Metahumans in der Unreal Engine für Multiuser-VR-Anwendungen

Bachelorarbeit

zur Erlangung des akademischen Grades eines

Bachelor of Science (B.Sc) Informatik

Angefertigt am Institut für Informatik der Johann Wolfgang Goethe-Universität, Frankfurt am Main

Author: Tobias Völkner
Matrikelnummer: 5328855

Betreuer:
Prof. Dr. Alexander Mehler

3. Februar 2024

Eidesstattliche Erklärung

Ich versichere, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, wurden in jedem einzelnen Fall als Entlehnung kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Frankfurt am Main, den 18. Juli 2023

Tobias Völkner

Danksagung

Mein herzlichster Dank geht an Herrn Prof. Dr. Alexander Mehler und Alexander Henlein aus, sowohl für die Betreuung als auch für die technische Unterstützung.
Ebenso danke ich Herrn Angelo Gebauer und Benjamin Marchal der Inosoft AG Marburg, die mir mit der Unreal Engine eine große Hilfe im Bereich der Animationen und der Metahumans waren.

Zusammenfassung

Metahumans ist ein innovatives Framework für die Unreal Engine, das hochgradig realistische digitale Charaktere zur Verfügung stellt. Metahumans zeichnen sich durch eine vollständige Control Rig aus, die es Entwicklern ermöglicht, vorgefertigte Animationen zu nutzen und sie nach Bedarf anzupassen und zu erweitern.

Im Rahmen dieser wissenschaftlichen Arbeit wird die Anwendung von Metahumans in der virtuellen Umgebung der Unreal Engine 5 untersucht. Das Hauptziel besteht darin, die Fähigkeit eines Metahumans zu untersuchen, mittels eines herkömmlichen Virtual Reality Headsets mithilfe von Motion Tracking gesteuert und animiert zu werden. Dabei wird speziell auf die Verwendung von Inverse Kinematics als Methode zur Erzeugung möglichst natürlicher Bewegungsabläufe eingegangen. Zusätzlich wird angestrebt, die Interaktion zwischen verschiedenen Metahuman-Avataren in einer Online-Sitzung zu ermöglichen.

Um den Einfluss auf das Immersionserlebnis der Benutzerinnen und Benutzer zu analysieren, werden Probandinnen und Probanden eingeladen, ihre Nutzererfahrungen zu evaluieren. Zu diesem Zweck werden zwei vergleichbare Level erstellt: eines in der Unreal Engine mit Metahumans und das andere in Unity mit den Meta Avataren von Oculus.

Diese wissenschaftliche Untersuchung zielt darauf ab, ein umfassendes Verständnis für die Leistungsfähigkeit von Metahumans zu erlangen, insbesondere im Vergleich zu anderen Avatar-Systemen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Aufgabenstellung

Ziel dieser Arbeit ist die Implementierung und Dokumentation einer Multi-User Virtual Reality Umgebung mit dem Metahuman Framework der Unreal Engine. Das VR-Avatar des Benutzers soll dabei ein Metahuman sein, dessen Bewegungen durch die Motioncontroller und das Headset des jeweiligen Virtual Reality Gerätes synchronisiert werden.

Die Avatare sollen dann in eine Multi-User-Umgebung geladen werden, um dort Aufgaben zu bearbeiten. Die Avatare erhalten die grundlegende Funktionen eines Unreal Engine VR Pawns [1], um mit der Umgebung und anderen Avataren zu interagieren.

Für diese Anwendung wird eine Evaluation der User Experience durchgeführt. Ziel dieser Evaluation ist es, den Grad der Immersion [2] bei der Verwendung eines Metahuman VR-Avatars zu bestimmen. Zum Vergleich werden [3]-Meta-Avatare von Occulus herangezogen.

1.2 Motivation

Metahumans sind der Versuch von Unreal, Menschen digital so realitätsnah wie möglich nachzubilden. Die Implementierung dieser Metahumans in eine virtuelle Umgebung bietet ein großes Potential für neue Erkenntnisse im Bereich der Immersion von Virtual Reality Nutzern. Die Motivation dieser Bachelorarbeit ist es, zu testen, inwieweit eine realistische Implementierung von menschlichen Avataren die User Experience beeinflussen kann. Anhand von ultrarealistischen Metahumans kann evaluiert werden, inwiefern die Immersion eines Benutzers durch die Nachbildung eines menschlichen Gegenübers und die Interaktion innerhalb einer virtuellen Umgebung gesteigert werden kann. Dabei wird evaluiert, wie sich die Nutzer in der Interaktion mit anderen Avataren erleben und ob sich die User Experience verändert.

Diese Erkenntnisse können Aufschluss darüber geben, welche Faktoren wichtig sind, um die User Experience zu intensivieren und emotionale Erlebnisse hervorzurufen.

Ebenso wichtig ist es, die Performance und Usability neuer Technologien wie Unreal Engine 5 (veröffentlicht: April 2022)[4] und Metahumans (in Entwicklung) zu testen.

Sowohl die Erkenntnisse aus dieser Arbeit als auch das entwickelte Projekt können als Assets in zukünftige VR-Projekte des TTLabs unter der Leitung von Prof. Dr. Alexander Mehler einfließen[5]. Speziell für das Va.Si.Li-Lab könnten Metahumans als alternative User-Avatare eingesetzt werden.

2 Stand der Wissenschaft und Technik

2.1 Virtual Reality

Virtual Reality (VR) ist eine Technologie, die es Benutzer/-innen ermöglicht, in eine möglichst immersive Umgebung einzutauchen[6]. Mit Hilfe von VR-Headsets, Motioncontroller und Motion-Trackern können sich Benutzer/-innen in einer virtuellen Welt bewegen, die ihnen das Gefühl geben kann, physisch präsent zu sein.

VR bietet eine Vielzahl von Anwendungsmöglichkeiten in so unterschiedlichen Bereichen wie Spieleentwicklung, Simulation, Bildung, Medizin und Architektur. Sie bringt neue Möglichkeiten für das Training in simulierten Umgebungen, die Erkundung virtueller Räume und der Entwicklung von Unterhaltungserlebnisse. Die Unreal Engine spielt eine wichtige Rolle bei der Darstellung der virtuellen Umgebung in hoher Qualität und mit geringer Latenz.

Die Entwicklung von VR-Technologien schreitet stetig voran, wodurch die Qualität, die Immersion und die Benutzerfreundlichkeit kontinuierlich verbessert werden. Mit dem Aufkommen von leistungsstärkerer Hardware, drahtlosen Headsets und der Integration von VR in mobile Geräte wird VR zunehmend zugänglicher und verbreiteter.

Insgesamt bietet Virtual Reality ein faszinierendes Erlebnis, das Benutzern ermöglicht, in eine computergenerierte Welt einzutauchen und diese zu erkunden. Die Technologie hat das Potenzial, viele Bereiche des menschlichen Lebens zu transformieren und neue Möglichkeiten für Interaktion, Unterhaltung und Lernen zu schaffen.

2.2 Unreal Engine

Die Unreal Engine ist eine leistungsstarke Spiel-Engine, die von Epic Games entwickelt wurde. Sie ermöglicht die Erstellung 3Dimensionalen Räumen für verschiedene Plattformen und bietet fortschrittliche Grafiken phsyische Simulationen. Die Engine wird von einer großen Community unterstützt und ist sowohl bei professionellen Entwicklern als auch bei Indie-Studios beliebt[7].

Die aktuelle Version ist Unreal Engine 5.2.1.

2.2.1 Graph Editor

Der Graph Editor der Unreal Engine ist ein Werkzeug, das Entwicklern eine visuelle Programmierumgebung bietet[8]. Es ist die Basis des Blueprint-Systems der Unreal Engine.

Der Graph Editor basiert auf einem Netzwerk von Knoten, in dem verschiedene Knoten miteinander verbunden werden, um logische Funktionen und Prozesse zu erstellen. Jeder Knoten repräsentiert eine bestimmte Aktion oder Operation, wie das Auslösen von Ereignissen, das Ändern von Variablenwerten oder das Ausführen von Code. Ausschlaggebend für die Ausführungsreihenfolge sind die *Execution pins*[9], durch die Funktionsaufrufe miteinander verbunden

werden. Dies ist vergleichbar mit der Zeilen-Reihenfolge in der herkömmlichen Programmierung. Für ein Beispiel praktischer Umsetzung siehe Kapitel 4.5.1.

2.2.2 VR-Pawn

Der VR-Pawn in der Unreal Engine ist ein spezieller Blueprint, der als Grundlage für die Erstellung von Virtual Reality (VR)-Interaktionen dient. Der VR-Pawn repräsentiert einen virtuellen Charakter, mit dem der Benutzer in der VR-Umgebung interagieren kann[1].

Der VR-Pawn enthält standardmäßig Komponenten und Funktionen, die für die Realisierung von VR-Interaktionen notwendig sind. Dazu gehören beispielsweise Tracking-Komponenten für die Position und Rotation des VR-Headsets sowie die Steuerung von Handbewegungen und Eingabegeräten wie VR-Controllern. Der VR-Pawn bildet die Grundlage für die Entwicklung von VR-Anwendungen und Spielen in der Unreal Engine.

Der VR-Pawn ist in der Unreal Engine der Blueprint, der als Basis für die Implementierung von VR-Interaktionen dient. Es ermöglicht Entwicklern, Benutzereingaben in der VR-Umgebung zu erfassen und daraus folgende Funktionen zu implementieren.

2.3 Metahuman

Metahuman ist ein Unreal Engine Framework, das es dem Benutzer ermöglicht, einen fotorealistischen digitalen Menschen zu verwenden.

2.3.1 Metahuman Creator

Unreal Engine bietet den Metahuman Creator an, mit dem eigene 'digitale Menschen' erstellt werden können. Der Metahuman Creator [10] ist ein Cloud-basiertes Tool, mit dem eigene Avatare erstellt werden.

Derzeit gibt es 66 Basis-Metahumans mit verschiedenen Variationen von Geschlecht, Alter, Größe, Hautfarbe, Haarfarbe etc. Der Creator bietet die Möglichkeit, eigene Metahumans auf Basis dieser Vorlagen zu erstellen. Behaarung, Körperform und Augenfarbe können ausgewählt werden. Das Gesicht kann verändert werden, indem Gesichtspartien von anderen Metahumans übernommen werden. Die Gesichtszüge basieren derzeit nur auf den Voreinstellungen der Metahumans. Eigene Gesichtszüge können nicht erstellt werden.

Die Abbildung 1 zeigt den Metahuman Creator. Auf der linken Seite befindet sich das Auswahlrad der Metahumans, deren Gesichtszüge zur Veränderung des aktuellen Metahumans verwendent werden können. In der Mitte der Auswahl befindet sich der Basis-Metahuman, der gerade bearbeitet wird. Weitere Anpassungsmöglichkeiten zur Veränderung des Metahuman werden in der Navigation angezeigt.

2.3.2 Metahuman in Unreal Engine

Eine Metahuman-Datei in Unreal Engine besteht aus einer Blueprint-Datei für den gesamten Metahuman und den zugehörigen Asset-Ordnern, die die

Abbildung 1: Metahuman Creator

Skeletal Meshes und Materialien enthalten, aus denen der Metahuman besteht. Ein einzelner Metahuman hat eine Download-Größe von 800-1000 Megabyte. Das Framework selbst hat eine Größe von 3 Gigabyte.Das Framework stellt das Metahuman Base Skeleton und zwei Control Rigs zur Verfügung. Animationen sind typischerweise mit viel Arbeit verbunden und die Control Rigs dienen dazu, einen Großteil der Arbeit vorzuprogrammieren. Bei den Metahumans gibt es ein Control Rig für den Körper und ein Control Rig für das Gesicht.

Das Hauptaugenmerk dieser Arbeit liegt auf den Bewegungen des Metahuman-Avatars mit den Motion Controllern der VR-Brille. Gestik und Mimik des Metahumans werden in diesem Projekt ausgelassen.

2.3.3 Skeletal Mesh

Ein Skeletal Mesh [11] besteht aus einem digitalen Skelett und einem Mesh[1] Animationsfunktionalität.

Ein digitales Skelett ist mit dem Oberflächenmaterial verbunden und steuert die Bewegungen.

Das Metahuman Base_Skeleton ist im Vergleich zu anderen Humanoid Rigs[2] wesentlich umfangreicher.

Im Vergleich hat die SK_Mannequinn der Unreal Engine 5 161 Knochen. Das BaseSkeleton eines Metahuman hat insgesamt 354 Knochen, davon 107 einer einzelnen Hand 107[3]. Hinzu kommen die Knochen, die in der Gesicht_Mesh enthalten sind. Das Gesicht hat insgesamt 875 Knochen, durch die die Mimik erzeugt wird. Alle Animationen eines Metahuman Avatars werden mit Hilfe des Base_Skeletons ausgeführt. Für diese Animationen bietet das Metahuman-

[1] Fasst die Oberflächeninformationen von Objekten typischerweise in Form von Polygonen zusammen
[2] Ein Skeletal Mesh mit menschliche Anatomie als Grundlage
[3] direkt auslesbar aus der Metahuman Blueprint

Abbildung 2: Metahuman Base_Skeleton

Framework eine eigene Control Rig.

2.3.4 Control Rig

Control Rigs sind ein leistungsstarkes Werkzeug der Unreal Engine zur Erstellung und Steuerung von Animationen[12]. Sie ermöglichen die Steuerung komplexer Bewegungen, Verformungen und Interaktionen von Charakteren und Objekten.

Control Rigs basieren auf einer hierarchischen Struktur von Steuerelementen, die mit den entsprechenden Teilen des Modells oder der Szene verknüpft sind. Diese Steuerelemente können verwendet werden, um Parameter wie Gelenkwinkel, Positionen, Rotationen und Skalierungen zu manipulieren und so die gewünschten Animationen zu erzeugen.

Ein wesentliches Merkmal von Control Rigs ist die Möglichkeit, Animationen in Echtzeit zu erzeugen und anzupassen. Durch die direkte Interaktion mit den Steuerelementen können Animatoren die Animationen intuitiv anpassen und verfeinern, um die gewünschten Ergebnisse zu erzielen.

Control Rigs bieten auch Funktionen zur Automatisierung und Vereinfachung von Animationen. Vordefinierte Aktionen und Bewegungsmuster können erstellt und wiederverwendet werden, um komplexe Animationen schneller und effizienter zu erstellen. Dies erhöht die Produktivität und ermöglicht es Animatoren, sich auf die künstlerischen Aspekte der Animation zu konzentrieren.

2.3.5 Groom

Die Metahuman Groom Assets in der Unreal Engine sind vordefinierte Haar- und Bartstile, die für die Verwendung mit Metahuman Charakteren entwickelt wurden[13]. Diese Groom Assets dienen als Ausgangspunkt für die Gestaltung und Anpassung der Haare und Bärte der Metahuman-Charaktere. Die Groom Assets enthalten bereits eine vorgefertigte Frisur oder Bartform, die mit den entsprechenden Haar- und Felleigenschaften wie Länge, Dichte, Farbe und Styling parametrisiert ist.

Mit den Groom Assets können verschiedene Parameter der Haare und Bärte angepasst werden, um das gewünschte Aussehen zu erzielen. Dazu gehört beispielsweise das Kämmen, Formen, Hinzufügen oder Entfernen von Haaren, Anpassen der Textur oder Einfärben der Haare.

2.3.6 LODSync

LODSync ist eine Technologie zur Synchronisierung von Daten in der Unreal Engine. Sie ermöglicht es, LOD-Daten (Level of Detail) effizient zu aktualisieren und zu verwalten[14].

In der Unreal Engine werden LODs verwendet, um die Detailstufen von 3D-Modellen anzupassen, je nachdem, wie weit sie vom Betrachter entfernt sind. Dies trägt zur Leistungsoptimierung in Echtzeit-3D-Anwendungen bei. Mit LODSync können Entwickler die LOD-Daten ihrer Modelle einfach synchronisieren, um sicherzustellen, dass die verschiedenen Detailstufen korrekt angezeigt werden.

2.4 Inverse Kinematics

Inverse Kinematik (IK) ist ein Konzept, das in der Animation verwendet wird, um realistische und natürliche Bewegungen von Figuren oder Kreaturen zu erzeugen. Es ermöglicht die Steuerung der Gelenke eines animierten Modells in Abhängigkeit von der gewünschten Position oder Ausrichtung des Endeffektors, z.B. der Hand oder des Fußes.[15]

Anstatt die Gelenkwinkel einzeln zu animieren, erlaubt die Inverse Kinematik dem Animator, die gewünschte Position oder Orientierung eines Endeffektors zu definieren, und das System berechnet automatisch die entsprechenden Gelenkwinkel, um diese Position zu erreichen. Dies erleichtert die Animation komplexer Bewegungen, da sich der Animator auf das gewünschte Ergebnis konzentrieren kann, anstatt jede einzelne Gelenkbewegung manuell zu steuern.

Inverse Kinematik wird häufig verwendet, um natürliche Bewegungen wie Gehen, Greifen oder Klettern zu animieren. Sie ermöglicht es Figuren, Hindernissen auszuweichen, Gegenstände zu greifen oder sich auf unebenem Gelände zu bewegen, indem die Gelenkwinkel automatisch berechnet werden, um die gewünschten Positionen und Orientierungen beizubehalten.

Die Berechnung der inversen Kinematik in der Animation kann auf verschiedene Weise erfolgen, abhängig von der Komplexität des Modells und der gewünschten Bewegung.

3 Technologische Grundlagen

3.1 Hardware

Für die Entwicklung und den Test der VR-Umgebung wurde Meta Quest 2 verwendet. Um damit testen zu können, muss die Oculus-Anwendung heruntergeladen und über einen Questlink mit dem PC verbunden werden.

Für die Evaluation wurde die Meta Quest 3 verwendet.

3.2 Wahl der Unreal Engine Version

Die Wahl der Unreal Engine Version hat sich im Laufe des Projekts geändert. Ursprünglich war geplant, Metahumans ausschließlich für die Unreal Engine 5 zu entwickeln. Daher wurden die ersten Versionen der Implementierung von Metahumans in Unreal in der Unreal Engine 5.1 realisiert. Nachteile dieser Implementierung waren jedoch die Performance. Da Metahumans ohnehin schon hohe Anforderungen an die Performance stellen, ist es wichtig, den Ressourcenverbrauch an anderen Stellen zu reduzieren. Insbesondere eigenständige Anwendungen für VR-Brillen werden deutliche Einschränkungen bei der Bildübertragungsrate und der allgemeinen Performance der Anwendung haben. Die Unreal Engine 5 hat einen höhere Grafikanspruch und baut detailliertere Schatten und Beleuchtung in die Level ein. Daher war die Performance von Anfang an eingeschränkt. Das größere Problem war jedoch die Suche nach Quellen und Tutorials für die Unreal Engine. Da es sich bei der Unreal Engine 5 um eine neue Technologie handelt, gibt es derzeit nur wenige Informationen über diese Engine. Auf der Suche nach guten Quellen für Animationen und verschiedene andere Funktionen fanden sich oft nur Material für die Unreal Engine 4. Die Portierung dieser Quellen auf die Unreal Engine 5 war zu diesem Zeitpunkt noch sehr unsicher. Außerdem wurde im Laufe der Recherchen herausgefunden, dass Metahumans auch für die neueste Version der Unreal Engine 4 (4.27) implementiert ist, weshalb die Entscheidung fiel, das Projekt in Unreal Engine 4.27 zu schreiben.

Im weiteren Verlauf des Projekts wurde jedoch deutlich, dass das Metahuman Framework in der technischen Umsetzung wesentlich besser mit der Unreal Engine 5 interagiert. Das Hauptproblem sind hier die Knochen des Skeletal Meshes der Metahumans und des Sk Mannequin. Die Skelettstruktur der Unreal Engine 5 Mannequin passt wesentlich besser zur Skelettstruktur der Metahumans, da die Skelette der Unreal Engine 4.27 wesentlich weniger Knochen haben. Dies wird vor allem dann relevant, wenn Animationen auf die Metahumans übertragen werden sollen. Dieser Vorgang wird auch Animation Retargeting genannt. Außerdem werden die Metahumans auch in Zukunft nur für die Unreal Engine 5+ weiterentwickelt und erweitert. Der Download der Metahumans für die Unreal Engine 4.27 war zudem umständlich. Zuerst musste die Quixel Bridge als eigenständige App heruntergeladen werden, dann musste die App mit dem aktuellen Projekt verbunden werden, um dann die Metahumans zu übertragen. Die Metahumans sind weniger detailliert und der Download selbst funktionierte am Ende der Entwicklung in Unreal 4.27 nicht

mehr. Das Projekt hatte zu diesem Zeitpunkt sehr viele Laufzeit-Probleme, weshalb das Projekt vor der Evaluation in der Unreal Engine 5.2.1 neu aufgebaut wurde.

3.2.1 Pro und Contra

Es war wichtig, das Projekt in Unreal Engine 4.27 zu schreiben, um die grundlegende Logik des Projekts zu verstehen und verschiedene Ressourcen auszuprobieren, die nur in Unreal Engine 4+ Versionen zur Verfügung stehen. So standen einige Animation Blueprints bereit, die sehr einfach auf die Metahumans übertragen werden konnten und kostenlos auf dem Marketplace zur Verfügung standen. Diese Animation Blueprints sind für die Unreal Engine 5+ in dieser Form nicht mehr verfügbar. Für ähnliche Animationen müsste man auf kostenpflichtige Produkte zurückgreifen.

Beide Versionen haben ihre individuellen Vor- und Nachteile, auch wenn die Unreal Engine 5 deutlich bessere Shader[16] zur Verfügung stellt und neue Features implementiert wurden, sind die Performanceanforderungen sowohl in der Entwicklung als auch in der Anwedung sehr hoch. Die Unreal Engine 4.27 ist derzeit noch die deutlich stabilere Version, so kam es bei den Unreal Engine 5+ Versionen immer wieder zu Abstürzen. Im Gegensatz dazu hatte die Version 4.27 während der gesamten Projektentwicklung kein einziges Mal Performanceprobleme.

3.2.2 Fazit

Um neue Features zu testen, ist die Unreal Engine 4.27 die bessere und stabilere Variante. das Metahuman-Framework wird jedoch ständig weiterentwickelt [17] und um das Projekt für zukünftige Arbeiten verfügbar zu halten, ist es sinnvoll, dies in Unreal Engine 5+ zu schreiben. Die grundlegenden Funktionen unterscheiden sich in den beiden Game Engines kaum.

3.3 Frameworks und Plugins

3.3.1 Quixel Bridge

Quixel Bridge[18] ist eine Verbindung zur Unreal Engine, um externe Ressourcen zu importieren und zu verwenden. Die Quixel Content Bibliothek wird als Metascans bezeichnet. Über die Quixel Bridge können Metascans in ein Projekt importiert werden.

Ab der Unreal Engine 5.0 ist die Quixel Bridge in den Editor integriert. Für die Unreal Engine 4.27 muss die Bridge heruntergeladen und mit dem Projekt verbunden werden. Für die Unreal Engine 5 und 4.27 werden unterschiedliche Metahumans verwendet.

Quixel Bridge dient als Bibliothek für Inhalte, die entweder von Privatpersonen, Drittanbietern oder Unreal selbst angeboten werden. Auf dem Quixel Bridge Marketplace werden Inhalte entweder kostenlos oder kostenpflichtig angeboten.

3.3.2 Advanced Session Plugin

Das Advanced Session Plugin wird verwendet, um Online-Sitzungen zu erstellen. Das Advanced Session Plugin verwendet Steam als Basis, um eine Verbindung zu einer anderen Sitzung herzustellen. Um dieses Plugin nutzen zu können, muss es in der Engine.ini der Unreal Engine konfiguriert werden. Danach ist es möglich eigene Sessions zu erstellen und zu finden. Es ist dann auch möglich, Sessions über Steam beizutreten. Das Advanced Session Plugin hat während der Entwicklung auf lokaler Ebene gut funktioniert, leider war es bis zum Ende des Projektes nicht möglich eine Session über zwei verschiedene Systeme laufen zu lassen. Die Server konnten zwar erstellt und auch gefunden werden, aber bei der Verbindung kam es ausnahmslow zu Timeout-Fehlern.

3.4 Versionskontrolle

Für die Versionskontrolle wird das Gitlab Repository des Text Technology Lab's verwendet:

https://gitlab.texttechnologylab.org/

Die aktuellste Version der Metahuman App kann unter:

https://gitlab.texttechnologylab.org/TobiasVoelkner/meta-human-vr

gefunden werden. Eine Verbindung zum Netzwerk der Goethe-Universität wird benötigt.

4 Technische Realisierung

Ein Großteil der Arbeit wurde in Unreal Engine 4.27 geschrieben, bevor die endgültige Version der Anwendung auf Unreal Engine 5.2 migriert wurde.

4.1 Import und Test der Metahumans

Zu Beginn wurde eine Testumgebung erstellt, in der die Metahumans getestet werden konnten, um sich mit ihnen vertraut zu machen. Die erste Version wurde in Unreal Engie 5.1 erstellt. Als Grundlage diente ein Third Person Template[19], um die Funktionalitäten eines Metahuman Skeletal Meshes einfacher testen zu können. Nachdem das Projekt erstellt wurde, wurde ein Metahuman namens Chandra mit der Quixel Bridge heruntergeladen.

In der Abbildung 3 ist der Blueprint von Chandra zu sehen. In diesem Blueprint ist der Viewport geöffnet, der die visuelle Darstellung von Chandra zeigt.

Abbildung 3: Chandra Blueprint

Um die Animationen des Metahumans testen zu können, wurde das Skeletal Mesh von Chandra in das Third-Person Blueprint des Third-Person Game Mode der Unreal Engine 5 eingefügt[4][20].

Das Third-Person Blueprint verwendet standardmäßig den SKM_Quinn_Simple, ein Unreal Engine 5 Mannequin. Die Unreal Engine 5 SK_Mannequins sind für das Metahuman Framework relevant, da es sich um ein Humanoid Rig handelt. Animationen, die für Unreal Engine 5 SK_Mannequins entwickelt wurden, können auf Metahumans übertragen werden.

Das Skeletal Mesh, das anstelle des SKM_Quinn_Simple verwendet wird, ist das m_med_nrw_body. Dies ist Chandra's eigenes Skeletal Mesh. Es im-

[4]Basis war ein Video-Tutorial

portiert nur die Gliedmaßen, die dem Metahuman zugeordnet sind. Das sind die Teile, die nicht von der Kleidung verdeckt werden.

Um den Körper zu vervollständigen, müssen die Füße, die Beine und der Torso als Unterkomponenten des Skeletal Mesh definiert werden. Diese Komponenten befinden sich im gemeinsamen Ordner des Metahuman Frameworks und sind nicht Teil des individuellen Metahumans. Für das Gesicht wird das Chandra_FaceMesh verwendet. Zusätzlich müssen Groom-Assets[5] übergeben werden. Damit die Groom-Assets korrekt angezeigt werden, wird die LODSync-Komponente übergeben.

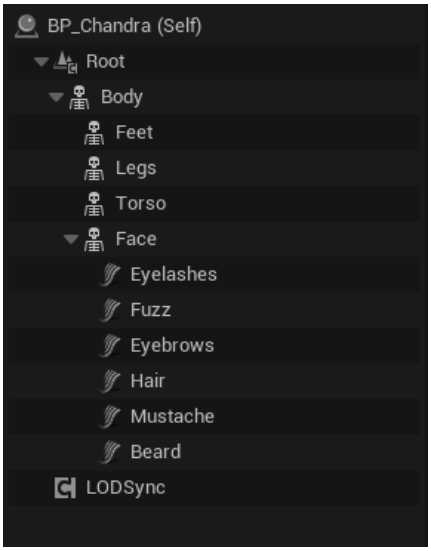

Abbildung 4: Metahuman Blueprint Struktur

Die Struktur des Blueprints ist in der Abbildung 4 dargestellt und muss in dieser Form in den Third Person Blueprint übernommen werden, um die Komponenten korrekt zu verknüpfen.

Um nun die Animationen ablaufen zu lassen, muss das ABP_Quinn_C als Anim Klasse definiert werden. Diese Klasse muss auf alle Komponenten angewendet werden, um eine vollständige Animation des Metahuman zu ermöglichen. Das Gesicht benötigt keine Anim-Klasse, es sei denn, Gestik und Mimik werden benötigt. Der ABP_Quinn_C Animation Blueprint eines Humanoid Rigs kann auf einen Metahuman angewendet werden. Dazu müssen die Skeletal-Mesh Komponenten des Metahumans dem SK_Mannequin zugewiesen werden[6]. Innerhalb des ABP_Quinn_C Animation Blueprint wird die Wurzel des Full Body IK[21] auf animation_root gesetzt. Dies geschieht, um die Animation des Metahuman Skeleton darzustellen.

Der Metahuman kann nun als Third-Person-Avatar verwendet werden. Eine kurze Demonstration der Animation wurde auf Youtube hochgeladen[22].

Das Ergebnis dieses Experiments ist, dass Metahumans ohne großen Aufwand in der Unreal Engine 5 animiert werden können und dass vorentwickelte Animationen des SK-Mannequins auf Metahumans übertragen werden können.

4.2 Einbindung der Metahumans in eine virtuelle Umgebung

Der nächste Schritt zur Erstellung einer funktionierenden VR-Umgebung mit Metahumans ist die Einbindung der Avatare in ein Virtual Reality Projekt der Unreal Engine.

[5] Die Assets für die Gesichtsbehaarung (Frisur, Wimpern, Augenbrauen, Bart, etc.)
[6] Rechtsklick->Skeleton->Assign Skeleton->SK_Mannequin

Es stellt sich die Frage, ob das Rendern der Metahumans in VR Unreal Engine möglich ist. Außerdem muss die Performance des VR Headsets eine gute Framerate erzeugen können. Um dies zu testen werden eine Reihe von Metahumans heruntergeladen, und vorerst ohne Animationen in das Unreal Engine VR-Level gestellt. Eine Unregelmäßigkeit fällt sofort auf, die Groom-Assets der Metahumans werden nicht oder nicht vollständig geladen. Der Grund dafür ist, wie sich später herausstellte, dass speziell beim Start eines Virtual Reality Projektes in Unreal die Groom-Assets nicht korrekt gerendert werden. Die Ursache für diesen Fehler ist vollständig geklärt. Die Abbildung 5

Abbildung 5: Groom-Asset Error

zeigt den Groom-Fehler, bei dem Assets unvollständig/transparent (links und rechts) oder gar nicht (Mitte) geladen werden. Dieser Fehler bleibt durchgehen bestehen für jedes Level → tritt bei jedem Level auf.

Dieser Level wurde aus der virtuellen Umgebung der Unreal Engine 5+ entnommen, aber der Fehler zieht sich durch das gesamte Projekt.

Trotz dieses Fehlers, wurde vorerst das Level mit der Oculus Quest im VR-Preview-Modus der Unreal Engine gestartet. Die Groom-Asset waren nach wie vor nicht korrekt gerendert[7].

4.3 Die Performance der Metahumans in einer virtuellen Umgebung

Bevor die Anwendung mit den Metahuman Skeletal Meshes gestartet wurde, wurde die Standard VR-Umgebung von Unreal getestet. In diesem Fall gab es kaum Verzögerungen in der Bildübertragungsrate, aber die Bildrate war relativ konstant.

Beim ersten Test der Metahumans im VR-Modus traten erhebliche Leistungseinschränkungen auf. Das Rendern des Metahumans beanspruchte fast die gesamte Kapazität der Oculus Quest 2 und beeinflusste somit die gesamte Performance der Brille. Die Bildwiederholrate wurde instabil und das Motion-Tracking der Kamera so stark verzögert, dass bei schnellen Drehungen das Bild schwarz wurde, da die GPU[8] der Occulus Quest 2 überlastet war. Teil-

[7] Online gibt es dafür eine Video Demonstration[23]
[8] Graphical Processing Unit: Grafikprozessor

weise blieb das Bild für mehrere Sekunden stehen.

Sobald das Sichtfeld von den Metahumans weggelenkt wurde, erholte sich die GPU wieder, aber die Bildwiederholrate lag immer noch weit unter der Norm. Es war also klar, dass die GPU entlastet werden musste, wenn man eine Chance haben wollte, die Metahumans in eine virtuelle Umgebung zu bringen.

Um die Performance zu verbessern, müssen Voreinstellungen im Projekt vorgenommen werden[24]. Dazu werden die Lichteinstellungen verändert und die Reflexionen abgeschaltet. Für den Unreal Engine Editor müssen außerdem die Engine Scalability Settings auf die niedrigste Einstellung (Low) gesetzt werden. Dies war bis zum Schluss die einzige Möglichkeit, die GPU des VR-Headsets nicht zu überlasten. Möglicherweise gibt es für ein Endprodukt noch weitere Einstellungen, aber für den Projektverlauf waren diese Einstellungen ausreichend, um die virtuelle Umgebung nutzbar zu machen. Allerdings ist die GPU des VR-Headsets auch mit dem Nachfolgemodell Occulus Quest 3 immer voll ausgelastet. Diese Performanceprobleme waren der ausschlaggebende Grund für den Einsatz der Unreal Engine 4.27 im weiteren Verlauf des Projekts.

Durch die geringere Anzahl an Shadern der Engine und den reduzierten Detailgrad der Metahumans erhoffte man sich eine Verbesserung der Performance für Metahuman in einer virtuellen Umgebung. Im Nachhinein stellte sich jedoch heraus, dass die Leistungssteigerung nur marginal und somit vernachlässigbar war.

4.4 Verwendung der Metahumans als VR-Pawn

Ab diesem Teil des Projekts wurde die Unreal Engine Version 4.27 verwendet. Alle Metahumans werden nun über die Anwendung Quixel Bridge heruntergeladen. Um einen Metahuman als VR-Pawn verwenden zu können, wird ähnlich vorgegangen wie im Kapitel 4.1.

Zu diesem Zweck wurde ein Pawn-Blueprint erstellt. In diesen Blueprint wird eine Scene-Component eingefügt. Diese Komponente dient dazu, einen Raum zu definieren, in dem sich die anderen Komponenten bewegen.

Um eine VR-Umgebung zu erstellen, benötigt man eine Kamera, zwei Motioncontroller (links und rechts) und das SkeletalMesh für den Metahuman. Abbildung 6 zeigt den Aufbau der PawnStructure.

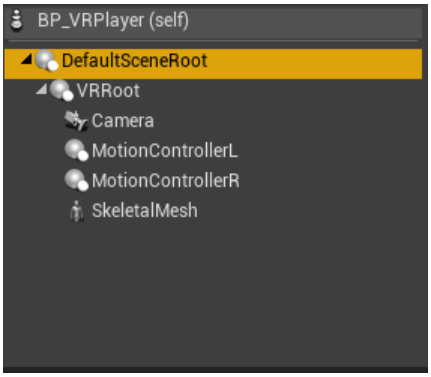

Abbildung 6: Pawn Struktur

Für Testzwecke und die Entwicklung der Animationen wurde das SK_Mannequin der Unreal Engine 4

verwendet, da für die Animationen kein Retargeting notwendig ist. Später wurde das SK_Mannequin durch den Metahuman ersetzt. Für die endgültige Struktur eines Metahuman-Pawn wird unter dem Skeletal Mesh die Struktur aus Abbildung 4 benötigt. Die Komponente Body ist in diesem Fall das Äquivalent zum SkeletalMesh.
Die Platzierung aller Komponenten in der Szene ist zweitrangig, da alle Komponenten dynamisch platziert werden. Nur die Unterkomponenten des Skeletal Mesh müssen die gleiche Platzierung und Rotation wie die Wurzel haben.

4.5 Animationen und Tracking des Metahumans als VR-Pawn

Dies war der komplizierteste und anspruchsvollste Teil der Arbeit. Das Endziel der Animation eines Metahumans ist eine Full-Body Inverse Kinematics aus Abschnitt 2.4. Die besondere Schwierigkeit besteht darin, aus den drei Inputs (Headset: Kopf, zwei Motioncontroller: Hände) eines Standard VR Headsets einen kompletten Körper mit Animation der Beine und Füße, des Unter- und Oberkörpers und der Arme darzustellen.

Die Umsetzung erfolgt durch das Motion Tracking des VR Headsets, das die Bewegungen mit den entsprechenden Gliedmaßen in der virtuellen Umgebung synchronisiert. Die synchronisierten Bewegungen sollen für den Benutzer so natürlich wie möglich wirken.

4.5.1 Motion Tracking

In diesem Schritt werden die Kamera- und Motion-Controller-Komponenten mit dem Kopf und den Händen verbunden und synchronisiert. Als Grundlage für die Motion-Tracking Logik diente das Youtube Video von Kaz Voten 'How To Make an IK Setup for VR In Unreal Engine 4 (Full Body Room Scale)'. Dieses Video beschreibt das Motion Tracking eines Skeletal Meshes für VR und diente als Grundlage für die Animationen des Metahumans. Der Inhalt dieses Videos wurde in der Unreal Engine 4 geschrieben und ist einer der Gründe, warum auf die Unreal Engine 4.27 gewechselt wurde.

In der Unreal Engine werden Nodes verwendet, um die Logik von Blueprints und anderen Assets zu schreiben. Die Logik im Backend wird in C++ geschrieben, aber der Editor kann sie mit Hilfe der Graph-Ansicht visualisieren.

Für den Pawn Blueprint ist der Eventgraph der Ausgangspunkt für alle Funktionsaufrufe. Die Abbildung 7 zeigt die beiden für den Pawn Blueprint relevanten Event-Trigger. Das BeginPlay-Ereignis wird zu Beginn der Instanziierung des Blueprints ausgelöst. Der Konsolenbefehl 'stereo on' aktiviert den Stereo Rendering Modus der Unreal Engine für VR. Dieser Modus ermöglicht ein tieferes Rendering und soll die Immersion in virtuellen Umgebungen erhöhen[25].
Set Tracking Origin Setzt den Ursprung für VR-Anwendungen auf *Floor Level*. Das Tracking-System sucht den nächstgelegenen Boden, ausgehend vom Startpunkt des Pawns, und setzt den Ursprungspunkt auf diesen Boden.
Der zweite Triggerpunkt im Event Graph ist der Event Tick. Even Tick ist ein Trigger, der bei jedem Frame-Update aufgerufen wird. Dieser Trigger wird

typischerweise für Bewegungsberechnungen verwendet, so auch in diesem Projekt. Bei jedem Event Tick wird die Funktion *Calculate Movement* aufgerufen.

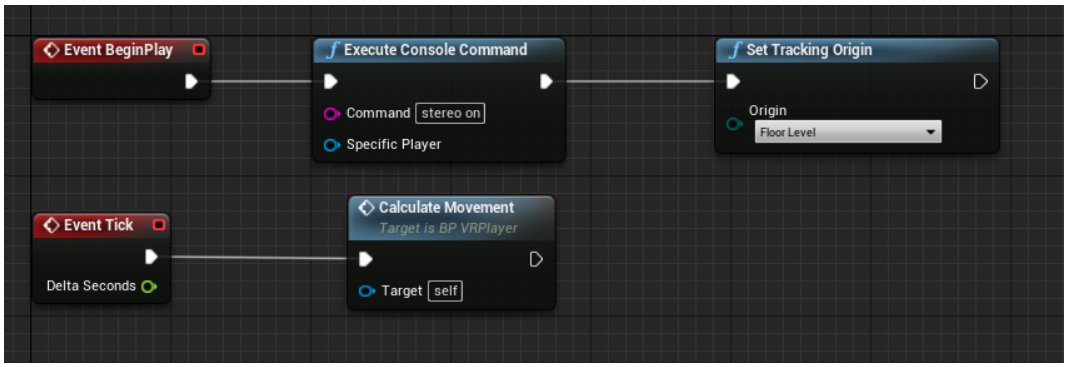

Abbildung 7: Event Graph Pawn Blueprint

Die Funktion Calculate Movement verwendet die Camera-Komponente aus der Abbildung 6 und liest deren Transform-Variable mit der Funktion *GetWorldTransform* aus. Diese Variable wird in ihre Unterkomponenten[9] aufgeteilt (split). Die Positionsvariable (Vector) und die Rotationsvariable (Rotator) werden in ihre X-, Y- und Z-Werte zerlegt. Für das Skeleton Mesh sind nur die Bewegungen auf der X- und Y-Achse und die Rotation auf der Z-Achse relevant. Diese Werte werden in einer lokalen Variablen 'Aktuelle Kameraposition' gesetzt. Abbildung 8 zeigt den Start der Calculate Movement Funktion. Ausgehend von der Set-Methode wird eine neue Funktion 'Get Distance Moved' aufgerufen, die die aktuellen X- und Y-Werte der Kamera erhält.

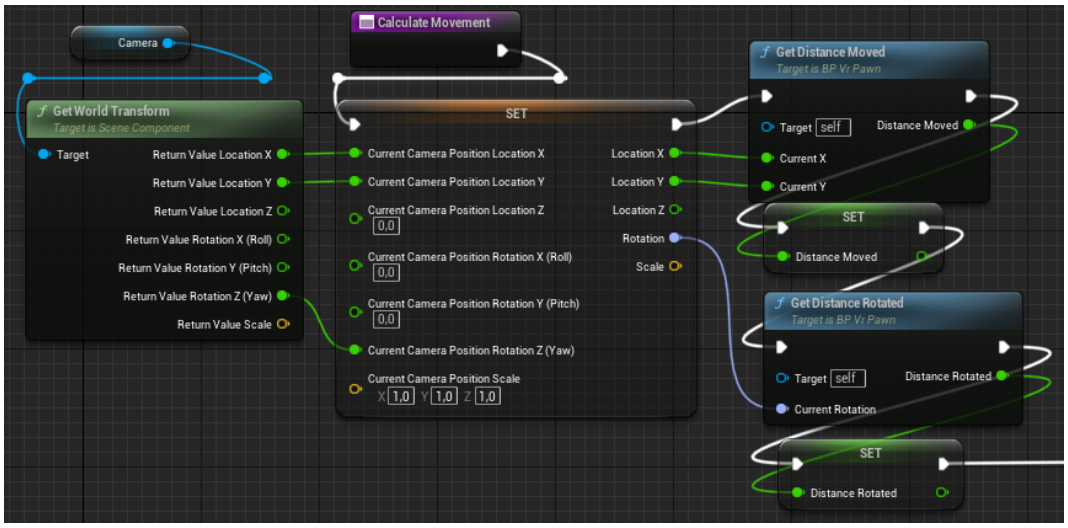

Abbildung 8: Calculate Movement Teil 1

[9]Position (Location), Rotation (Rotation)

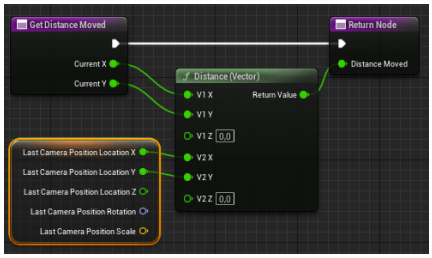

Abbildung 9: Get Distance Moved

In der Abbildung 9 ist Distance Moved zu sehen. Diese Funktion prüft, wie weit sich die Position der Kamera seit dem letzten Funktionsaufruf (Frameupdate) verändert hat. Die entsprechenden X- und Y-Werte werden der Transform-Variablen *Last Camera Position* entnommen und der Funktion Distance(Vector) übergeben.

Distance(Vector) berechnet den Abstand zwischen zwei verschiedenen 3D-Vektoren. In diesem Fall ist die horizontale Bewegung nicht relevant, daher werden die Vektoren geteilt und Z wird auf 0 gesetzt. Das Ergebnis wird in den Ausgabewert gestellt.

Zurück in der Funktion *Calculate Movement* wird der berechnete Wert als Variable mit dem Namen *Distance Moved* gesetzt.

Für die Drehung wird eine entsprechende Funktion aufgerufen (siehe Abbildung 10). Da es sich in diesem Fall nur um einen Wert (Z-Achse) handelt, ist eine normale Fließkomma-Division ausreichend. Anschließend wird dieser Wert durch die Funktion *ABS* in seinen positiven Absolutwert umgewandelt und mit dem Ausgang verbunden.

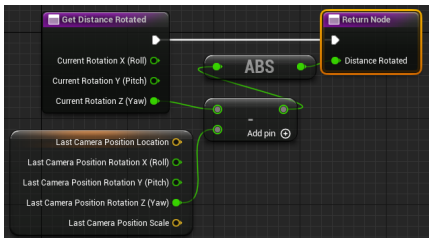

Abbildung 10: Get Distance Rotated

In der Calculate Movement Funktion, wird dies ebenfalls in eine Variable *Distance Rotated* übertragen.

4.5.2 Bewegungsschwellen

Die Abbildung 12 zeigt den restlichen Teil der Calculate Movement Function.

Ein Teil dieser Funktion dient zur Glättung der Bewegungen. Um zu verhindern, dass kleine schnelle Bewegungen zu einem großen Anstieg der Bewegung des Skeletal Mesh führen, werden Bewegungsschwellen (Movement Thresholds) eingefügt.

Zu Beginn wird eine Sequenz gestartet, die zu einem Branch (Bedingung) führt. Diese Bedingung fragt ab, ob Distance Moved oder (OR) Distance Rotated die jeweiligen 'Thresholds' überschreitet.

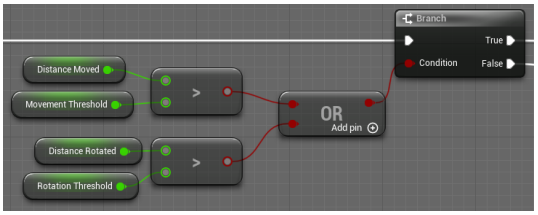

Abbildung 11: Bewegungsschwellen

Wenn diese Abfrage *true* ist, geht die Funktion zur Berechnung der Bewegungsgeschwindigkeit und der Bewegungsrichtung über.

Wenn diese Abfrage *false* zurück-

gibt, geht die Ausführung zu einem neuen Branch über.

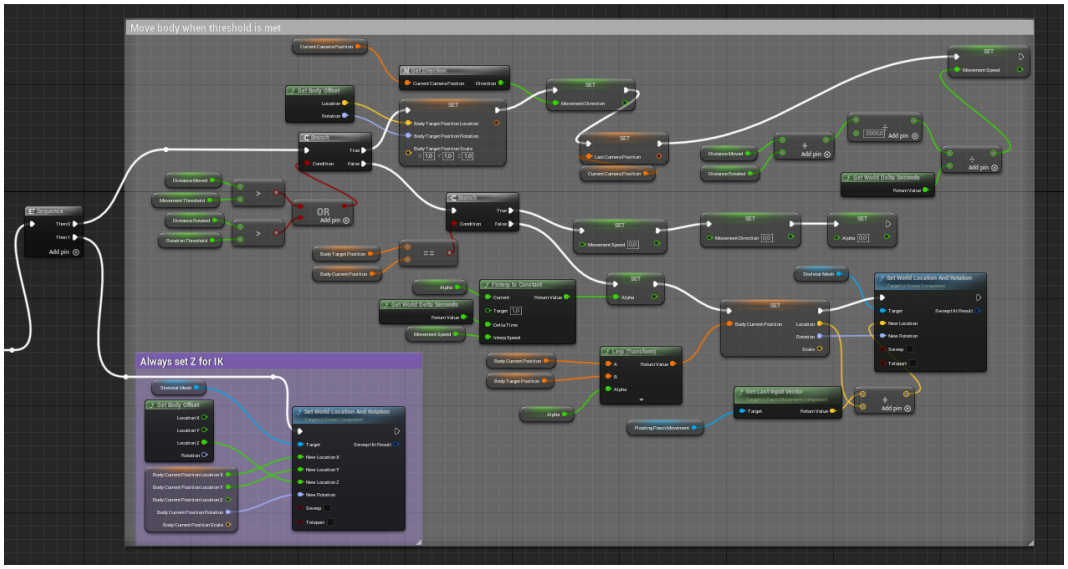

Abbildung 12: Calculate Movement Teil 2

4.5.3 Bewegungen des Skeletal Meshes

Für das Motion Tracking ist relevant, dass das Skeletal Mesh durch die Rotation und Bewegung der Kamera eine *Movement Speed* und eine *Movement Direction* Variable berechnet. Diese werden nur dann neu berechnet, wenn die Bewegungsschwellen überschritten werden.

Die Abbildung 13 zeigt die Formel zur Berechnung der Bewegungsrichtung. Es ist zu erkennen, dass die Transformationsvariablen *Current Camera Position* und *Last Camera Position* in ihren Ortsvektor und ihren Rotator zerlegt werden.

Die Rotator-Variablen werden in ihren Richtungsvektor transformiert und normiert. Z wird auf 0 gesetzt. Die Vektoren werden addiert und anschließend halbiert, um den Mittelwert zu erhalten.

Die Ortsvektoren werden ebenfalls normiert und Z wird auf 0 gesetzt. Anschließend wird der Vektor *Current Camera Position* vom Vektor *Last Camera Position* subtrahiert.

Aus den resultierenden Vektoren wird das Skalarprodukt und das Kreuz berechnet.

Aus dem Skalarprodukt wird der Arcuscosinus berechnet. Dieser wird mit 1 multipliziert, wenn das Kreuzprodukt größer als 0 ist, und mit -1 multipliziert, wenn das Kreuzprodukt kleiner als 0 ist.

Diese Berechnung stammt aus dem Video von Kaz Voeten[26]. Sie überträgt effektiv die Richtung und die Gradzahl, um die sich die Kamerakomponente innerhalb einer Bildaktualisierung geändert hat.

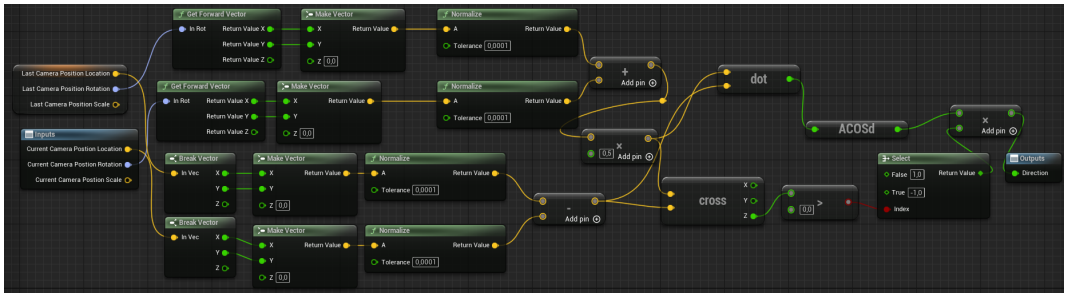

Abbildung 13: Macro Get Direction

Nach der Berechnung von *Get Direction* wird das Ergebnis in *Movement Direction* geschrieben. Anschließend kann die *Last Camera Position* gleich der *Current Camera Position* gesetzt werden, um diese für die nächste Bildaktualisierung zu verwenden.

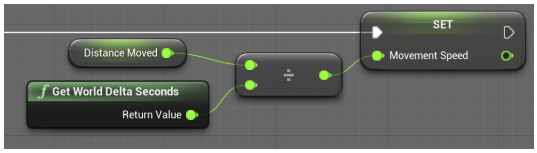

Abbildung 14: Set Movement Speed

Daraus wird die *Movement Speed* berechnet (siehe Abbildung 14). Für die Movement Speed wird die Distance Move durch 'World Delta Second' (Bildübertragungsrate) dividiert.

Diese Bewegungsvariablen werden solange beibehalten, bis die gewünschte Position erreicht ist. Die Bewegungsschwellen im Abschnitt 4.5.2 dienen der Glättung dieser Bewegungsänderung. Während der Tests und auch während der Evaluation hat sich jedoch herausgestellt, dass diese Bewegungsschwellen die Ursache für verzögerte Bewegungen sein können, da sie in Kombination mit der starken Belastung der Brille durch die Metahumans nicht schnell genug berechnet werden.

Für zukünftige Projekte ist es ratsam, hier nach Methoden zu suchen, um die Leistung der Bewegungsberechnung zu verbessern.
An dieser Stelle wird der zweite Teil der Sequenz gestartet, um sowohl die Rotation als auch die Z-Achse der *Location* für das Seleton Mesh an die Kamera anzupassen. Dazu wird die Methode *Get Body Offset* aufgerufen (siehe Abbildung 15).
Diese Methode verwendet die Höhe des aktuellen Skeleton Mesh und subtrahiert diese von der Kameraposition. Das Skeletal Mesh wird entlang der Z-Achse negativ verschoben, um den Kopf ungefähr auf der Höhe der Kamera zu positionieren. Zusätzlich wird das Skeletal Mesh um 90 Grad gedreht, um es mit der gleichen Rotation der Kamera auszurichten. Dies ist notwendig, da das Skeleton Mesh und das Unreal Camera Mesh unterschiedliche Grundaus-

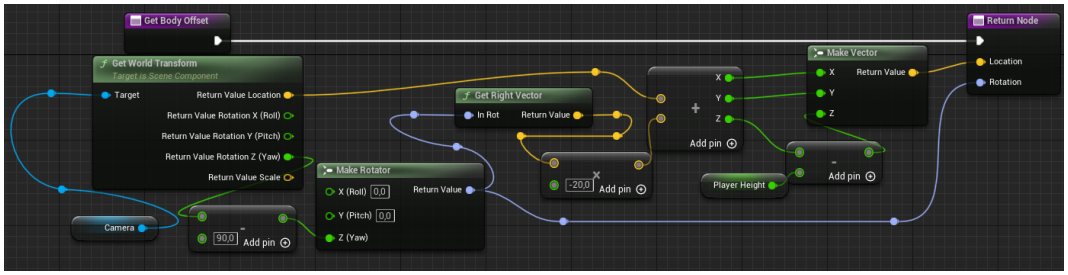

Abbildung 15: Body Offset

richtungen haben.

Zum Thema Motion Tracking und Bewegungssynchronisation gibt es viele verschiedene Umsetzungsmöglichkeiten und Ansätze.

In diesem Projekt wurden Richtungsvektoren in Kombination mit einer Geschwindigkeitsberechnung gewählt, um vorgefertigte Animations-Blueprints auszulösen, die das Skeletal Mesh gehen, sich drehen und laufen lassen.

Es gäbe noch viele weitere Möglichkeiten der Implementierung, die jedoch den Rahmen dieser Bachelorarbeit sprengen würden. Mehr dazu im Kapitel 6.

4.5.4 Animationen des VR-Pawns

Für jede weitere Animation wird nun eine Animation Blueprint erzeugt. Diese wird im Skeleton Mesh innerhalb des Pawn-Blueprint unter Anim-Class hinzugefügt.

Im EventGraph des Animation Blueprint wird eine Sequenz gestartet, in der der erste Schritt darin besteht, den Pawn Owner[10] zu validieren und nun als Variable zu setzen.

Im zweiten Schritt der Sequenz wird dieser Pawn Owner in den VR Pawn Blueprint konvertiert, um die Funktionen für Bewegung und Motion Tracking im Animation Blueprint nutzen zu können. Anschließend wird die Funktion *Foot IK* aufgerufen.

Da wir ein Standard VR Headset verwenden, ist Motion Tracking für die Füße verfügbar. Die Beine und Füße müssen also relativ zum Headset und zum Boden interagieren.

Dazu wird das Skeletal Mesh des VR-Pawns in die Funktion geladen und die Position der Füße extrahiert. Für den linken und rechten Fuß wird jeweils eine *Trace Ground* Funktion ausgeführt. Diese Funktion wird verwendet, um den Boden um 200 Einheiten sowohl in negativer als auch in positiver Richtung entlang der Z-Achse zu bestimmen. Wenn ein fester Boden gefunden wurde, wird der Abstand zum Boden ermittelt und in der Funktion *Foot IK* gespeichert.

[10] gibt Skeleton Mesh zurück, das diesen Animation Blueprint ausführt

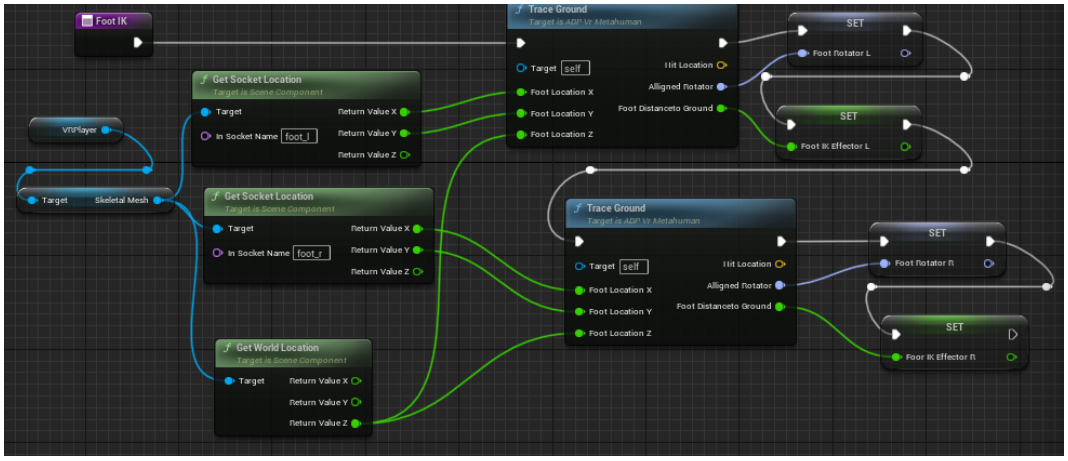

Abbildung 16: Foot IK

Im AnimGraph[27] des Animation Blueprints werden die gespeicherten Werte verwendet, um daraus eine Two-Bone Inverse Kinematics zu erzeugen (siehe Kapitel 2.4). Sobald sich das Skeletal Mesh nun dem Boden nähert, werden die Knie gebeugt und es sieht so aus, als würde der Avatar in die Hocke gehen.

Auf ähnliche Weise werden die Hände programmiert. Auch hier wird Two Bone IK der Unreal Engine verwendet, das Tracking wird in diesem Fall mit den Motioncontrollern des VR-Pawns realisiert.

Der Kopf bewegt sich evenfalls durch das Tracking des Headseats.

Auch hier sind die Möglichkeiten der Umsetzung sehr weitreichend, da es möglich ist, Motion Tracking für weitere Extremitäten wie z.B. die Hüfte oder die Beine einzurichten [21]. Für einen grundlegenden Test eines Metahumans ist die Animation jedoch völlig ausreichend.
Zu diesem Zeitpunkt funktionieren die gewünschten Animationen für das SK_Mannequin.

4.5.5 Übertragung der Animationen auf den Metahuman

Abbildung 17: Metahuman Body IK Error

Um nun den Metahuman in den aktuellen VR-Pawn einzufügen und mit den Animationen zu verbinden, muss ähnlich wie im Kapitel 4.1 vorgegangen werden.

Beim Versuch, die Animationen auf den Metahuman zu übertragen, wurden die Unterschiede in den Skeletten deutlich, denn während das Tracking für Hände, Kopf und Füße funktionierte, traten beim restlichen Knochengerüst Fehler auf. Die Knochen waren offensichtlich verdreht und verformt und die inverse Kinematik führte nicht mehr zum gewünschten Ergebnis. In Abbildung 17 ist ein Beispielbild der fehlerhaften Bewegungen zu sehen. Online ist auch eine Videoaufnahme dieses Fehlverhaltens [28].

Um dieses Problem zu beheben, wurde das Metahuman-eigene Control Rig (siehe Kapitel 2.3.4) in die Animation integriert. Durch das Einfügen des Control Rigs als Startpose wurde die Body IK korrigiert und die korrekte Inverse Kinematik wiederhergestellt. In der Abbildung 18 ist die korrigierte Pose des Metahumans zu sehen. Es gibt auch ein Beispiel Video[29]

Abbildung 18: Korrigierte Body IK

4.6 Steuerung und Funktionen des Metahuman VR-Pawns

Um nun den erstellten VR-Pawn mit Funktionen auszustatten, wurden neue Funktionen geschrieben und teilweise Methoden aus dem Unreal VR-Pawn in den Metahuman VR-Pawn übernommen.

Um die Grundfunktionen eines VR-Pawn zu erhalten, mussten die Funktionen Greifen, Bewegen und Drehen implementiert werden.

Für das Greifen wurden die Funktionen Grab Left und Grab Right vom Unreal VR-Pawn übernommen. Die Funktion sucht nach dem nächstgelegenen greifbaren Objekt in einem bestimmten Umkreis und fügt es der Hand (dem Motion Controller) hinzu.

Für die Rotation wurde die Funktion Snap Turn des VR-Pawns übernommen. Diese Funktion dreht den aktuellen Pawn um eine bestimmte Gradzahl. Die Steuerung wurde für den VR-Pawn auf andere Eingänge gelegt, da die Bewegung im Raum neu geschrieben werden musste.

Die Bewegung des Unreal VR-Pawns ist nur mittels Telportation möglich, für die Zwecke des Projekts war es jedoch wünschenswert, eine lineare Bewegung mit dem Joystick des Motioncontrollers zu implementieren.

Für die Umsetzung wurde ein Online Video[30] von russelllowe auf Youtube als Hilfe verwendet.

4.7 Online Sitzung

Für die Implementierung einer Online-Sitzung wurde das Advanced Session Plugin aus dem Bereich 3.3.2 verwendet. Zwar konnten die Grundfunktionen implementiert und die Anwendung mit Steam verbunden werden, jedoch war es nicht möglich, eine Multiuser-Sitzung zu erstellen. Mehr dazu im Abschnitt 6.

5 Evaluation der Nutzergruppe

Um die Umgebung zu testen, wurde eine Evaluierungsanwendung erstellt. Für einen direkten Vergleich wurde die Va.Si.Li-Lab Applikation[31] der Goethe Universität verwendet. Diese Applikation verwendet Meta Avatare in Unity.
Sowohl im Metahuman Projekt als auch im Va.Si.Li-Lab wurde zweimal das gleiche Experiment für die Benutzer aufgebaut.
Da die Evaluation während der Prüfungsphase des Sommersemesters 2023 stattfand, standen für den Test nur 4 Testpersonen zur Verfügung. Weitere Versuche sind notwendig, um genauere Ergebnisse zu erhalten.

5.1 Versuchsaufbau

Die Testperson erscheint in der virtuellen Umgebung und findet einen Spiegel vor. In diesem Spiegel kann er/sie sich selbst betrachten und soll nun eine Morgenroutine beginnen. Die Art und Weise, wie diese Routine durchgeführt wird, ist frei wählbar, die grundlegenden Bewegungen sind jedoch enthalten:
- Zähne putzen
- Gesicht waschen
- Haare kämmen

Im nächsten Schritt soll der Benutzer/die Benutzerin zum nächsten Ereignisstand navigieren. Dort findet er/sie einen Ball, den er/sie nun in einen Basketballkorb werfen soll. Hierbei sind mehrere Versuche möglich. Im letzten Schritt findet der User nun einen Tisch mit sechs greifbaren Würfeln vor. Die Aufgabe besteht nun darin, diese zu einer Pyramide aufzustapeln und im nächsten Schritt daraus einen Turm zu bauen.
Die Abbildung 19 zeigt die virtuelle Umgebung der Unreal Engine.

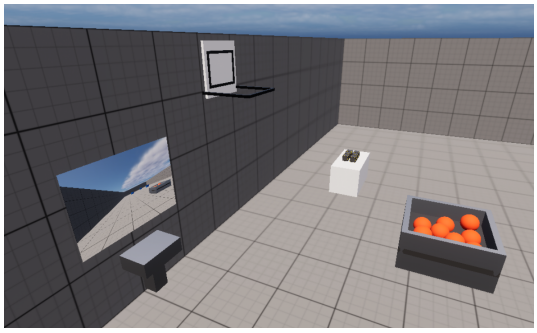

Abbildung 19: Evaluationsumgebung der Unreal Engine

5.2 Ziel der Evaluation

Ziel der Evaluation ist es, die User Experience im Vergleich der beiden Projekte bewerten zu können. Dabei geht es vor allem um die Frage, ob der Metahuman als VR-Pawn zur Immersion der Nutzerinnen und Nutzer beiträgt.
In der ersten Aufgabe der Morgenroutine liegt der Fokus auf der Beziehung

der Testperson zum Metahuman.
Die Frage ist, ob ultrarealistische Metahumans einen immersiven Effekt haben, oder ob sie vielleicht sogar eine ablehnende Haltung entwickeln oder ein Gefühl des Unbehagens entsteht.
Bei der zweiten Übung, dem Basketballwurf, geht es darum, die Gromotorik und das räumliche Vorstellungsvermögen zu testen.
In der letzten Übung werden die Feinmotorik und das räumliche Vorstellungsvermögen getestet.

5.3 Ergebnisse

Leider waren die Angaben der Proband/-innen zu gering, um daraus statistische Ergebnisse ableiten zu können. Den Testpersonen (P1-P4) wurden Aussagen vorgelegt, denen sie zustimmen sollten. Folgende Antwortmöglichkeiten standen zur Verfügung

Auswahlmöglichkeit					
Aussage	Ich stimme überhaupt nicht zu	Ich stimme eher nicht zu	teils teils	Ich stimme eher zu	Ich stimme voll und ganz zu
Wert	1	2	3	4	5

Tabelle 1: Auswahlmöglichkeiten

Proband/-in 1 und 2 starteten zuerst in der Metahumans-Testumgebung. Proband/-in 3 und 4 starteten zuerst in der Meta-Avatar-Umgebung.

5.3.1 Meta Avatare Aufgaben

In diesem Abschnitt wurde die Nutzerwahrnehmung in der Testumgebung der Meta-Avatare ausgewertet.

Morgenroutine Meta Avatare				
Aussage	P1	P2	P3	P4
Die Bewegungsabläufe waren flüssig und ich konnte alle Aktionen ohne Probleme durchführen	2	5	5	5
Die Bewegungsabläufe waren mir von meiner eigenen Morgenroutine sehr vertraut.	2	4	3	4
Während des Blickes in den Spiegel habe ich einen Bezug zu meinem Avatar herstellen können	2	4	2	4
Ich habe mich während meiner Morgenroutine mit meinem Avatar identifiziert	2	2	2	4
Der Blick in das eigene Spiegelbild hat mir ein wohles Gefühl gegeben	1	1	3	4
Der Blick in das eigene Spiegelbild hat mir ein unangenehmes Gefühl gegeben	2	1	3	2

Tabelle 2: Morgenroutine Meta Avatare

Raum Interaktion Meta Avatare				
Aussage	P1	P2	P3	P4
Die Bewegungsabläufe waren flüssig und ich konnte alle Aktionen ohne Probleme durchführen	4	3	2	4
Die Bewegungsabläufe haben sich natürlich und intuitiv angefühlt	4	4	3	4
Während der Bewegungsabläufe habe ich einen Bezug zu dem Körper des Avatars herstellen können	3	4	2	4
Der Blick auf den Körper des Avatars hat mir das Gefühl gegeben, den eigenen Körper zu betrachten	3	3	2	3
Die Bewegungen des Avatars sind sehr gut mit den eigenen Bewegungen synchronisiert (keine Verzögern oder Ruckeln)	4	4	1	4

Tabelle 3: Raum Interaktion Meta Avatare

5.3.2 Metahuman Aufgaben

In diesem Abschnitt wurden die Metahumans getestet. Auch hier wurden die gleichen Aussagen bewertet.

Morgenroutine Metahuman				
Aussage	P1	P2	P3	P4
Die Bewegungsabläufe waren flüssig und ich konnte alle Aktionen ohne Probleme durchführen	2	4	2	5
Die Bewegungsabläufe waren mir von meiner eigenen Morgenroutine sehr vertraut.	2	4	3	4
Während des Blickes in den Spiegel habe ich einen Bezug zu meinem Avatar herstellen können	2	3	3	4
Ich habe mich während meiner Morgenroutine mit meinem Avatar identifiziert	1	2	2	4
Der Blick in das eigene Spiegelbild hat mir ein wohles Gefühl gegeben	2	1	2	4
Der Blick in das eigene Spiegelbild hat mir ein unangenehmes Gefühl gegeben	4	1	3	2

Tabelle 4: Morgenroutine Metahumans

Raum Interaktion Metahuman				
Aussage	P1	P2	P3	P4
Die Bewegungsabläufe waren flüssig und ich konnte alle Aktionen ohne Probleme durchführen	2	3	2	5
Die Bewegungsabläufe haben sich natürlich und intuitiv angefühlt	1	4	3	4
Während der Bewegungsabläufe habe ich einen Bezug zu dem Körper des Avatars herstellen können	2	4	2	4
Der Blick auf den Körper des Avatars hat mir das Gefühl gegeben, den eigenen Körper zu betrachten	2	2	2	3
Die Bewegungen des Avatars sind sehr gut mit den eigenen Bewegungen synchronisiert (keine Verzögern oder Ruckeln)	3	5	2	4

Tabelle 5: Raum Interaktion Metahumans

5.3.3 Metahuman Steuerung & User Experience

Steuerung Metahuman				
Aussage	P1	P2	P3	P4
Die Steuerung und Bewegung war intuitiv bedienbar	4	5	5	3
Ich konnte mich gut mit dem Metahuman bewegen und orientieren	2	5	5	3
Ich hatte Schwierigkeit damit Gegenstände zu greifen und zu bewegen	3	2	4	4
Ich hatte oft einen Orientierungsverlust mit dem Metahuman	1	1	1	4

Tabelle 6: Steuerung Metahumans

Fazit gz				
Aussage	P1	P2	P3	P4
Der Metahuman hat in der VR-Umgebung einen großen Mehrwert geliefert und hat stark zur Immersion beigetragen.	2	2	3	4
Durch das realistische Erscheinungsbild des Metahuman konnte ich mich besonders starken Bezug zu Avatar herstellen.	1	2	3	4
Mein Avatar hat nicht dazu beigetragen das VR-Szenario anregender zu machen, ich hätte kein Avatar gebraucht.	4	4	4	2
Der Metahuman hat mir ein unwohles, unsicheres Gefühl gegeben.	2	1	3	2

Tabelle 7: User Experience Metahumans

5.3.4 Fazit

User Experience Metahuman				
Aussage	P1	P2	P3	P4
Die VR-Anwendung mit Metahumans hat mir Spaß gemacht	3	4	4	5
Ich würde mich persönlich für zukünftige Arbeiten mit Metahumans in VR interessieren	4	4	4	5
Ich kann mir eine Multiplayer VR-Umgebung mit Metahumans sehr gut vorstellen	4	4	4	5

Tabelle 8: Fazit

6 Schlussfolgerung und Ausblick

6.1 Auswertung der Evaluation

Aufgrund der geringen Datenmenge ist es nicht möglich, ein statistisch valides Ergebnis zu erhalten.

Allgemeiner Konsens war, dass Metahuman für VR ein interessantes Themenfeld für zukünftige Arbeiten sein könnte. Die Technologie war im Rahmen dieser Bachelorarbeit noch nicht so weit, dass sie einen signifikanten Mehrwert für die Benutzerinnen und Benutzer bieten konnte, aber es lässt sich vielleicht schlussfolgern, dass vieles mit der noch nicht ausgereiften Umsetzung der Animation zusammenhängt.

In vielen Fällen gab es Probleme mit dem Greifen von Objekten und Bewegungen, und auch die inverse Kinematik schien am Ende noch Probleme zu bereiten.

Im direkten Vergleich mit den Meta-Avataren konnten die Meta-Humans nicht zur Immersion der Nutzer/innen beitragen. Der Bezug zu den Avataren wurde durch den erhöhten Realismus der Avatare nicht erhöht, in einigen Fällen sogar verringert.

Zusammenfassend lässt sich sagen, dass für eine abschließende und aussagekräftige Evaluierung von Meta-Humans in VR eine ausgereiftere Technologie und ein größerer Pool an Testpersonen erforderlich wären.

6.2 Die Suche nach Quellen

Wie aus dem Literaturverzeichnis hervorgeht, ist die Dichte an verlässlichen Quellen für Metahumans noch sehr gering.

Die beste Möglichkeit ist die aktuelle Dokumentation der Unreal Engine. Zwar finden sich vereinzelt relevante wissenschaftliche Quellen, aber durch die Neuveröffentlichung der Unreal Engine 5 und den damit verbundenen Metahumans ist der Stand der Wissenschaft zum jetzigen Zeitpunkt in ständiger Bewegung.

Für die Animation von Full Body IK's in Virtual Reality gibt es derzeit noch keine fertige Lösung. Full Body IK's für VR sind immer noch eine Herausforderung, wenn kein Full Body Motion Tracking verwendet wird[21].

Für die Lösung rund um die Animationen für die Unreal Engine und die Metahumans erwies sich Youtube als eine der effektivsten kostenlosen Quellen, auch wenn sie an sich nicht wissenschaftlich ist.

Da die Aufgabe darin bestand, einen Metahuman, also einen VR-Pawn, funktionsfähig zu machen und es derzeit keine gleichwertige Lösung für eine Full Body VR IK für Metahumans gibt, war dies der einzige Weg, einen funktionierenden Ansatz zu finden.

Durch die kollegiale Beratung von Herrn Angelo Gebauer von der Inosoft AG wurde dringend davon abgeraten, Metahumans in VR umzusetzen. Der einfache Grund war die hohe Performanceanforderung.

6.3 Besondere Herausforderung des Projektes

Neben sporadischen Quellenrecherchen wird immer wieder deutlich, dass sich sowohl die Unreal Engine 5+ noch in der Entwicklung befindet, als auch die Metahumans noch nicht vollständig entwickelt sind. So sieht man auf den Discord Servern der Unreal Engine Developer Community, sowie in den Online Foren der Unreal Engine und Epic-Development oft ungelöste Fragen der User und Mitglieder.

Dies erschwert die Weiterentwicklung des eigenen Projekts, da für bestimmte Anwendungsfälle noch keine Erfahrungswerte vorliegen und man daher oft gezwungen ist, viele Aufgaben durch Trial-and-Error zu lösen.

Ein weiteres Problem, das in so kurzer Zeit kaum zu beheben ist, sind die Inverse Kinematics, die immer wieder ungenaue Bewegungen plotten. Auf der einen Seite ist es sehr gut, dass die IK dem/der Entwickler/-in einen großen Teil der Animationsentwicklung abnimmt, auf der anderen Seite hat man dadurch auch wenig Möglichkeiten, bestimmte Animationsabläufe selbst zu beeinflussen.

In letzter Instanz war es leider auch nicht möglich, eine Session mit dem Advanced Session Plugin zu erzeugen. Nach aktuellen Recherchen ist dies die beste Lösung, um zwei eigenständige Anwendungen miteinander zu verbinden, aber auch nach längerer Zeit mit etlichen Lösungsansätzen konnte kein Ergebnis erzielt werden, mit dem eine Multiplayer-Anwendung erstellt und gehalten werden konnte.

Dies war einer der größten Frustrationspunkte bei der Entwicklung der Anwendung, da die Immersion und die Metahumans in einem anderen Detaillierungsgrad hätten getestet werden können.

6.4 Future Work

Trotzdem ist Metahuman ein Framework mit großem Potential. Nicht nur die Texturen und Shader sind beeindruckend, sondern auch die Möglichkeiten der Mimik, die in diesem Projekt überhaupt nicht erforscht wurden, und das Metahuman Control Rig, das extrem mächtig und detailliert ist.

Für die Zukunft ist es interessant, eine Metahuman VR-Umgebung zu entwerfen, in der Mimik und Gestik implementiert sind und in der die Multiplayer-Funktionalität implementiert ist.

Es ist auch interessant zu erforschen, ob es andere Möglichkeiten gibt, das Motion-Tracking und die Bewegung im Raum so zu implementieren, dass die Verzögerungen der Bewegungen reduziert oder vielleicht sogar eliminiert werden.

Darüber hinaus kann eine Verbesserung der Performance auch das Rendern von Texturen in höherer Auflösung ermöglichen, sofern das Backend durch eine effizientere Programmierung entlastet wird. Vor dem Hintergrund, dass die aktuelle Anwendung nur in der niedrigsten Auflösung nutzbar ist, könnte eine Verbesserung der Grafik zu einer erhöhten Immersion der Nutzerinnen und Nutzer führen.

6.5 Fazit und persönliche Meinung

Metahuman ist ein potentiell sehr mächtiges Framework, das die Möglichkeit bietet, sehr hochauflösende Avatare für virtuelle aber auch nicht-virtuelle Produkte zu verwenden. Die Arbeit mit diesen Avataren ist alles andere als anspruchslos, doch gleichzeitig ab einem gewissen Punkt sehr intuitiv.

Das große Problem, vor allem in Bezug auf virtuelle Umgebungen, wird immer die Performance sein. Der Detailgrad der Metahumans ist extrem beeindruckend, aber auch extrem schwerfällig. Und obwohl man mit der Umsetzung

der Mimik sehr gute Ergebnisse erzielen könnte. Sehe ich derzeit noch starke Schwächen, vor allem bei den fehlerhaften inverse Kinematics, die ein wirklich immersives virtuelles Erlebnis sehr wahrscheinlich verhindern.

Nichtsdestotrotz, wenn Unreal Engine's Metahumans weiterentwickelt werden und die aktuellen Schwächen behoben werden, könnte das Framework eine sehr gute Möglichkeit bieten, die VR-Pawn Entwicklung und damit VR-Anwendungen einer breiteren Masse zugänglich zu machen.

7 Referenzen

Literatur

[1] Unreal Engine, "Vr template." https://docs.unrealengine.com/4.27/en-US/Resources/Templates/VRTemplate/. [Abruf: 18. Juni 2023].

[2] J. Misersky, D. Peeters, and M. Flecken, "The potential of immersive virtual reality for the study of event perception," *Frontiers in Virtual Reality*, vol. 3, p. 697934, 2022.

[3] about.meta, "Meta Avatar." https://about.fb.com/news/2023/04/meta-avatars-new-body-shapes-hair-clothing/. [Abruf: 18. Juni 2023].

[4] Unreal Engine, "Unreal Engine 5 release." https://docs.unrealengine.com/5.0/en-US/unreal-engine-5.0-release-notes/. [Abruf: 18. Juni 2023].

[5] A. Mehler, M. Bagci, A. Henlein, G. Abrami, C. Spiekermann, P. Schrottenbacher, M. Konca, A. Lücking, J. Engel, M. Quintino, J. Schreiber, K. Saukel, and O. Zlatkin-Troitschanskaia, "A multimodal data model for simulation-based learning with Va.Si.Li-Lab," in *Proceedings of HCI International 2023*, Lecture Notes in Computer Science, Springer, 2023. accpeted.

[6] H. E. Lowood, "virtual reality." https://www.britannica.com/technology/virtual-reality. [letztes Update: 17. Juni 2023].

[7] D. Coldewey, "Why should you care about Unreal Engine 5." https://techcrunch.com/2022/04/11/what-is-epic-games-unreal-5/. [Abruf: 17. Juli 2023].

[8] Unreal Engine, "Graph Editor." https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/Editor/UIComponents/GraphEditor/. [Abruf: 17. Juli 2023].

[9] Unreal Engine, "Nodes." https://docs.unrealengine.com/5.2/en-US/nodes-in-unreal-engine/. [Abruf: 17. Juli 2023].

[10] Unreal Engine, "Metahuman Creator." https://metahuman.unrealengine.com/mhc. [Abruf: 15. Juli 2023].

[11] michaeljcole.github.io, "Skeletal Meshes." https://michaeljcole.github.io/wiki.unrealengine.com/Matinee_Basics:_Skeletal_Meshes/. [Abruf: 20. Juni 2023].

[12] Unreal Engine, "Control Rig." https://docs.unrealengine.com/5.0/en-US/control-rig-in-unreal-engine/. [Abruf: 20. Juni 2023].

[13] Unreal Engine, "Groom Assets." https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/Hair/GroomAssetEditor/. [Abruf: 18. Juni 2023].

[14] Unreal Engine, "ULODSyncComponent." https://docs.unrealengine.com/4.27/en-US/API/Runtime/Engine/Components/ULODSyncComponent/#:~:text=This%20is%20a%20component%20that,Name. [Abruf: 17. Juli 2023].

[15] A. Aristidou and J. Lasenby, "Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver," [veröffentlich: September 2009].

[16] M. Hergaarden, "Graphics shaders," *Graphics Shaders*, p. 1, 2011.

[17] U. Engine, "Metahuman Latest Updates." https://www.unrealengine.com/en-US/blog/new-metahuman-animator-feature-set-to-bring-easy-high-fidelity-performance-c [Abruf: 15. Juli 2023].

[18] Quixel, "Quixel Bridge." https://quixel.com/bridge. [Abruf: 07. Juli 2023].

[19] Unreal Engine, "Third Person Template." https://docs.unrealengine.com/4.27/en-US/Resources/Templates/ThirdPerson/. [Abruf: 07. Juli 2023].

[20] G. Games, "How to Replace the Mannequin with a Metahuman in Unreal Engine 5." https://www.youtube.com/watch?v=VEhSX04mxOY&list=PLFTG2NqD2cIpFNFrg2vhWf8G25DRFed8B&index=4&t=460s&ab_channel=GorkaGames. [Abruf: 16. Juli 2023].

[21] P. Caserman, "Full-body motion tracking in immersive virtual reality-full-body motion reconstruction and recognition for immersive multiplayer serious games,"

[22] T. Völkner, "Chandra Third-Person Video." https://www.youtube.com/watch?v=lE45JAnDkYk&ab_channel=TobiasV%C3%B6lkner. [Abruf: 16. Juli 2023].

[23] T. Völkner, "Metahumans in VR." https://www.youtube.com/watch?v=DqWFDprwib4&ab_channel=TobiasV%C3%B6lkner. [Abruf: 16. Juli 2023].

[24] Gamium Dev, "Unreal Engine 5 Editor Lag Fix | UE5 FPS Boost For Low End PC's." https://www.youtube.com/watch?v=W9aX5lyzW9U&list=PLFTG2NqD2cIpFNFrg2vhWf8G25DRFed8B&index=34&t=6s&ab_channel=GamiumDev. [Abruf: 16. Juli 2023].

[25] Unreal Engine, "VR Performance Features." https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/VR/DevelopVR/VRPerformance/. [Abruf: 16. Juli 2023].

[26] K. Voeten, "How To Make an IK Setup for VR In Unreal Engine 4 (Full Body Room Scale)." https://www.youtube.com/watch?v=qBooEZnlAA4&t=169s&ab_channel=KazVoeten. [Abruf: 16. Juli 2023].

[27] Unreal Engine, "Anim Graph." https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/AnimGraph/. [Abruf: 17. Juli 2023].

[28] T. Völkner, "Body IK Error." https://www.youtube.com/watch?v=Rin4CWF35lI&ab_channel=TobiasV%C3%B6lkner. [Abruf: 17. Juli 2023].

[29] T. Völkner, "Corrected Body IK." https://www.youtube.com/watch?v=isIkJWOF7Xs&ab_channel=TobiasV%C3%B6lkner. [Abruf: 17. Juli 2023].

[30] russelllowe, "Adding Walking Movement To VR Pawn." https://www.youtube.com/watch?v=wNBr7Tb2CXQ&ab_channel=russelllowe. [Abruf: 17. Juli 2023].

[31] G. Abrami, A. Mehler, M. Bagci, P. Schrottenbacher, A. Henlein, C. Spiekermann, J. Engel, and J. Schreiber, "Va.si.li-lab as a collaborative multiuser annotation tool in virtual reality and its potential fields of application," in *Proceedings of 34th ACM Hypertext Conference (HT 23)*, 2023. accpeted.