dadurch wurden die Controller ohne Probleme in den Replays mit aufgenommen.

5.4.2 Aufnahme

Vor der Aufnahme musste die Ausgangs-Kameraposition relativ zum Mittelpunkt des von der Vive getrackten Raums vermessen werden. Dies wird in der Post-Produktion für die akkurate Rekonstruktion der Kamerabewegung benötigt. Gemessen wurde relativ zum Vive Tracking-Ursprung, wie in Abbildung 5.17 zu sehen ist.
Bei der Aufnahme für diese Compositing-Technik tritt eine Schwierigkeit für den Kameramann auf, da er die virtuelle Umgebung nicht sehen kann, aber über die Kamerabewegung in der virtuellen Umgebung entscheidet. Die Kamera wurde also, sofern möglich, auf einen der Vive Controller gerichtet, welcher auch in der virtuellen Umgebung zu sehen ist.
Um eine möglichst flüssige Kamerabewegung durchzuführen, wurde das in Abbildung 5.18 zu sehende Kamera-Rig verwendet.
Da die Kamera in der Post-Produktion mit Hilfe von After Effects getrackt werden soll, wurden auf dem Greenscreen Markierungen in Form von schwarzem Klebeband platziert und ein Stativ in das Bild gestellt, damit im dreidimensionalen Raum ausreichend Fixpunkte für

Abbildung 5.18: Das Kamera-Rig, das bei den Aufnahmen zur Rekonstruktion der Kamerabewegung verwendet wurde.

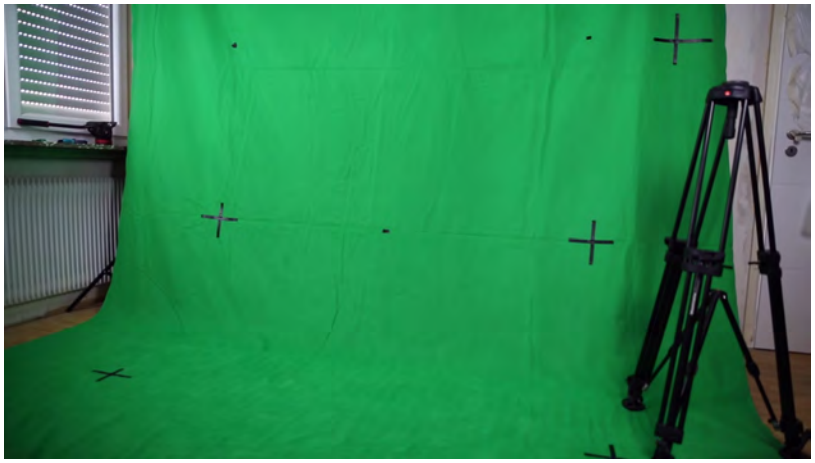

Abbildung 5.19: Abgebildet sind die Markierungen auf dem Greenscreen, sowie das Stativ das ebenfalls als Fixpunkt beim Tracking dienen sollte.

das Kameratracking vorhanden sind. Dies ist in Abbildung 5.19 zu sehen. Für die ersten Aufnahmen wurde noch kein Stativ als Fixpunkt verwendet, und der Boden war nicht zu sehen, dies führte dazu, dass die Kamera nicht erfolgreich getrackt werden konnte. Deswegen wurde für die letzten Takes eine Kamerabewegung gewählt, bei der der Boden zu sehen ist und das Stativ so aufgestellt, dass mehr Fixpunkte für After Effects zu finden waren. Bevor die Kameraaufnahme gestartet wurde, wurde in der Unreal Engine Anwendung die Replay-Aufnahme mit dem Konsolenbefehl „demorec REPLAYNAME“ gestartet und anschließend mit „demostop“ beendet.

Abbildung 5.20: Zu sehen sind von After Effects Kameratracker gefundene Tracking-Punkte, die sich zum Teil auf dem VR-Anwender befinden. Außerdem ist die Maske zu sehen, die die Bereiche neben dem Greenscreen, die Markierungen und das Stativ ausblendet.

5.4.3 Post-Produktion

Zunächst musste die Kamera in einer Aufnahme getrackt werden. Dazu wurde die Aufnahme in *After Effects* importiert und der 3D Camera Tracker gestartet. Nachdem dieser automatisch Fixpunkte zum Tracken gefunden hat, wurden einige davon entfernt, wenn sie sich an Stellen befanden, die sich bewegen. So wurden zum Beispiel einige Fixpunkte auf dem VR-Anwender entfernt. Ein Nachteil von *After Effects'* Kameratracker ist, dass sich nicht manuelle Fixpunkte setzen lassen, sondern nur automatisch gefundene Fixpunkte entfernen lassen. Dadurch wird das Verbessern des Trackings umständlicher. Es waren mehrere Aufnahmen nötig, bis einige erfolgreich getrackt werden konnten. Bei den Aufnahmen, die nicht getrackt werden konnten, gab es folgende Probleme: Auf dem Greenscreen und den daran befestigten Markierungen wurden vom After Effects 3D Camera Tracker in einigen Aufnahmen nur wenige Trackingpunkte gefunden. Da sich die meisten Trackingpunkte dann auf dem Schauspieler befanden, wurde die Kamera nicht sehr gut getrackt. Ein Beispiel für solche Tracking-Punkte ist in Abbildung 5.20 zu sehen. Für das finale Tracking der Kamera wurden vor allen die Tracking-Punkte auf den Armen und ähnlichen Stellen, die sich viel bewegen, entfernt.

Bei einigen Aufnahmen war der Boden nicht zu sehen, dadurch konnte die Kamera nicht vollständig dreidimensional getrackt werden, da Trackingpunkte in nur einer Ebenen vorhanden waren. Um neben dem Boden weitere dreidimensionale Trackingpunkte zu erhalten, wurde ein Stativ in das Bild gestellt. Es ist außerdem zu beachten, dass für Markierungen ein möglichst nicht reflektierendes Material verwendet werden sollte, denn auch dies könnte von After Effects nicht korrekt erkannt werden.

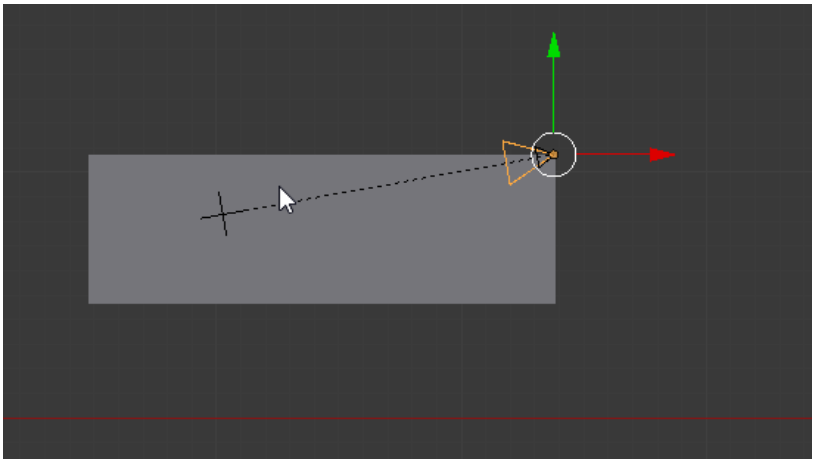

Abbildung 5.21: Die Plane zum Positionieren der Kamera in Blender.

Nachdem die Kamera dreidimensional in After Effects getrackt worden war, wurde sie mit einem Script in ein 3D Programm exportiert. Dazu wurde Blender verwendet und das „After Effects to Blender"-Skript von *Atom*[2]. Das Skript erstellt eine Python-Datei, die die Transformationen der Kamera beinhaltet. In Blender wird die erstellte Datei als Skript eingelesen, wodurch eine animierte Kamera erstellt wird. Die Kamera muss in Blender so positioniert werden, dass sie sich in Unreal Engine richtig bewegt und an derselben Position relativ zum VR Pawn ist, wie die reale Kamera relativ zum Tracking-Urpsrung positioniert war. Die vom Skript erstellte Kamera wurde zur Sicherheit dupliziert und danach an ein „Empty"(in anderen 3D-Anwendungen auch Locator genannt) geparented. Dieses Empty wurde benutzt, um die Kamera zu transformieren, ohne die Animation zu verändern.

Zum Positionieren der Kamera in der *Unreal Engine* wurde eine Plane in Blender erstellt. Diese wurde dann so skaliert, dass sie den durchgeführten Messungen entspricht (Siehe Abbildung 5.21). Ein Eckpunkt der Plane stellt die Startposition der Kamera dar. Der gegenüberliegende Eckpunkt ist an der Position des Tracking-Ursprungs. Die Plane wurde zunächst in die *Unreal Engine* importiert, um die Skalierung zu überprüfen und herauszufinden, wie die Kamera gedreht werden muss, um dem Koordinatensystem der *Unreal Engine* zu entsprechen. Außerdem wird die nötige Transformation in der Unreal Engine bestimmt, indem die Plane, die den Abstand zum Vive Tracking-Ursprung darstellt, in der Unreal Engine an die entsprechende Stelle vor dem VR-Pawn platziert wurde. Die dafür nötige Transformation wird dann benutzt, um die Kamera in Blender korrekt zu positionieren (Siehe Abbildung 5.22). Dabei ist zu beachten, dass das Vorzeichen der Y-Koordinate für Blender geändert werden muss, und dass die Standardeinheit der Unreal Engine Zentimetern entspricht und Blenders Metern.

[2] https://goo.gl/cRDZXT

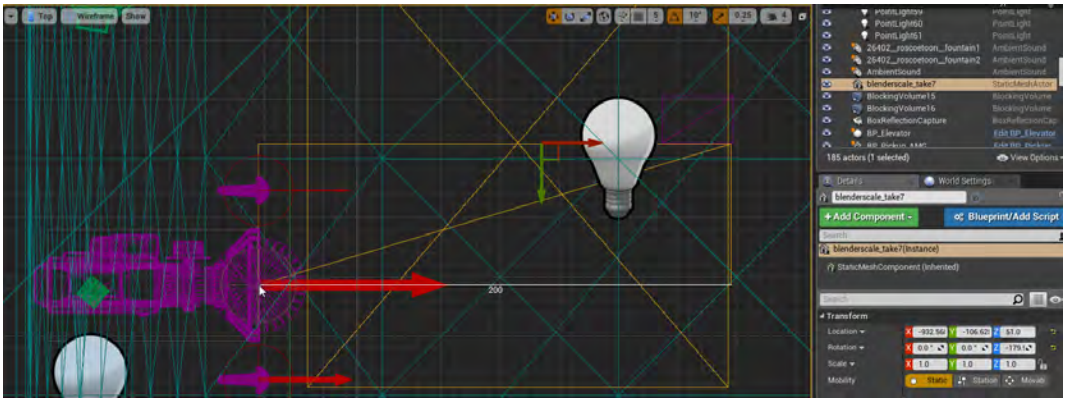

Abbildung 5.22: Verwendung der in Blender erstellten Plane zum Positionieren der Kamera in der Unreal Engine.

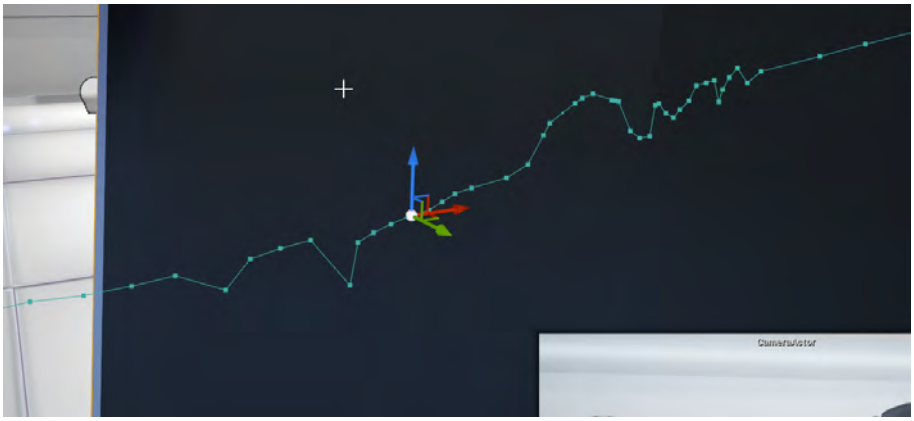

Abbildung 5.23: 3D-Darstellung der Keyframes der Kamerabewegung in der Unreal Engine. In dieser sind einige Sprünge zu sehen.

Zum Exportieren wurde eine neue Kamera in Blender erstellt. Diese neue Kamera erhält durch ein Constraint die Transformationsdaten der getrackten Kamera. Die Animationsdaten der Kamera werden durch „Link Animation Data“ (Strg+L) weitergegeben, sodass alle Keyframes vorhanden sind. Die Kamera wurde dann als FBX-Datei für die *Unreal Engine* exportiert. In der *Unreal Engine* wurde diese FBX-Datei für einen bereits erstellten Kamera-Actor in einer Level Sequence importiert. Beim Abspielen der Level Sequence fiel auf, dass die Kameraanimation an einige Stellen Sprünge hatte, die in der realen Kamerabewegung nicht vorhanden waren, die in Abbildung 5.23 zu sehen sind. Die Keyframes an diesen Stellen wurden direkt in der Unreal Engine entfernt, wodurch die Kamerabewegung weicher wurde.

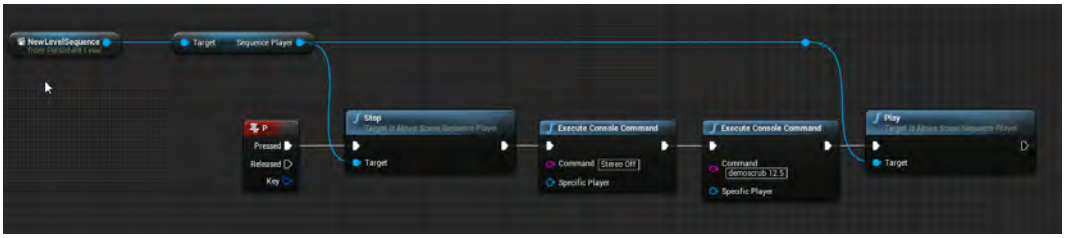

Abbildung 5.24: Funktion zum gleichzeitigen Starten der Kameraanimation und des Replays. Der Konsolenbefehl „demoscrub 12.5“ springt zur passenden Stelle im Replay.

Danach stellte es allerdings ein Problem dar, den Start der Kameraanimation mit dem Start des Replays zu synchronisieren. Dafür wurde eine Funktion in der Unreal Engine programmiert, die das Replay an einem angegebenen Zeitpunkt startet und gleichzeitig die Kameraanimation in der Level Sequence. Diese Funktion ist in Abbildung 5.24 zu sehen. Der Start der Kameraanimation wurde in der Unreal Engine zunächst ungefähr mit dem Start des Replays synchronisiert und beides zusammen mit Hilfe von *NVidia's Shadowplay* aufgenommen. Die Footage wurde in das After Effects Projekt geladen und dort konnte genau erkannt werden, wie weit sich das Timing der Kamerafahrt vom Start des Replays unterscheidet. Der zeitliche Unterschied wurde dann zum festgelegten Startpunkt des Replays hinzugefügt, wodurch das Timing stimmte. Nach einigen Probieren und Vergleichen der Controllerpositionen und Kameraanimation in After Effects, wurde die passende Stelle im Replay gefunden. Durch einen Tastendruck wurde dann das Replay gleichzeitig mit der Kameraanimation gestartet. Dies wurde dann aufgezeichnet und die Aufnahme wieder in After Effects importiert und mit dem Realvideo synchronisiert.
Für die Aufnahme der virtuellen Umgebung wurden allerdings die Meshes der Motion Controller ausgeblendet, denn es war eine sich stetig ändernde Abweichung in der Position der realen und der virtuellen Motion Controller.
Abschließend mussten mehrere Masken in After Effects animiert werden, die die für das Tracking platzierten Markierung, das Stativ und die Bereiche neben dem Greenscreen ausblenden. Dies war bereits in Abbildung 5.20 zu sehen.

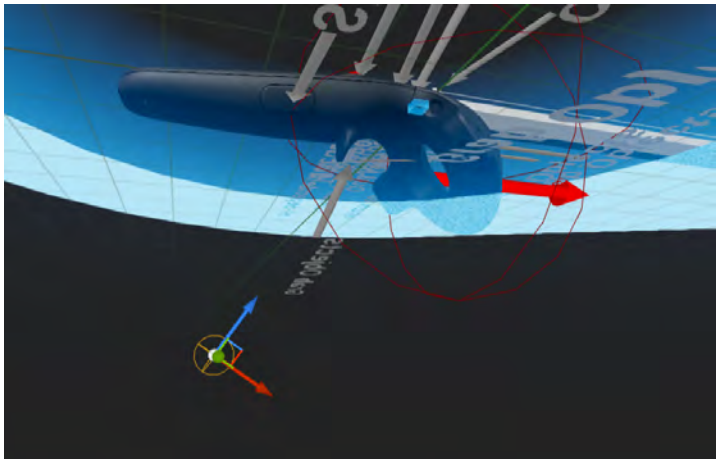

Abbildung 5.25: Die Sphere-Komponente „ControllerCamLocator“ im Motion Controller Actor.

5.5 Kameratracking mit einem VR-Controller

5.5.1 Implementierung

Die virtuelle Kamera muss mit einem der Motion Controller verbunden werden, der die reale Kamera trackt und die virtuelle Kamera dadurch positioniert. Dafür wird die Scene Capture Component an einen der beiden Motion Controller Actor „attached“, nachdem ein Button am rechten Controller gedrückt wurde. Dafür hat der Motion Controller Actor eine Komponente namens „ControllerCamLocator“, die so unterhalb des Controller Meshes platziert wurde, dass sie der Position des realen Kamerasensors gleichkommt, wenn der reale Controller über der Kamera platziert ist, was in Abbildung 5.25 zu sehen ist. Die Funktion zum „Attachen“ der Scene Capture Component 2D ist in Abbildung 5.26 zu sehen und befindet sich in der Blueprint des VR Pawns.

Ist die Kamera (SceneCapture2D) am Controller „attached“, kann man auf dem PC-Bildschirm erkennen, wie die Perspektive durch das Bewegen des Controllers verändert wird. Die Tastenbelegung muss unter Umständen so angepasst werden, dass benötigte Taste auch noch gedrückt werden können, wenn der Controller über der Kamera fixiert ist.

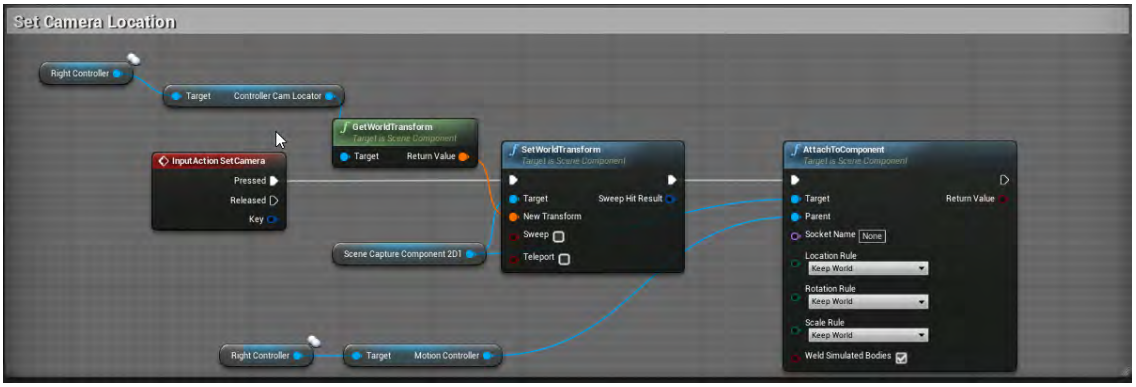

Abbildung 5.26: Die Funktion zum „Attachen“ der Scene Capture Component 2D. Die virtuelle Kamera wird zunächst an der Position des ControllerCamLocator's positioniert und anschließend an den rechten Controller „attachen“.

5.5.2 Aufnahme

Die Kamera kann für diese Compositing Technik wieder mit dem PC verbunden werden, außerdem wird der externe Kontrollmonitor am Kamera-Rig mit einem HDMI-Kabel als zusätzlicher Monitor am PC angeschlossen.
Hier wird ebenfalls OBS verwendet, um eine Live-Produktion zu simulieren. Außerdem wird die Vorschau von OBS auf dem Kontrollmonitor angezeigt. Abbildung 5.27 zeigt, wie die Vollbild-Vorschau aktiviert wird. Dadurch ist es dem Kameramann möglich virtuelle und reale Kameraperspektive gleichzeitig zu sehen, was in Abbildung 5.28 abgebildet ist. Dadurch kann die Kamera besser auf virtuelle Gegenstände ausgerichtet werden.

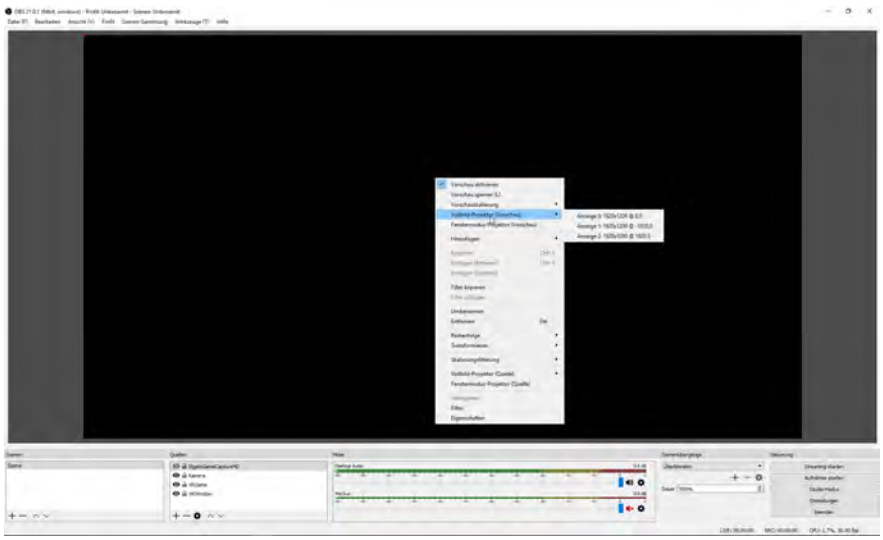

Abbildung 5.27: Abgebildet ist, wie die Vollbildvorschau für den Kontrollbildschirm in *OBS Studio* aktiviert wird.

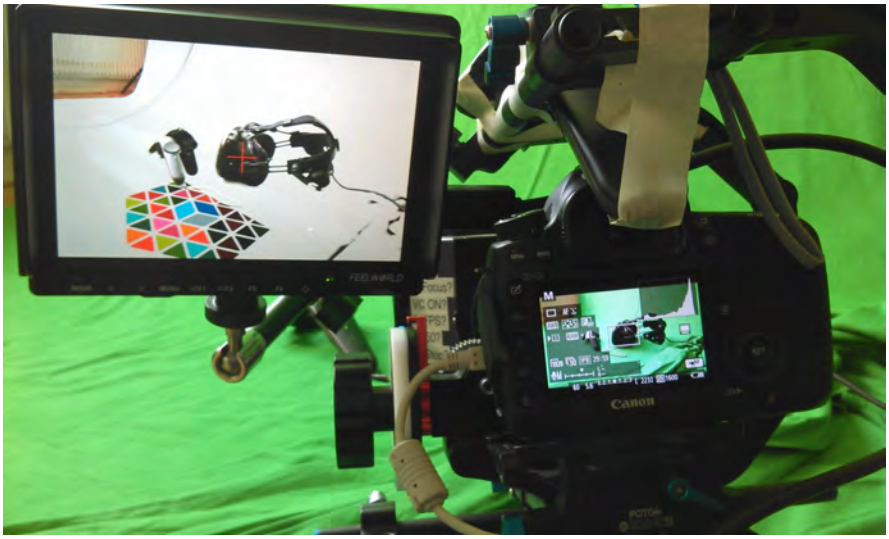

Abbildung 5.28: Auf dem Kontrollbildschirm am Kamera-Rig wird die Live-Vorschau von OBS angezeigt.

Abbildung 5.29: Zu sehen ist, wie der Vive-Controller am Kamera-Rig mit Klebeband fixiert wurde.

Der Vive Controller, zum Tracken der Kamera, wurde oberhalb von dieser mit Klebeband fixiert (Siehe Abbildung 5.29). Damit realer und virtueller Controller sich im Bild an derselben Stelle befinden, wurde der Controller zum Kameratracken, also eigentlich die virtuelle Kamera, so bewegt und ausgerichtet, dass die Controller-Positionen übereinstimmen. Dafür wurde der zweite Controller an ein Stativ vor der Kamera gehängt und mit Hilfe des Kontrollbildschirms die reale und virtuelle Position des Controllers verglichen. Nachdem der Controller korrekt ausgerichtet war, wurde er fixiert.
Beim Aufnehmen war darauf zu achten, dass stets Greenscreen hinter dem Schauspieler zu sehen ist, damit er freigestellt werden kann. Die Freiheit in der Kamerabewegung führt schnell zu derartigen Fehlern.
Zum Synchronisieren der Aufnahmen konnte bei dieser Compositing-Technik die Trigger-Taste des an der Kamera fixierten Controllers verwendet werden: Beim plötzlichen Loslassen macht diese ein lautes Geräusch. In der Unreal Engine war eine Funktion implementiert worden, die ein Standardgeräusch der Engine beim Loslassen des Triggers abspielt. Allerdings konnte nicht sofort nach dem Loslassen der Trigger-Taste mit den Aktionen des Schauspielers begonnen werden, denn der Controller bewegt sich durch das abrupte Loslassen kurz.

5.5.3 Post-Produktion

Für diese Compositing-Technik musste in der Post-Produktionsphase nach dem Synchronisieren der Videos, lediglich der Greenscreen freigestellt werden und eine Maske animiert werden, die die Außenbereichen des Greenscreens abdeckt.
Die Maske animieren beansprucht etwas Zeit, da die Kamerabewegung komplett frei war. Bei Fehlern des Kameramanns, durch die der Schauspieler im Bild nicht mehr vor dem Greenscreen ist, kann dies unter Umständen mit einer Maske korrigiert werden. Allerdings ist dies zeitaufwendiger und nur bei kleinen Fehler sinnvoll, wie zum Beispiel bei einen Ellenbogen der über den Greenscreen hinausragt.

Kapitel 6

Evaluation und Ergebnisse

In diesem Kapitel werden zunächst die bei Implementierung, Aufnahme und Post-Produktion der vier Mixed Reality-Compositing-Techniken gemachten Erfahrungen sowie benötigte Zeit bei der Umsetzung behandelt. Anschließend wird die durchgeführte Evaluation behandelt. Sie befasste sich damit, wie die verschiedenen Videos der Compositing-Techniken wahrgenommen werden. Abschließend wird das aus Evaluation, Zeitaufwand und Erfahrungen entstehende Gesamtergebnis aufgezeigt.

6.1 Vergleich der Mixed Reality-Compositing-Techniken

Die vier erstellten Videos, die auch in der Evaluation verwendet werden, sind in Anhang A zu finden. In diesem Unterkapitel werden die Besonderheiten, Möglichkeiten, Unterschiede und Gemeinsamkeiten der vier umgesetzten Mixed Reality-Compositing-Techniken behandelt, wie sie beim Umsetzen der Techniken aufgefallen sind. Folgend werden auch die Zeitaufwände eingeschätzt, die zu erwarten sind, wenn die Compositing-Technik angewandt wird, und diese Schätzung begründet. Dabei wird davon ausgegangen, dass die Vorgehensweise zuvor bekannt ist oder einem Leitfaden gefolgt wird und ein Video, das nicht länger als eine Minute ist, produziert wird. Somit wird vom Zeitaufwand, der bei der Umsetzung dieser Arbeit nötig war, abgewichen. Denn teilweise waren umfangreiche Recherchen oder die Entwicklung des Umsetzungsprozesses nötig.
Für alle, in dieser Arbeit behandelten Mixed Reality-Compositing-Techniken, war zunächst derselbe, folgende Grundaufwand nötig: Die in Kapitel 5 Umsetzung der Mixed Reality-Compositing-Techniken, im Unterkapitel 5.1 Entwicklung der Testumgebung behandelte Implementierung des VR Spectator Screens sowie das Anpassen der virtuellen Kamera an die reale Kamera und das Ändern der Farbkorrektur und Belichtung mussten in der *Unreal Engine* zunächst durchgeführt werden. Dafür ist ein Zeitaufwand von ein bis drei Stunden zu erwarten. Dieser ist abhängig davon, wie viele Anpassungen an dem virtuellen Kamerabild nötig sind, um das gewünschte Bild zu erreichen. Außerdem war für alle Compositing-Techniken der Aufbau eine Greenscreens und der für diesen benötigte Beleuchtung nötig. Für den Aufbau ist von einem Zeitaufwand von circa ein bis zwei Stunden auszugehen, abhängig von der Größe des Greenscreens.

Die Umsetzung der Mixed Reality-Compositing-Technik „**Statische Kamera**“ ist von den vier in dieser Arbeit umgesetzten Techniken am einfachsten und schnellsten umzusetzen. Der Programmieraufwand in der *Unreal Engine* war nicht besonders hoch, da lediglich eine Funktion zur akkuraten Positionierung der Kamera entwickelt werden musste. Außerdem ließe sich die Kameraposition auch präzise vermessen, anstatt die Motion Controller zu verwenden, wenn nichts programmiert werden soll. Mit einem Zeitaufwand von circa einer Stunde kann gerechnet werden.
Der Aufnahmenprozess war bei dieser Technik nicht sehr aufwendig, denn die Kamera muss nur positioniert und richtig ausgerichtet werden. Anschließend können die Aufnahmen in der Anwendung und an der Kamera bereits gestartet werden.
Beim Post-Produktionsprozess war neben dem Synchronisieren der Aufnahmen und dem Auskeyen des Greenscreens nur noch eine Maske anzulegen, um Bereiche neben dem Greenscreen auszublenden. Wenn ein größerer Greenscreen verwendet wird, wäre dies jedoch nicht nötig.
Diese Compositing-Technik eignet sich für Live-Produktionen, denn die in der Post-Produktion durchgeführten Aktionen könnte auch *OBS Studio* durchführen. Es wird neben der Grundausstattung für die Mixed Reality-Aufnahmen lediglich noch ein Stativ für diese Technik benötigt. Außerdem ist kein zu großer Greenscreen bei Live-Produktionen von Nöten, denn die Kamera wird während der Aufnahme nicht anders ausgerichtet werden. Allerdings ist diese Technik offensichtlich aufgrund der statischen Kamera eingeschränkt.
Das Verwenden einer statischen Kamera ist zum Beispiel für Aufnahmen von Endnutzern einer VR-Anwendung in ihren privaten Räumlichkeiten sehr gut geeignet. Denn es wird wenig technische Ausstattung benötigt und die Umsetzung ist sehr einfach. Der Entwickler einer VR-Anwendung müsste hierfür lediglich die nötige Darstellung am Hauptbildschirm des Computers entsprechend anpassen und das freie Positionieren der externen, virtuellen Kamera ermöglichen.
Ungeeignet ist die Technik jedoch für Produktionen, bei denen Kamerabewegungen zwingend notwendig sind. Wie zum Beispiel bei Filmproduktion, bei denen die Kamerabewegung zum Storytelling beiträgt.

Bei der Mixed Reality-Compositing-Technik „**Festgelegte Kamerabewegung**“ ist ein hoher Vorbereitungsaufwand nötig. Denn vor den Aufnahmen muss die Kamerabewegung genau festgelegt sein, damit sie sowohl real als auch virtuell akkurat ausgeführt werden kann. Vorteilhaft ist hierbei, dass Fehler bei der Ausrichtung der Kamera frühzeitig bemerkt und verbessert werden können, während sie bei freien Kamerabewegungen unter Umständen erst bei der Post-Produktion auffallen. Die Implementierung in der *Unreal Engine* ist nicht besonders kompliziert, da nur eine Kamerafahrt animiert werden muss, zum Beispiel auch mit dem *Level Seqeuencer*, und eine Funktion die diese startet. Trotzdem kann das Animieren der Kamerafahrt, abhängig von deren Komplexität, länger dauern. Außerdem sollte, wie für die statische Kamera auch, eine Funktion zur Positionsbestimmung der Kamera entwickelt werden. Somit ist ein Zeitaufwand von zwei bis fünf Stunden für Implementierung und Vorbereitung realistisch. Eine aufwendige Kamerafahrt kann aber nur mit der richtigen technischen Ausstattung durchgeführt werden, wie zum Beispiel einen automatisierten Ka-

meraarm. Bei der Verwendung eines einfachen Kamerasliders, wie er auch in dieser Arbeit verwendet wurde, ist der Zeitaufwand für die Implementierung deutlich geringer.
Für die Aufnahme ist durch die Verwendung zusätzlicher Technik ein längerer Aufbau nötig. Außerdem wird je nach gewünschter Kamerabewegung auch sehr spezielle und häufig teure Kameratechnik benötigt, die komplett automatisiert werden kann. Dies gilt es bei Verwendung dieser Technik besonders zu beachten.
Wenn allerdings ein handbetriebener Kameraslider verwendet wird, sind unter Umständen auch zusätzliche Aufnahmen nötig, bis die Kamerabewegungen synchron sind. Zusätzlich muss beim Ausrichten und der Positionsbestimmung der Kamera durch Motion Controller oder Vermessung beachtet werden, dass sowohl zu Beginn als auch am Ende der Kamerafahrt virtuelle und reale Kameras noch gleich ausgerichtet sind und keine Abweichungen bei den Controllerpositionen auftreten. Der Zeitaufwand für die Aufnahme kann, wegen der genannten Gründe, zwischen einer und fünf Stunden liegen.
Der Postproduktionsprozess läuft ähnlich ab, wie für die statische Kamera. Zusätzlich ist aber das Animieren einer Maske nötig, wenn der Greenscreen nicht groß genug war und Bereiche neben diesen ausgeblendet werden müssen. Abhängig von der Art der Kamerabewegung kann dies auch länger dauern. Ein Zeitaufwand von circa zwei Stunden ist somit zu erwarten.
Wenn die Kamerabewegung automatisiert ist oder auch im Handbetrieb gut synchronisiert wird, ist auch mit dieser Technik eine Live-Produktion möglich. Dabei können aber leicht Fehler auftreten, wenn zum Beispiel die Kamerabewegung nicht ausführlich getestet wurde. Außerdem ist diese Compositing-Technik nicht sehr flexibel. Wenn während des Aufnahmeprozesses Änderungen an der Kamerafahrt vorgenommen werden sollen, muss zusätzlich in der Engine die Kamerabewegung geändert werden. Erst danach kann die Aufnahme stattfinden, denn virtuelle und reale Aufnahme finden parallel statt.
Eignen kann sich diese Technik zum Beispiel für Produktionen in einem Fernsehstudio. Es können Kamerafahrten für vorhandene, automatisierte Kameratechnik, wie zum Beispiel Kamerakräne, vor einer Show programmiert werden. Die Kamerabewegungen werden während der Aufnahme oder der Live-Austrahlung der Show auch in der virtuellen Umgebung durchgeführt, um Virtualität und Realität zusammen zu setzen.
Ungeeignet ist diese Technik jedoch, wenn spontane Aufnahme mit unvorhergesehenen Kameraausrichtungen gemacht werden soll, wie es zum Beispiel nötig wäre um auf Aktionen des VR-Anwenders zu reagieren.

Die Mixed Reality-Compositing-Technik „**Rekonstruktion der Kamerabewegung**“ ist am aufwendigsten in der Post-Produktion, benötigt aber wenig Implementierungsarbeit. So mussten in der *Unreal Engine* nur die Einstellungen für das Replay System vorgenommen werden, aber auch getestet werden, ob dieses richtig funktioniert. Das Testen und das Aktivieren der Replication für Actor, die aufgezeichnet werden sollen, kann dabei abhängig von der Anzahl der aufzuzeichnenden Actor einiges an Zeit in Anspruch nehmen. Abhängig davon liegt der Zeitaufwand zwischen einer und drei Stunden.
Bei der Aufnahme ist zusätzlich der Einsatz eines Kamerarigs empfehlenswert. Im Prinzip kann die Aufnahme frei sein, sollte aber ruhig sein, damit sie getrackt werden kann. Außerdem müssen Fixpunkte im Kamerabild vorhanden sein. Solche wurden vor dem Start der

Aufnahme am Greenscreen angebracht. Trotzdem ist nicht mit mehr als zwei Stunden Zeitaufwand zu rechnen.
Der Post-Produktionsprozess war bei dieser Technik am umfangreichsten. Zunächst musste die Kamera erfolgreich getrackt werden, bei ungünstigen Trackingpunkten war dies jedoch nicht möglich. Ein Beispiel dafür ist das fünfte Video in Anhang A, bei dem die dreidimensionale Bewegung der Kamera nicht richtig getrackt wurde und virtuelle und reale Kamera sich dadurch unterschiedlich schnell nach vorne bewegen.
Nachdem die Kamera erfolgreich getrackt worden war, musste sie aus *After Effects* über *Blender* in die *Unreal Engine* exportiert werden und dabei richtig positioniert werden. Anschließend musste die virtuelle Kamerfahrt mit dem Replay der Anwendung synchronisiert werden, um dann aufgenommen zu werden. Schließlich konnte in *After Effects* der Greenscreen ausgekeyt werden und eine Maske für die Bereiche neben diesen animiert werden, die für diese Technik genauer animiert werden musste, da die Kamerabewegung komplett frei war. Bei diesem Post-Produktionsprozess können an vielen Stellen Fehler auftreten, die eventuell auch dazu führen, dass eine neue Aufnahme notwendig wird. Bei der Post-Produktion dieser Technik ist mit einem Zeitaufwand von circa acht Stunden zu rechnen, der auch davon abhängt, wie schnell der Computer die dreidimensionale Kamerabewegung berechnen kann.
Die Aufnahmen der Virtualität und Realität finden hier versetzt statt, da eine Kamera nicht anhand ihres Bildes in Echtzeit getrackt werden kann und somit ist auch keine Live-Produktion mit dieser Technik möglich. Einen Vorteil bietet diese Technik aber durch das Ermöglichen einer relativ freien Kamerabewegung und die Komplexität der Kamerabewegung hat im Gegensatz zur festgelegten Kamerabewegung keinen großen Einfluss auf die Umsetzungsdauer. Allerdings ist die Umsetzungsdauer insgesamt bereits sehr hoch.
Der aufwändige Post-Produktionsprozess hat den Vorteil, dass viele Anpassungen noch in der Post-Produktion vorgenommen werden können. So kann ein fehlerhaftes Kameratracking auch manuell solange angepasst werden, bis es genau zur Aufnahme passt und eventuell noch besser wirkt, als beim Tracking der Kamera mit VR-Controller. Dadurch eignet sich diese Technik gut für umfangreiche Filmproduktionen, wie zum Beispiel Werbefilme mit denen VR demonstriert werden soll.
Da diese Technik eine freie Kameraausrichtung erlaubt, eignet sie sich auch, um spontan auf Aktionen des VR-Anwenders zu reagieren. Das Problem dabei ist jedoch, dass der Kameramann nicht sehen kann, auf was die Kamera in der virtuellen Umgebung gerade ausgerichtet ist.

Wie bei der Rekonstruktion der Kamerabewegung bietet auch die Mixed Reality-Compositing-Technik „**Kameratracking mit einem VR-Controller**“ die Möglichkeit eine Kamera frei zu bewegen. Zusätzlich bietet das Tracking mit VR-Controllern den Vorteil, die Ausrichtung der virtuellen Kamera ohne Zeitversatz sehen zu können.
Bei der Implementierung dieser Technik war Testen sehr wichtg. So musste die virtuelle Kamera möglichst genau wie die reale Kamera unter dem virtuellen VR-Controller platziert werden. Wenn für die Aufnahme ein dritter Motion Controller zum Tracken der Kamera verwendet werden soll, ist es nötig diesen in die *Unreal Engine* zu integrieren. Dadurch entsteht ein Zeitaufwand von einer bis drei Stunden.
Bei der Aufnahme war der Aufwand sehr gering. Die Besonderheiten bei dieser Technik waren

die Verwendung eines Kamera-Rigs, wie auch bei der Rekonstruktion der Kamerabewegung, an das ein Motion Controller befestigt wurde. Zum Ausgleichen von Abweichungen bei den Controllerpositionen im Bild musste der befestigte Controller neu ausgerichtet werden, da die Kamera im Rig nur zur Seite gedreht werden konnte. Außerdem wurde bei der Aufnahme ein mit dem Computer verbundener Kontrollbildschirm verwendet, um einem Kameramann eine Live-Vorschau aus *OBS Studio* anzuzeigen. Trotzdem liegt der zu erwartende Zeitaufwand bei der Aufnahme bei einer bis zwei Stunden, denn alle zuvor aufgeführten Punkte lassen sich sehr schnell umsetzen.
Der Post-Produktionsprozess ist nahezu identisch mit dem der festgelegten Kamerabewegung, außer dass die animierte Maske etwas aufwendiger zu animieren ist, da die Kamerabewegung komplett frei sein kann. Somit liegt der Zeitaufwand für die Post-Produktion bei circa zwei Stunden.
Ein Nachteil dieser Technik ist, dass wenn zwei Motion Controller vom aufzunehmenden VR-Anwender verwendet werden sollen, ein dritter benötigt wird, um die Kamera zu tracken.
Wie bereits durch die Verwendung von *OBS Studio* für die Live-Vorschau klar wird, ist diese Compositing-Technik für Live-Produktionen geeignet. Dabei kann die Kamerabewegung komplett frei sein und ist nur durch die Grenzen des Greenscreens eingeschränkt. Dadurch kann diese Technik sinnvoll bei Live-Fernsehproduktionen sein, die sowohl Compositing mit einer virtuelle Umgebung als auch freie Kamerabewegungen benötigen.
Diese Technik kann auch bei Live-Demonstrationen von Virtual Reality, zum Beispiel auf Messen, sinnvoll eingesetzt werden. Dabei kann ein VR-Anwender vor einem Greenscreen mit gefilmt werden, um ihn auf einem Bildschirm für andere Besucher der Messe direkt in der virtuellen Umgebung darzustellen und darüber hinaus könnten andere Besucher die getrackte Kamera selbst bedienen, um sich frei umzusehen.
Abhängig vom verwendeten VR-System ist das Tracking der Kamera mit Hilfe von VR-Controller durchaus sehr präzise und findet in Echtzeit statt. Trotzdem sind kleine Ungenauigkeit beim Tracking möglich, wodurch vor allen Aufnahmen mit hoher Brennweite oder hohem digitalen Zoom problematisch werden. Denn bereits kleinste Sprünge im Tracking stören das Bild sehr stark.

In Tabelle 6.1 ist noch einmal der geschätzte Zeitaufwand der vier Compositing-Techniken dargestellt. Für alle Techniken kommen noch ein bis drei Stunden für die grundlegende Implementierung und circa zwei Stunden für den Grundaufbau hinzu. In Tabelle 6.2 ist der tatsächliche Zeitaufwand dargestellt, der bei der Umsetzung der Compositing-Techniken bei *Qubic* nötig war. Zusätzlich waren noch circa zwölf Stunden für die grundlegende Implementierung und neun Stunden für den Grundaufbau erforderlich. Es ist zu bedenken, dass im tatsächlichen Zeitaufwand die Zeit für Recherchen zu benötigten Techniken, Tests, erneute Versuche, Umbau von einer Technik zur anderen und für das Bekanntmachen mit neuer Technik enthalten ist.

Tabelle 6.1: Geschätzte Zeitaufwände der vier Mixed Reality-Compositing-Techniken.

	Statische Kamera	**Festgelegte Kamerabewegung**	**Rekonstruktion der Kamerabewegung**	**Kameratracking mit VR-Controller**
Implementierung	1 h	2-5 h	1-3 h	1-3 h
Aufnahme	1 h	1-5 h	2 h	1-2 h
Post-Produktion	1 h	2 h	8 h	2 h
Gesamt	3 h	5-12 h	11-13 h	4-7 h

Tabelle 6.2: Tatsächliche Zeitaufwände der vier Mixed Reality-Compositing-Techniken im Rahmen dieser Arbeit.

	Statische Kamera	**Festgelegte Kamerabewegung**	**Rekonstruktion der Kamerabewegung**	**Kameratracking mit VR-Controller**
Implementierung	5 h	6 h	4 h	4 h
Aufnahme	5 h	6 h	3 h	4 h
Post-Produktion	3 h	3 h	12 h	2 h
Gesamt	13 h	15 h	19 h	10 h

6.2 Evaluation

Zum Evaluieren der vier umgesetzten Compositing-Techniken wurde eine Umfrage erstellt, mit der Befragte die erstellten Videos bewerten sollen. In dieser Umfrage wird zunächst zu jedem der vier erstellten Videos gefragt, wie sehr die Art der Kamerabewegung dem / der Befragten gefällt, wie natürlich die Kamerabewegung wirkt und wie integriert in die virtuelle Umgebung der VR-Anwender wirkt. Am Ende der Umfrage sollen die Befragten die Compositing-Techniken anhand ihrer Kamerabewegungen in einer Rangfolge einordnen und die Compositing-Technik auswählen, deren Kamerabewegung am meisten dabei hilft, die VR-Interaktionen des Anwenders zu verstehen. Screenshots des Evaluations-Formulars sind in Anhang B Evaluationsfragen zu finden.
Die Umfrage hat keine bestimmte Zielgruppe, denn auch das Verwenden von VR ist nicht nur an eine eingegrenzte Anwendergruppe gerichtet. Somit sollten Videos zum Illustrieren der virtuellen Realität auch nicht die Zielgruppe einschränken.

An der Umfrage haben zwölf Personen teilgenommen. Nachfolgend werden die Ergebnisse der Evaluation behandelt. Die vollständigen Ergebnisse der Umfrage sind in Tabellenform in Anhang C Evaluationsergebnisse zu finden.
Die ersten Fragen, die sich jeweils immer auf eine Compositing-Technik beziehen, konnten mit einem Wert von eins bis zehn bewertet werden. Die durchschnittliche Bewertung der vier Mixed Reality-Compositing-Techniken anhand der drei Fragen „Wie ansprechend ist die Art der Kamerabewegung des Videos?“, „Wie natürlich wirkt die Kamerabewegung auf Sie?“ und „Wie integriert wirkt der Virtual Reality-Anwender in die virtuelle Umgebung?“ ist in Abbildung 6.1 dargestellt.
Es fällt auf, dass die vierte Compositing-Technik, für die die Kamera mit einem VR-Controller getrackt wurde, bei allen Fragen im Durchschnitt am besten bewertet wurde. Alle vier Arten der Kamerabewegungen werden von den Befragten im Durchschnitt eher als Positiv bewertet. Allerdings gefällt die festgelegte Kamerabewegung auf einem Slider am wenigsten und mit nur 0,83 Punkten mehr gefällt die mit VR-Controller getrackte Bewegung den Befragten im Durchschnitt am besten. Also sind die Differenzen hier nicht besonders hoch.
Bei der Frage, ob die Kamerabewegungen natürlich wirken, sind klarere Differenzen zu sehen. So wirkt die festgelegte Kamerabewegung im Durchschnitt am unnatürlichsten. Die freien Kamerabewegungen wirken am natürlichsten, wobei die rekonstruierte Kamerabewegung von den Befragten als weniger natürlich, als die mit VR-Controller getrackte Kamerabewegung wahrgenommen wird. Bei den Compositing-Techniken Eins und Vier wirkt der VR-Anwender für die Befragten am integriertesten und in der zweiten Compositing-Technik ist der Anwender nach Angaben der Befragten nahezu genauso gut integriert. Nur bei der dritten Compositing-Technik, der Rekonstruktion der Kamerabewegung, ist der Anwender den Befragten zufolge deutlich schlechter in die virtuelle Umgebung integriert.

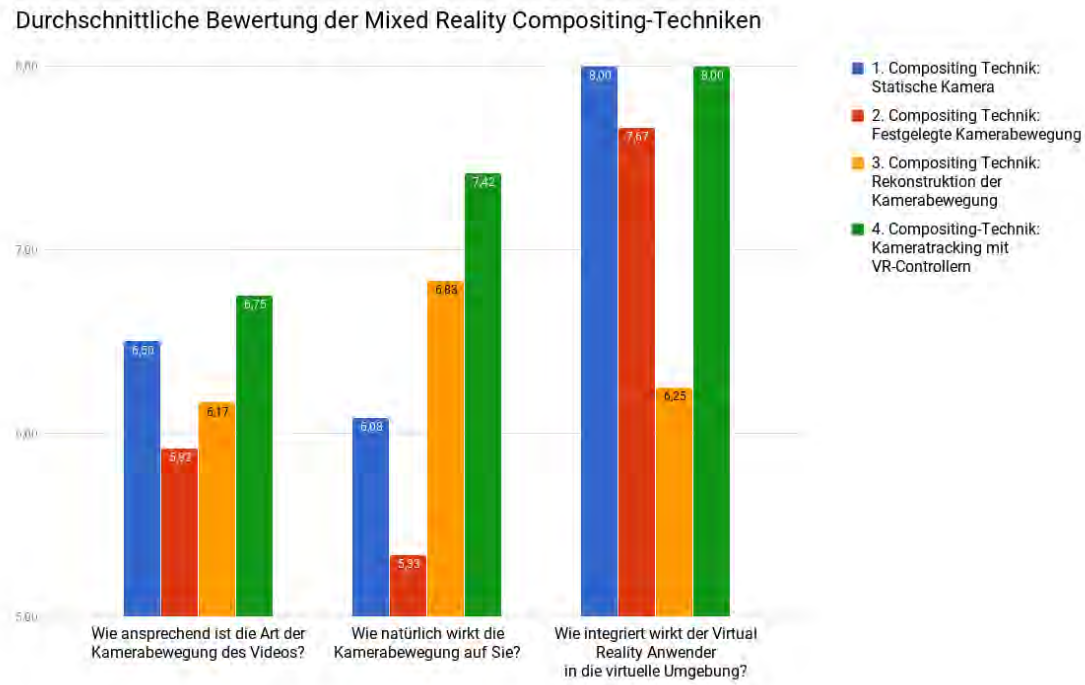

Abbildung 6.1: Durchschnittliche Bewertung der Mixed Reality-Compositing-Techniken

Im abschließenden Teil der Umfrage sollten die vier Mixed Reality-Compositing-Techniken in eine Rangfolge eingeordnet werden, basierend darauf welche Kamerabewegung den Befragten am besten gefiel. Das Gesamtergebnis dieser Einordnung ist in Abbildung 6.2 zu sehen. Aus den durchschnittlichen Platzierungen der Compositing Techniken ergibt sich die in Tabelle 6.3 zu sehende Rangfolge. Dieser Rangfolge kann entnommen werden, dass die Compositing-Technik „Kameratracking mit VR-Controller“ den Befragten im Durchschnitt am meisten zusagte und die „festgelegte Kamerabewegung“ am wenigsten. Die Compositing-Techniken „statische Kamera“, mit einer durchschnittlichen Platzierung von 2,42, und „Rekonstruktion der Kamerabewegung“, mit einer durchschnittlichen Platzierung von 2,5, gefielen den Befragten ähnlich gut.

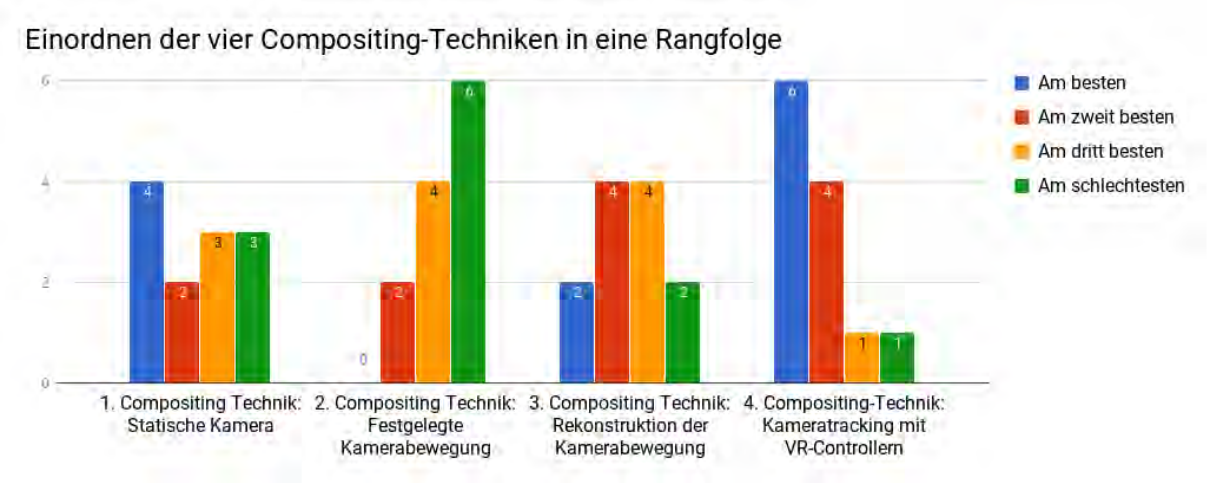

Abbildung 6.2: Einordnen der vier Mixed Reality-Compositing-Techniken in eine Rangfolge

Tabelle 6.3: Rangfolge der vier Mixed Reality-Compositing-Techniken.

Platzierung	**Durchschnittlicher Platzierungswert**	**Mixed Reality-Compositing-Technik**
1.	1,75	4. Compositing-Technik **Kameratracking mit VR-Controller**
2.	2,42	1. Compositing-Technik **Statische Kamera**
3.	2,5	3. Compositing-Technik **Rekonstruktion der Kamerabewegung**
4.	3,33	2. Compositing-Technik **Festgelegte Kamerabewegung**

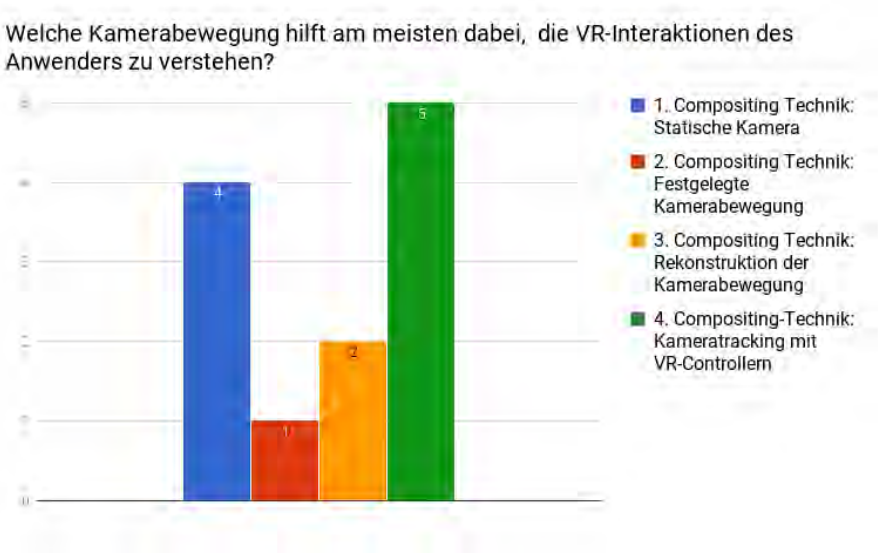

Abbildung 6.3: Welche Kamerabewegung hilft am meisten dabei, die VR-Interaktionen des Anwenders zu verstehen?

Abschließend wurde danach gefragt, welche Compositing-Technik am meisten dabei hilft, die VR-Interaktionen des VR-Anwenders zu verstehen. Das Ergebnis dazu ist in Abbildung 6.3 zu sehen. Es ist zu erkennen, dass das Kameratracking mit VR-Controller und die statische Kamera am meisten zum Verständnis der VR-Interaktionen beitragen, während die anderen beiden Compositing-Techniken dabei nur von wenigen als hilfreich angesehen werden.

6.3 Zusammengefasste Ergebnisse

Aus der Evaluation, dem geschätzten und dem tatsächlichen Zeitaufwand geht hervor, dass die Compositing-Techniken „statische Kamera“ und „Kameratracking mit VR-Controller“ am meisten zu empfehlen sind. Denn sie wurden von den Befragten am besten bewertet und benötigen am wenigsten Zeit in der Umsetzung. Außerdem gibt es bei beiden Techniken, wenige von der Technik ausgehende Probleme, die die Umsetzung erschweren oder verlängern können.

Im Vergleich dazu erzeugt die Compositing-Technik „festgelegte Kamerabewegung“ beim Verwenden von einfacher, technischer Ausstattung, wie die Verwendung eines handbetriebenen Kamerasliders, der Evaluation zu Folge kein besonders gutes Ergebnis. Mit komplett automatisierter Kamerasteuerung kann ein besseres Ergebnis erreicht werden, aber trotzdem ist ein hoher Vorbereitungsaufwand nötig und die Technik ist nicht flexibel.

Die Compositing-Technik „Rekonstruktion der Kamerabewegung“ besitzt eine Kamerabewegung, die von Befragten als relativ natürlich wahrgenommen wurden (siehe Abbildung 6.1), hebt sich aber ansonsten in der Evaluation nicht besonders hervor. Der benötigte Zeitaufwand für diese Technik ist in der Regel um einiges höher, als der der anderen Techniken, vor allem im Post-Produktionsprozess. Außerdem kann es beim Tracking oder der Rekonstruktion der Kamerabewegung zu Problemen und Fehlern kommen, die erneute Aufnahmen nötig machen.

Insgesamt sind die beiden Mixed Reality-Compositing-Techniken „statische Kamera“ und „Kameratracking mit VR-Controller“ am sinnvollsten für den Einsatz bei Mixed Reality-Video-Produktionen. Wenn Kamerafahrten auf einem Slider oder vergleichbarem durchgeführt werden sollen, wie bei der Compositing-Technik „festgelegte Kamerabewegung“, kann trotzdem ein VR-Controller zum Tracking der Kamera verwendet werden, um Probleme zu vermeiden.
Ob eine statische Kamera verwendet werden soll oder die Kamera mit einem VR-Controller getrackt werden soll, ist abhängig von dem gewünschten Video und muss vom Video-Produzenten entschieden werden.

Kapitel 7

Zusammenfassung und Ausblick

In diesem Kapitel wird die Arbeit zunächst abschließend zusammengefasst und dabei auf die Ergebnisse eingegangen. Außerdem wird ein Ausblick darauf gegeben, wie die vier Mixed Reality-Compositing-Techniken eingesetzt werden können und wie sie entwickelt werden können.

7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurden die vier Mixed-Reality Compositing-Techniken „statische Kamera“, „festgelegte Kamerabewegung“, „Rekonstruktion der Kamerabewegung“ und „Kameratracking mit VR-Controller“ erfolgreich mit der *Unreal Engine* umgesetzt und in einer Umfrage bewertet. Bei der Umsetzung der Compositing-Techniken fielen deutliche Unterschiede beim nötigen Aufwand der vier Techniken auf. Vor allen beim Verwenden einer statischen Kamera ist der Zeitaufwand gering und weder Implementierung noch der Post-Produktionsprozess besonders aufwendig. Ähnlich sieht es beim Tracking der Kamera mit Hilfe von VR-Controllern aus, allerdings ist dabei in der Regel ein etwas höherer Implementierungsaufwand nötig und auch in der Post-Produktion kann mehr Aufwand nötig sein, wenn eine Maske animiert werden muss.
Deutlich ist aber, dass die anderen beiden Compositing-Techniken um einiges aufwendiger sein können. So wird bei der festgelegten Kamerabewegung eine ausführliche Vorbereitung benötigt und bei der Rekonstruktion der Kamerabewegung ist der Post-Produktionsprozess sehr umfangreich. Außerdem haben beide Techniken den Nachteil, dass bei Problemen in die vorige Phase zurückgegangen werden muss. Wenn bei der festgelegten Kamerabewegung während der Aufnahme Ungenauigkeiten bei den Kamerabewegungen auffallen, muss in den Implementierungsprozess zurück, um Anpassungen an der Kamerabewegung zu machen. Und wenn bei der Rekonstruktion der Kamerabewegung während der Post-Produktion Probleme beim Tracken der Kamera auftreten oder die getrackte Kamerabewegung nicht zur realen Kamerabewegung passt, muss neu aufgenommen werden.

Bei der Umsetzung der vier Compositing-Techniken konnte auch erkannt werden, dass mit der richtigen technischen Ausstattung einige Probleme gelöst oder verringert werden können. So kann ein größerer Greenscreen beziehungsweise ein Greenscreen-Raum verhindern, dass Bereiche außerhalb des Greenscreens aufgenommen werden und dadurch Masken in der Post-Produktion animiert werden müssen. Bei der Compositing-Technik „festgelegte Kamerabewegung“ hilft automatisierbares Kamerazubehör, um virtuelle und reale Kamerabewegung identisch und gleichzeitig durchzuführen. Zum Rekonstruieren der Kamerabewegung kann eine bessere Tracking-Software helfen, da der 3D Kameratracker von *After Effects* deutlich eingeschränkt ist.

Eine weitere gemachte Feststellung ist, dass die Compositing-Techniken „statische Kamera“, „festgelegte Kamera“ und „Kameratracking mit VR-Controller“ für Live-Produktionen geeignet sind. Dies wurde durch die Verwendung von *OBS Studio* getestet. Dabei ist zu beachten, dass bei einer Live-Produktion keine Maske animiert werden kann und somit ein ausreichend großer Greenscreen benötigt wird.

Es konnte auch in Erfahrung gebracht werden, dass für die *Unreal Engine* im Vergleich zu *Unity* kaum Plugins oder Anleitungen für Mixed Reality-Aufnahme existieren. Allerdings gibt es für die *Oculus Rift* ein *Unreal Engine*-Plugin und eine zugehörige Anleitung, mit dem Mixed Reality-Aufnahmen relativ einfach zu machen sind. Für die *HTC Vive* existiert jedoch nichts Vergleichbares.

7.2 Ausblick

Auch schon vor dieser Arbeit wurden Mixed Reality-Videos unter Verwendung von statischen Kameras oder mit VR-Controller zum Tracken einer Kamera erstellt. Diese Arbeit zeigte zwei weitere Compositing-Techniken, um dies zu ermöglichen, und wie die *Unreal Engine* für Mixed Reality-Videos eingesetzt werden kann.
In Zukunft müssen mit diesen Techniken aber nicht nur Mixed Reality-Videos erstellt werden, um Virtual Reality zu demonstrieren. Auch generell wenn ein Compositing mit Echtzeit-Renderings durchgeführt werden soll, sind die gezeigten Compositing-Techniken sinnvoll. Insbesondere für Live-Produktion, bei denen in Echtzeit Compositings durchgeführt werden müssen oder sollen, können drei der gezeigten Compositing-Techniken eingesetzt werden.
Ein Problem kann dabei jedoch die Tracking-Genauigkeit der Room-Scale VR-Systeme sein, denn sie sind nicht für sehr hohe Präzision ausgelegt. Trotzdem kann die Präzision für viele Kameraperspektiven ausreichen, wenn nicht sehr kleine Bewegungen genau getrackt werden müssen.

Ein Feature, das noch für die *Unreal Engine* entwickelt werden sollte, ist eine VR-System unabhängige Möglichkeit Vordergrund und Hintergrund des VR-Anwenders getrennt zu rendern, damit der Anwender besser in die Umgebung integriert werden kann. Ein solches Feature bietet das *Oculus Rift* Plugin für die *Unreal Engine* und sollte folglich bereits umsetzbar sein.

In Zukunft wäre es sinnvoll, die verschiedenen Room-Scale VR-Headsets, wie *HTC Vive* und *Oculus Rift* ausführlich auf ihre Unterschiede beim Einsetzen für Mixed Reality-Compositings zu untersuchen. Dadurch kann zum Beispiel herausgefunden werden, ob die Unterschiede in der Tracking-Präzision der Systeme überhaupt relevant sind. Außerdem kann ein Vergleich zwischen den Plugins gezogen werden, die für das Mixed Reality-Compositing eingesetzt werden.

Weiterhin ist es sinnvoll zu prüfen, ob für die Compositing-Technik „Rekonstruktion der Kamerabewegung“ eine andere Software zum Tracken der Kamera verwendet werden kann, um ein besseres Ergebnis zu erreichen. In dieser Arbeit wurde nur der 3D Kameratracker von *After Effects* verwendet. Dieser Kameratracker kann zum Beispiel mit den Kameratrackern von *Mocha Pro*[1] oder *Nuke*[2] verglichen werden.

[1] https://borisfx.com/products/mocha/
[2] https://www.foundry.com/products/nuke

Anhang A

Videos

Folgende Videos zu den Mixed Reality Compositing-Techniken sind mit dem angegebenen Dateinamen auf der DVD zu finden:

Nummer	Dateiname	Beschreibung
1	01-StatischeKamera	Dieses Video wurde unter der Verwendung einer statischen Kamera erstellt
2	02-FestgelegteKamerabewegung	Für dieses Video wurden die reale und virtuelle Kamerabewegung zuvor festgelegt und gleichzeitig ausgeführt. Dazu wurde ein Kameraslider benutzt.
3	03-RekonstruierteKamerabewegung	Die Kamerabewegung wurde bei diesem Video mit Hilfe von After Effects nach der Aufnahme getrackt und in der Unreal Engine wiederholt.
4	04-KameratrackingMitVRController	Zum Tracken der Kamera wurde für dieses Video ein VR-Controller an der Kamera befestigt.
5	05-FehlerhafteRekonstruktion	Dieses Video zeigt ein fehlerhaftes Kameratracking beim Rekonstruieren der Kamerabewegung

Anhang B

Evaluationsfragen

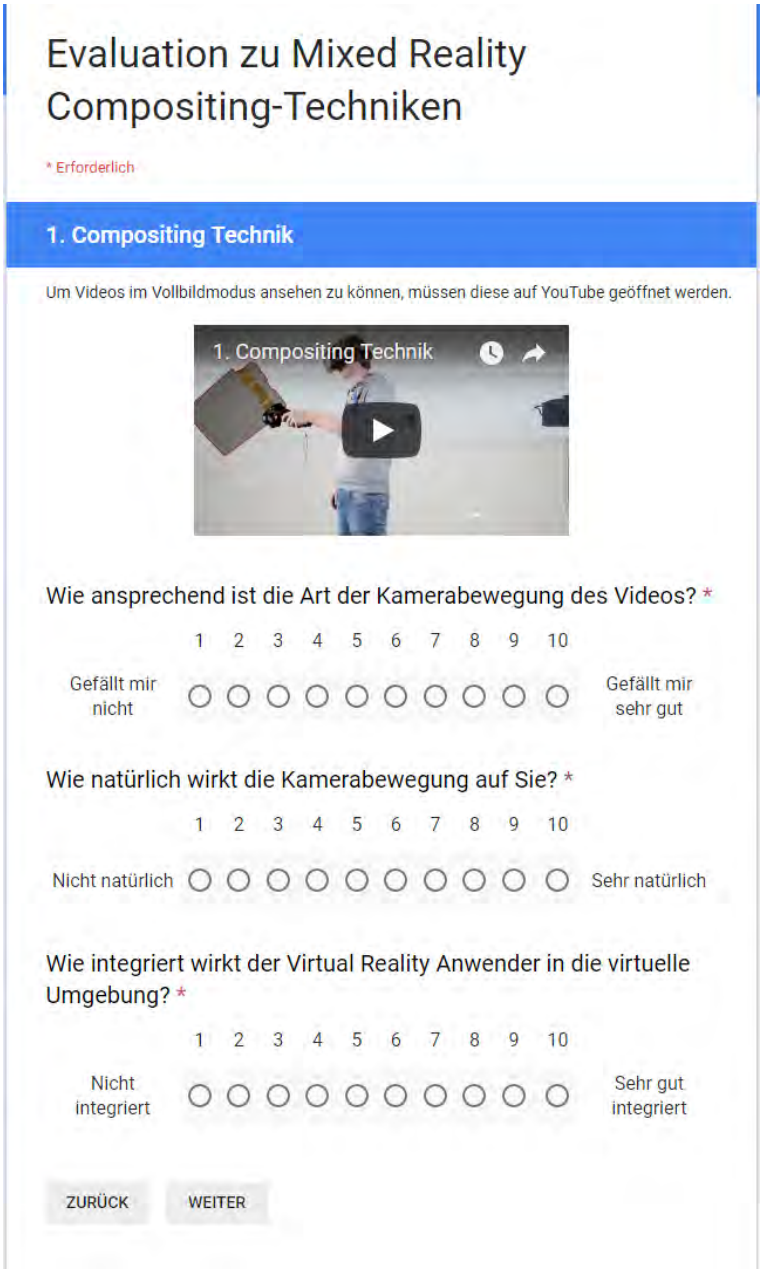

Abbildung B.1: Die erste Seite des Evaluations-Formulars

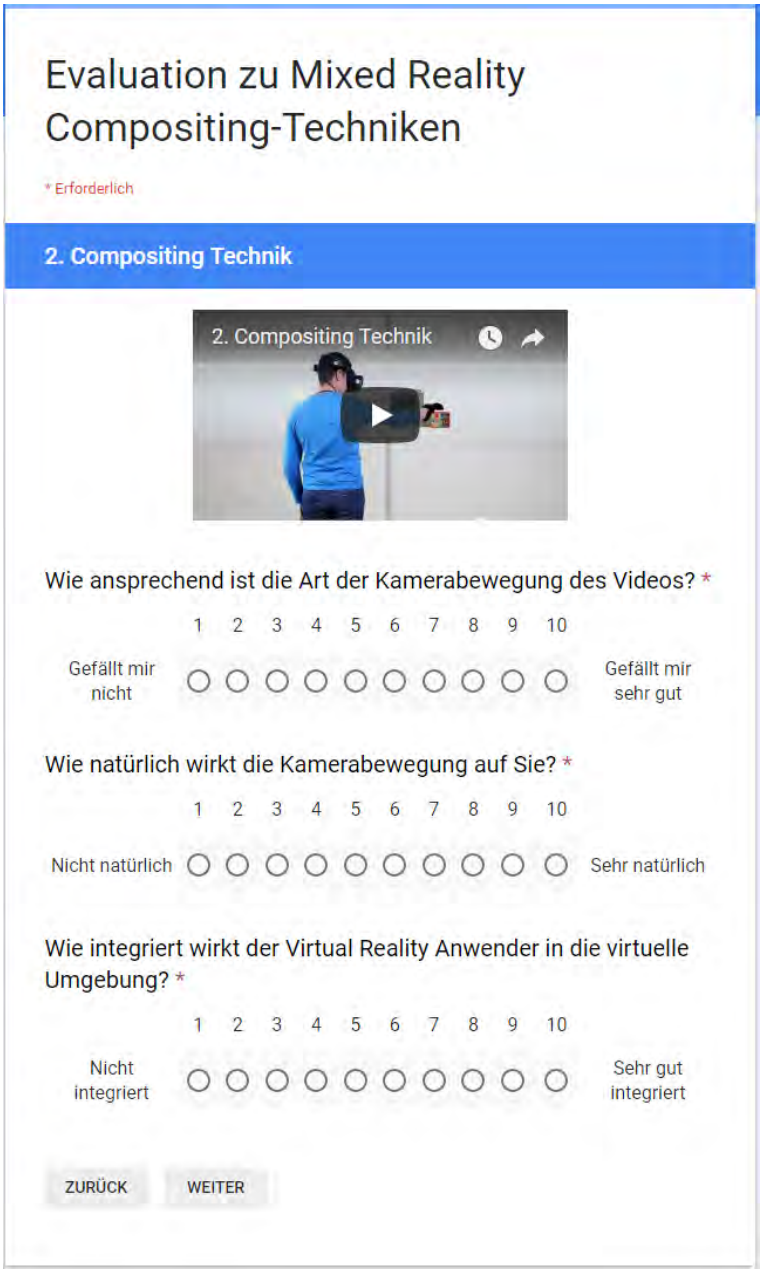

Abbildung B.2: Die zweite Seite des Evaluations-Formulars

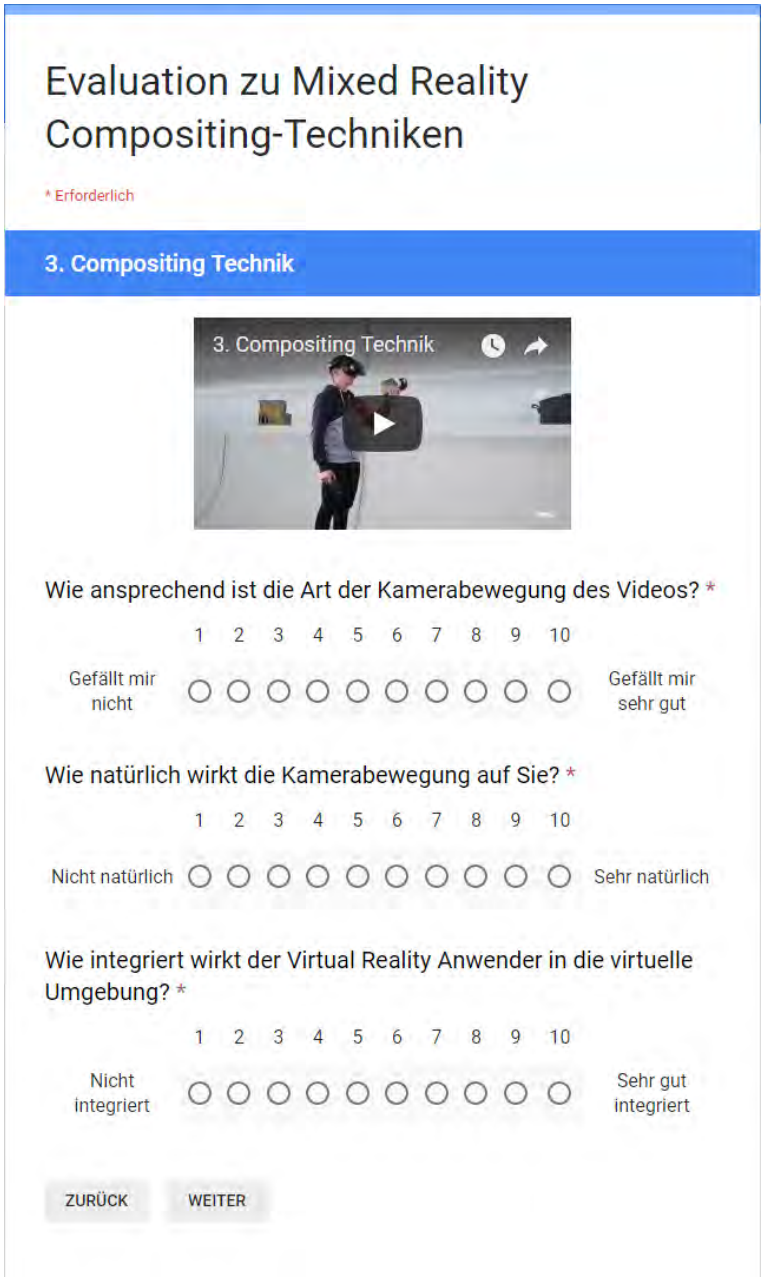

Abbildung B.3: Die dritte Seite des Evaluations-Formulars

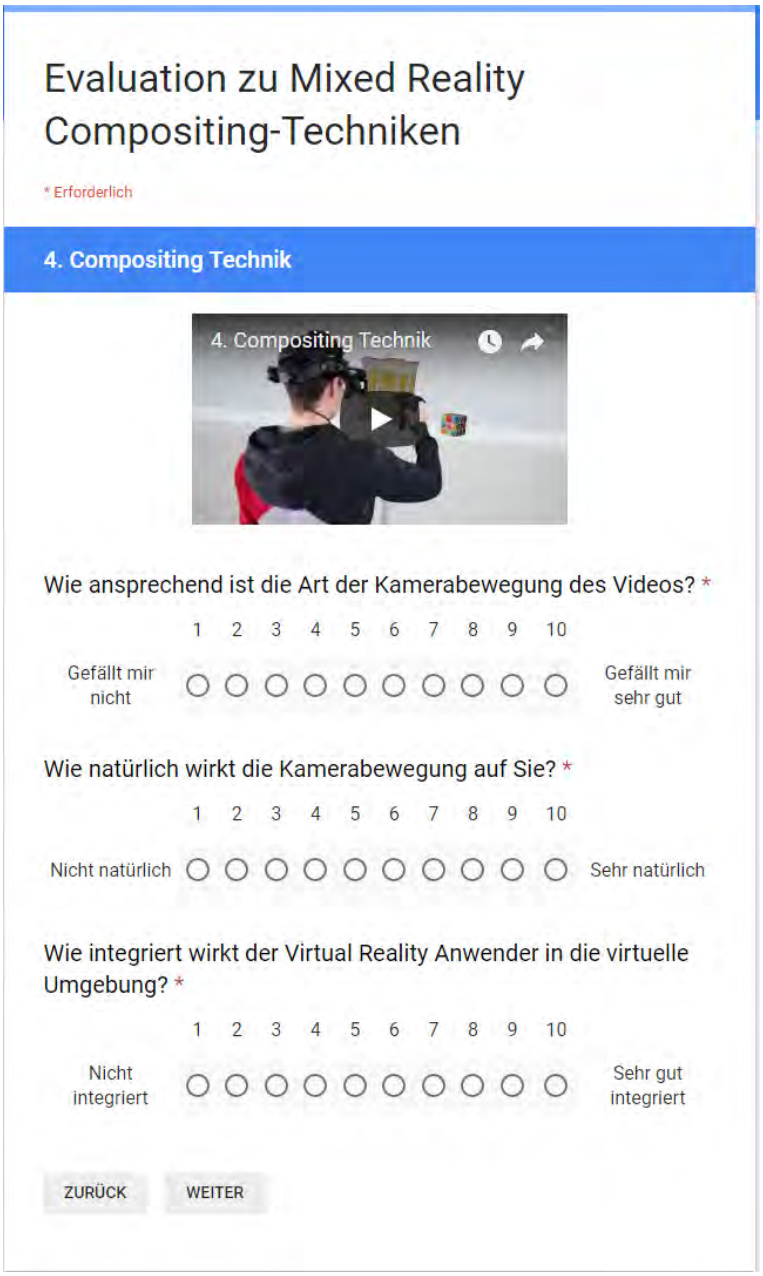

Abbildung B.4: Die vierte Seite des Evaluations-Formulars

Evaluation zu Mixed Reality Compositing-Techniken

* Erforderlich

Abschließende Fragen

Ordnen sie die Kamerabewegungen in eine Rangfolge, abhänging davon welche ihnen am besten gefiel. *

	Am besten	Am zweit besten	Am dritt besten	Am schlechtesten
1. Compositing Technik	○	○	○	○
2. Compositing Technik	○	○	○	○
3. Compositing Technik	○	○	○	○
4. Compositing Technik	○	○	○	○

Welche Kamerabewegung hilft am meisten dabei, die VR-Interaktionen des Anwenders zu verstehen? *

- ○ 1. Compositing Technik
- ○ 2. Compositing Technik
- ○ 3. Compositing Technik
- ○ 4. Compositing Technik

Abbildung B.5: Die fünfte und letzte Seite des Evaluations-Formulars

Anhang C

Evaluationsergebnisse

Tabelle C.1: Bewertung der vier Mixed Reality-Compsiting-Techniken: Angegeben werden die Anzahl an Stimmen für jede mögliche Punktzahl

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1. Compositing-Technik: Statische Kamera										
Wie ansprechend ...	0	0	2	1	1	0	3	4	0	1
Wie natürlich ...	1	0	0	0	5	1	1	3	0	1
Wie integriert ...	0	0	0	1	0	1	2	2	4	2
2. Compositing-Technik: Festgelegte Kamerabewegung										
Wie ansprechend ...	1	1	0	2	0	3	1	2	1	1
Wie natürlich ...	2	1	0	1	3	0	2	1	1	1
Wie integriert ...	0	0	2	4	2	1	0	2	2	2
3. Compositing-Technik: Rekonstruktion der Kamerabewegung										
Wie ansprechend ...	0	0	2	2	0	3	1	2	1	1
Wie natürlich ...	0	1	0	0	2	1	3	2	3	0
Wie integriert ...	0	0	0	2	4	1	1	3	0	1
4. Compositing-Technik: Kameratracking mit VR-Controllern										
Wie ansprechend ...	0	0	1	0	1	5	1	1	2	1
Wie natürlich ...	0	0	0	1	1	1	2	3	4	0
Wie integriert ...	0	0	0	0	0	2	1	5	3	1

Tabelle C.2: Einordnen der Compositing-Techniken in eine Rangfolge: Gezeigt werden die Stimmen pro Platzierung für jede Compositing-Technik

	1.	**2.**	**3.**	**4.**
1. Compositing-Technik: Statische Kamera	4	2	3	3
2. Compositing-Technik: Festgelegte Kamerabewegung	0	2	4	6
3. Compositing-Technik: Rekonstruktion der Kamerabewegung	2	4	4	2
4. Compositing-Technik: Kameratracking mit VR-Controllern	6	4	1	1

Tabelle C.3: „Welche Kamerabewegung hilft am meisten dabei, die VR-Interaktionen des Anwenders zu verstehen?“ Gezeigt werden die Stimmen für jede Compositing-Technik.

1. Compositing-Technik: Statische Kamera	**2. Compositing-Technik:** Festgelegte Kamerabewegung	**3. Compositing-Technik:** Rekonstruktion der Kamerabewegung	**4. Compositing-Technik:** Kameratracking mit VR-Controllern
4	1	2	5

Glossar

Actor Ein Actor ist in der *Unreal Engine* jedes Objekt, das in einem Level platziert werden kann. Actors unterstützen 3D Transformationen

Blueprint Blueprints ermöglichen es Entwicklern in der *Unreal Engine* Programmlogik existierenden Gameplay Klassen mit Hilfe des *Blueprint Visual Scripting* hinzuzufügen. Gameplay Klassen sind alle Klassen die bei Laufzeit der Anwendung für den Anwender relevant sind und manipuliert werden können. Beim erstellen einer Blueprint muss eine Elternklasse ausgewählt werden, auf der die Blueprint basiert. Die Elternklassen sind Actor, Pawn, Character, PlayerController und Game Mode.

Character Ist ein Pawn, der in der Lage ist zu gehen, zu laufen, zu springen und weiteres.

Component Eine Component (zu deutsch Komponente) ist ein Teil eines Actors in der *Unreal Engine*, der eine bestimmte Art von Objekt beinhaltet. Das Objekt kann selbst ein Actor sein.

Game Mode Definiert was in der *Unreal Engine* für ein Spiel gespielt wird und welche Regeln gelten. Dazu zählt zum Beispiel auch welcher Pawn standardmäßig vom Anwender/Spieler kontrolliert wird.

Pawn Ein Pawn ist ein Actor, über den der Anwender die Kontrolle übernehmen und ihn steuern kann.

Rendern ist in der Computergrafik der Prozess der Bilderzeugung anhand von Rohdaten, die das Bild geometrisch beschreiben.

Replay Replays oder auch häufig Demos genannt, sind in Videospielen und deren Engines Aufnahmen des Spielgeschehens. Aber nicht Form eines Videos sonder als Aufzeichnung der gesamten Aktion im 3D-Raum des Spiels.

PlayerController Ist ein Actor der *Unreal Engine* der Eingaben des Anwenders an Pawns weiterleitet, damit der Anwender diese steuern kann.

Timelines Eine Timeline ist eine verfügbare Node im *Blueprint Visual Scripting* der *Unreal Engine*. Eine Timeline erlaubt es, über einen festgelegten Zeitraum Zahlen, Vektoren oder Farben zu verändern und auszugeben oder Events zu vorbestimmten Zeitpunkten auszulösen. Dies geschieht durch das erstellen verschiedener Arten von Tracks in der Timeline und das setzen von Keys in diesen.

Tracking Prozess zur Positionsbestimmung eines Objektes in Bewegung.

VR Kurzform für Virtual Reality, zu deutsch virtuelle Realität.

Literaturverzeichnis

[Asa18] Asa Bailey ; On-set Facilities (Hrsg.): *On-set Facilities Real-time VFX Compositing Set-ups. - On Set Facilities*. http://www.onsetfacilities.com/on-set-facilities/real-time-compositing-and-vfx/. Version: 2018

[Ben13] Bendel, Oliver: *Virtuelle Realität*. http://wirtschaftslexikon.gabler.de/Archiv/-2045879784/virtuelle-realitaet-v1.html. Version: 2013

[DBG13] Dörner, Ralf ; Broll, Wolfgang ; Grimm, Paul: *Virtual und Augmented Reality (VR / AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. Berlin Heidelberg : Springer Berlin Heidelberg, 2013 (eXamen.press). http://dx.doi.org/10.1007/978-3-642-28903-3. – ISBN 978–3–642–28903–3

[Deu16] Deutsche Bank ; eMarketer (Hrsg.): *Prognose zur Anzahl der Virtual-Reality-Nutzer weltweit von 2016 bis 2020*. https://de.statista.com/statistik/daten/studie/426237/umfrage/prognose-zur-anzahl-der-aktiven-virtual-reality-nutzer-weltweit/. Version: 2016

[DFB16] Deloitte ; Fraunhofer ; Bitkom: *Head Mounted Displays in deutschen Unternehmen: Ein Virtual, Augmented und Mixed Reality Check*. https://www2.deloitte.com/de/de/pages/technology-media-and-telecommunications/articles/head-mounted-displays-in-deutschen-unternehmen.html. Version: 2016

[Kre16] Kreylos, Oliver: *Lighthouse tracking examined*. http://doc-ok.org/?p=1478. Version: 2016

[MK94] MILGRAM, Paul ; KISHINO, Fumio: A Taxonomy of Mixed Reality Visual Displays. In: *IEICE TRANSACTIONS on Information and Systems* E77-D (1994), Nr. 12, S. 1321–1329. – ISSN 0916–8532

[NLL17] Niehorster, Diederick C. ; Li, Li ; Lappe, Markus: The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research. In: *i-Perception* 8 (2017), Nr. 3, S. 2041669517708205. http://dx.doi.org/10.1177/2041669517708205. – DOI 10.1177/2041669517708205. – ISSN 2041–6695

[Ocu17] Oculus VR: *System Configurations*. https://support.oculus.com/guides/rift/latest/concepts/mr-rigs/. Version: 2017

[TBM17] Tyrrell, John ; Bancroft, Joschua ; M., Gerald ; Intel (Hrsg.): *Sharing VR Through Green Screen Mixed Reality Video*. https://software.intel.com/en-us/articles/sharing-vr-through-green-screen-mixed-reality-video. Version: 2017