Virtual-Reality Toolchain

Grundprojekt Ausarbeitung

Johann Bronsch

Sommersemester 2017

HAW Hamburg

Abstract. Das Ziel dieser Ausarbeitung ist es einen Einblick zum Zwischenstand der Projektarbeit zu liefern. Dabei sollen sowohl die technischen als auch theoretischen Grundlagen, welche für die Entwicklung einer Virtual-Reality Toolchain benötigt werden, analysiert werden. Darüber hinaus sollen die verschiedenen Schritte bei der Entwicklung von virtuellen Welten beschrieben und beispielhaft dargestellt werden.

1 Einleitung

1.1 Motivation

Der Wunsch eine vollkommen immersive, virtuelle Welt zu erschaffen ist nicht neu. Jener wurde von einigen Philosophen, Forschern oder auch Autoren formuliert. Seit der Fernsehserie Star Trek, wo solch eine virtuelle Welt vorgestellt wurde, kann man sich ein Bild von so einer virtuellen Welt und deren Möglichkeiten machen (vgl. Figure 1). Ebenso die Technologie, welche in der Serie benutzt wird, wurde in einigen Episoden erläutert.

Einige Theorien zur Funktion lassen sich anhand der Figure 1 ableiten. So wird an den Wänden die Landschaft dargestellt und die Möglichkeit sich nahezu unendlich weit fortzubewegen wird mittels Kraftfelder realisiert. Der schwierigste Part an der Umsetzung, wie sie in Star Trek beschrieben wird, ist die Generierung von Objekten mit denen man interagieren kann. In Star Trek wird dies mittels der sogenannten Replikations-Technologie realisiert. Diese erstellt aus den in der Luft vorhandenen Moleküle greifbare Objekte (vgl. [25]).

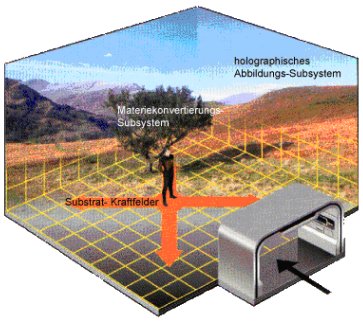

Fig. 1. Konzept Holodeck [6]

Da jedoch keine dieser Technologien, bis auf die Projektion von Landschaften auf Wänden, bei jetzigem Forschungsstand realisierbar ist müssen andere Wege gefunden werden um solche Welten möglichst realistisch zu gestalten.

Es arbeiten jedoch viele Forschungsteams an der Verwirklichung jener Idee. So wie die Forschungsgruppe aus dem Hasso Plattner Institut[1] der Universität Potsdam. Die Idee der Forschungsgruppe aus Potsdam ähnelt sehr einem Holodeck aus der Fernsehserie Star Trek. Bei dem TurkDeck, vgl. [4], handelt es sich jedoch um eine Umsetzung mithilfe eines Head-Mounted-Displays. Das Ziel war es jedoch eine immersive Umgebung zu schaffen in der, der Anwender die virtuelle Realität nicht nur sehen und hören kann, sondern auch fühlen. Hierfür wird die virtuelle Welt in der realen mithilfe von Bausteinen nachgebaut. Dabei sollen die Bausteine nur an den Stellen aufgebaut werden, an denen der Anwender diese tatsächlich auch erreicht. In der virtuellen Welt können die Bausteine mit angereicherten Informationen überblendet werden. So könnte z.B. aus einer Wand ein Display werden mit dem ein Anwender interagieren kann und darüber hinaus ein haptisches Feedback bekommt.

Für die Darstellung von virtuellen Welten werden vermehrt Head-Mounted-Displays eingesetzt. Jene bieten ein sehr immersives Feedback, bezogen auf die visuelle und auditive Wahrnehmung. Die Generierung von haptischen Feedback ist jedoch um ein vielfaches Komplexer. Ins besondere bei Objekten die nur in der virtuellen Welt existieren. Daher beschäftigen sich einige Forschungsteams explizit mit dieser Problemstellung. Die Gemeinsamkeit bei allen Lösungen jenes Problems ist die Entwicklung eines Exoskeletts, welcher ein haptisches Feedback liefern soll. Der Force-Feedback Handschuh *Dexmo*, vgl. [10], des chinesischen Start-Ups Dexta Robotics, ist solch ein Ansatz. Eine weitere sehr interessante Arbeit auf diesem Gebiet ist die *Iterative Konstruktion eines individualisierten Force-Feedback Handschuhs mittels Rapid-Prototyping-Techniken*, vgl. [26]. Das Ziel dabei ist die Entwicklung eines parametrisierten Force-Feedback-Handschuhs. Somit könnte jeder Anwender den für sich passenden Handschuh, mithilfe eines 3D-Druckes, herstellen.

Auch das Fraunhofer Institut für Integrierte Schaltungen mit Ihrem Projekt Holodeck 4.0[2] beschäftigt sich mit den Problemstellungen von Virtual Reality. Dabei werden Head-Mounted-Displays eingesetzt um virtuelle Welten darzustellen, und die Unity3D-GameEngine (vgl. Abschnitt 2.2) um solche zu erschaffen. Jedoch ist dies sehr aufwändig und komplex und kann meist nur von Personen mit umfangreichen IT-Kenntnissen umgesetzt werden.

Das Ziel dieser Arbeit ist es die technologischen Anforderungen zu analysieren. Darüber hinaus sollen auch die Arbeitsabläufe definiert und analysiert werden die für die Erschaffung von virtuellen Welten benötigt werden.

1.2 Aufbau der Ausarbeitung

Im folgenden Kapitel 2 werden zunächst die Technologien und Frameworks zur Darstellung von virtuellen Welten vorgestellt. Darüber hinaus werden einige wissenschaftliche Arbeiten im Bereich der virtuellen Realität vorgestellt.

Anschließend werden in Kapitel 3 und 4 die Anwendungen zum Erstellen und Anzeigen von virtuellen Objekten und Welten vorgestellt und erläutert. Darüber hinaus

[1] HPI - Universität Potsdam - http://hpi.de/; letzter Zugriff: 02.07.2017

[2] Holodeck - Fraunhofer IIS - http://www.iis.fraunhofer.de/de/ff/lok/proj/holodeck.html; letzter Zugriff: 02.07.2017

sollen einige Arbeitsschritte benannt werden, die das Verwenden jener Anwendungen erleichtern sollen.

In Kapitel 5 wird ein abschließendes Fazit gezogen, welches die Machbarkeit des Projektes nachweisen soll. Des Weiteren wird es einen Ausblick über die zukünftige Arbeit geben.

1.3 Laborumgebung Creative Space for Technical Innovations

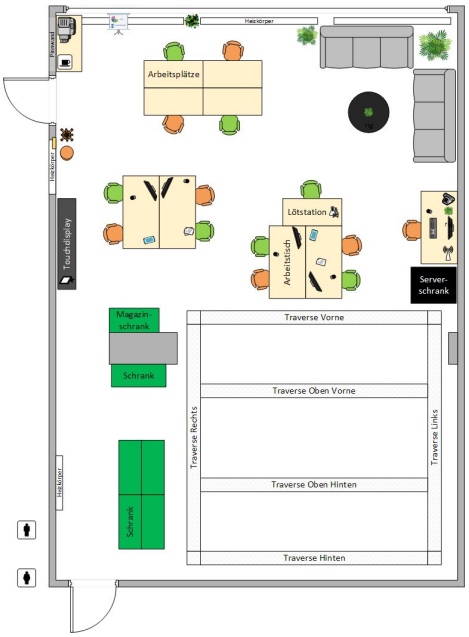

Fig. 2. CSTI-Raumplan [5]

Dieses Projekt wird im Rahmen des *Creativ Space for Technical Innovations*, kurz CSTI, des Informatik Departments der HAW Hamburg umgesetzt. Das CSTI ist ein Forschungslabor mit den Arbeitsschwerpunkten *Smart Objects*, *Emotion Tracking Systeme*, *Virtual und Augmented Reality* und weiteren. Weiterführende Informationen zu den Projekten und zum Team können der Internetpräsenz des CSTI entnommen werden[3].

[3] CSTI - HAW Hamburg - http://csti.haw-hamburg.de/; letzter Zugriff: 02.07.2017

2 Analyse

In diesem Abschnitt sollen zum einen die grundlegenden Technologien, Frameworks und einige wissenschaftlichen Arbeiten, im Bereich des Virtual-Reality, vorgestellt und erläutert werden. Ebenso deren Herausforderungen im Bezug auf virtuelle Welten.

2.1 Virtual-Reality Technik

Die Technologie zum darstellen von virtuellen Welten steckt noch in den Kinderschuhen, obwohl die ersten Versuche schon einige Jahrzehnte zurückliegen. Einer der Pioniere in diesem Bereich ist Ivan Edward Sutherland, welcher auf Figure 3 zu sehen ist. Mit der in [19] veröffentlichen Idee eines ultimativen Displays hat er vermutlich den Grundstein für die Entwicklung der ersten Anzeige-Geräte dieser Art gelegt. Diese waren jedoch nicht leistungsfähig und die Darstellungsqualität war sehr gering. Dies reduzierte im starken Maße den Grad der Immersion.

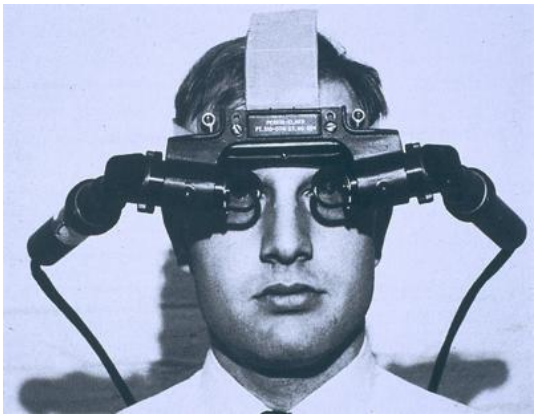

Fig. 3. Ein von Sutherland entwickeltes Head-Mounted-Display [20]

Aufgrund der rasanten Entwicklung im Bereich der Computertechnologie lassen sich heutzutage nicht nur realitätsnahe Umgebungen generieren, in die ein Benutzer eintauchen kann, sondern auch qualitativ hochwertige Anzeige-Geräte herstellen, auch Head-Mounted-Displays oder HMDs genannt.

Dies ist einer der Gründe wieso immer mehr Literatur, zum Thema Virtual Reality, erscheint, vgl. [7]. Hierbei handelt es sich um ein Buch, welches die grundlegenden Eigenschaften und Technologien hinter Virtual- und Augmented Reality vermittelt. Des Weiteren werden in [7, S.46-57] auch die Herausforderungen von VR angesprochen, wie Simulator Sickness oder auch mögliche Diskrepanzen in der Raumwahrnehmung.

In den folgenden Abschnitten sollen einige dieser HMDs benannt und beschrieben werden.

Oculus Rift

Die Oculus Rift gehört zu den ersten kommerziellen Geräten der letzten Jahre. Es gab zwar Mitte der 90er Jahre die ersten VR-Brillen, bzw. VR-Helme zu vermarkten, diese sind jedoch Aufgrund der sehr schlechten Bildqualität gescheitert (vgl. Forte VFX1 Headgear, Nintendo Virtual Boy, Sega VR, uvm. [22]).

Die erste Fassung der Oculus Rift (Development Kit 1) hatte jedoch eine nicht zufriedenstellende Bildqualität (640x800 Pixel pro Auge [18]). Diese spiegelte sich im Grad der Immersion wieder. Die Bildauflösung stieg jedoch mit jeder veröffentlichen

Version, sodass die Oculus Rift Consumer Edition eine Auflösung von 1080x1200 Pixel pro Auge hat.

Darüber hinaus ist die Implementierung in die gängigen Grafikengines sehr komfortabel, was die Entwicklung neuer Applikationen vereinfacht.

HTC Vive

Die HTC Vive ist das direkte Konkurrenzprodukt der Oculus Rift und erschien Anfang April 2016. Die Besonderheiten des Systems sind das optische Trackingsystem und die Controller, welche im Lieferumfang enthalten sind. Das Trackingsystem ermöglicht es den Entwicklern Anwendungen zu entwickeln, in denen der Nutzer sich in einem bestimmten Grad, frei bewegen kann. Darüber hinaus ist die Auflösung der Consumer Edition ähnlich der Oculus Rift und beträgt 1080x1200 Pixel pro Auge [24]. Es sind jedoch Systeme mit bis zu 8k-Auflösung in Entwicklung. Der derzeitige Auflösungssieger bei den Prototypen ist laut [24] die Weiterentwicklung der HTC Vive. Dessen Auflösung beträgt 1200x1800 Pixel pro Auge. Neben der Auflösung sind die verwendeten Technologien bei den Displays enorm wichtig. Der momentane Standard ist ein OLED Display.

Für ein immersives Erlebnis ist nicht nur die Qualität der Darstellung wichtig, sondern auch der Grad der Freiheit die ein System bieten kann. Dieser wird durch die Kabelverbindung von HMDs eingeschränkt. Auch wenn viele Unternehmen und Forschungseinrichtungen damit beschäftigt sind ein kabelloses System zu entwickeln, ist das Problem hierbei eine latenz- und verlustfreie Übertragung. Um die Datenmenge für die übertragenden digitalen Videodaten zu berechnen kann man sich folgende Formel [15] zu Hilfe nehmen.

$$(PixelBreite * PixelHöhe) * Farbtiefe * Frames = xBit \quad (1)$$

Bei einer Auflösung von 1080x1200 Pixel, einer Bildwiederholrate von 90 Hz und einer Farbtiefe von 24 Bit, wäre die Datenmenge für ein Auge wie folgt zu berechnen.

$$(1080 * 1200) * 24 * 90 = \frac{(1296000 * 24 * 90)}{(1024^3)} = \frac{2799360000}{1073741824} \approx 2,61 GBit/s \quad (2)$$

Somit ist die benötigte Datenrate für beide Augen bei 5,22 GBit/s, exklusive der Datenmenge für die Audioübertragung. Durch die Senkung der Farbtiefe auf 8 bzw. 10 Bit lassen sich die Datenmengen auf 1,74 GBit/s bei 8 Bit und 2,17 GBit/s bei 10 Bit reduzieren. Der neue Wireless LAN Standard IEEE 802.11ad erreicht eine Datentransferrate von bis zu 7 GBit/s, vgl. [28], unter Laborbedingungen. Diese Werte sind jedoch in der Praxis um ein vielfaches geringer.

Die meisten Wireless HDMI Systeme, die heute verfügbar sind, komprimieren die Daten um mit dem immensen Datenvolumen zurecht zu kommen. Durch diese Komprimierung kommt es zu einer Verzögerung. Jene ist beim Abspielen von Medien nicht problematisch, da Bild und Ton immer noch synchron abgespielt werden. Diese Verzögerung ist bei VR-Systemen jedoch spürbar und löst in den meisten Fällen die so genannte Simulator Sickness, vgl. [2, S. 14], aus.

HTC arbeitet mittlerweile an der Vive 2, welche den Angaben von [23] mit zwei 4k Displays bei 120 Hz ausgestattet ist und ohne Kabel auskommt. Die somit benötigte Datenrate würde bei einer Farbtiefe von 8 Bit bei ungefähr 14,83 GBit/s liegen.

Mobile Virtual Reality

Bei Mobile VR handelt es sich um VR-Systeme, die ein Mobiltelefon als Darstellungsmedium nutzen, vgl. Figure 4. Es existieren mittlerweile unzählige Ausführungen, z.B. aus Pappe aber auch aus Plastik. Ziel dabei ist es eine preiswerte Alternative zu den gängigen HMD-Geräten zu erschaffen. Da hierbei das eigene Mobiltelefon sowohl zur Darstellung als auch zur Berechnung von 3D-Welten genutzt wird.

Fig. 4. Darstellung eines Google-Cardboards mit einem IPhone 6s [1]

Solche Mobile VR Systeme sind daher sehr beliebt im "Low-Budget"-Bereich, da hierfür die Anschaffung eines leistungsfähigen Rechners wegfällt. Darüber hinaus werden diese Systeme stetig weiter entwickelt. Das neueste Software Development Kit von Google mit dem Namen Daydream[4] ermöglicht hierbei den einfachen Aufbau von VR-Anwendungen in der Android Entwicklungsumgebung. Zusätzlich werden auch Unity3D und Unreal-Engine Pakete angeboten, welche im folgenden Abschnitt genauer erläutert werden.

Aufgrund der geringeren Leistung eines Mobiltelefons, im Vergleich zu einem Rechner, ist jedoch die generierte Welt meist nicht so detailreich. Dies hat wiederum Auswirkungen auf den Grad der Immersion.

2.2 Entwicklungsumgebungen für grafische Anwendungen

Entwicklungsumgebungen sind in erster Linie hilfreiche Werkzeuge die den Anwender bei der Softwareentwicklung unterstützen sollen. Die Unterstützung reicht vom einfachen Autovervollständigen von Befehlen, bis hin zu Designwerkzeugen für die Entwicklung von GUIs, grafische Benutzeroberfläche (engl. graphical user interface).

Dabei existieren verschiedene Entwicklungsumgebungen für die unterschiedlichen Entwicklungssprachen und Aufgabenbereichen. Die in den folgenden Abschnitten erläuterten Entwicklungsumgebungen werden primär bei der Entwicklung von grafischen Anwendungen eingesetzt und sind daher für diesen Bereich optimiert. Daher sollen die folgen Abschnitte einen technischen Überblick über die häufig eingesetzten Entwicklungsumgebungen liefern und einige Besonderheiten hervorheben.

[4] Google VR - Daydream - http://developers.google.com/vr/daydream/overview; letzter Zugriff: 02.07.2017

Unity3D

In diesem Abschnitt werden kurz die Eigenschaften der Gaming-Engine Unity3D[5] erläutert. Eine genaue Übersicht der Funktionen und Möglichkeiten von Unity3D werden in Kapitel 4 beschrieben.

Unity3D ist eine Laufzeit- und Entwicklungsumgebung, welche die Entwicklung von 2D- und 3D-Anwendungen ermöglicht. Dabei ermöglicht Unity3D die Entwicklung von Anwendungen für die unterschiedlichsten Plattformen, wie PCs, Spielkonsolen, mobilen Geräten und Webbrowser. Jene wird durch die Verwendung der gängigsten 3D-Computergrafik APIs ermöglicht, Direct3D[6] und OpenGL[7]. Letztere wurde jedoch von der Vulkan API abgelöst und ist, so wie der Vorgänger auch, Open-Source[8].

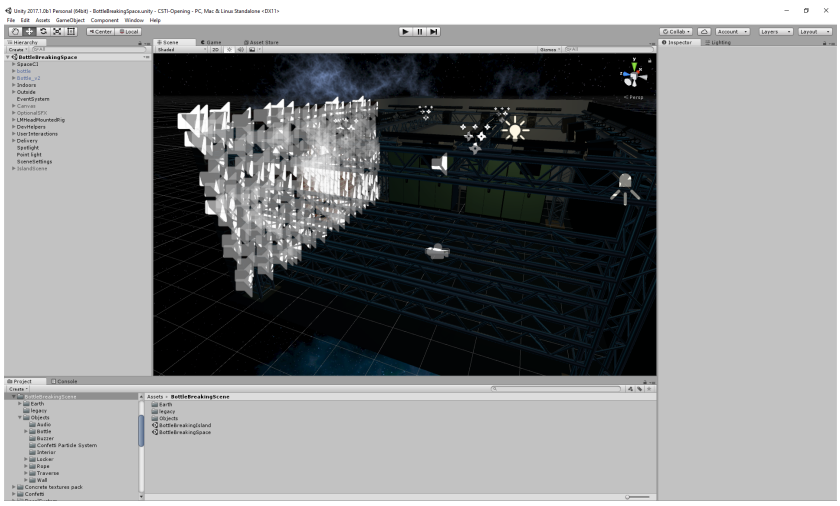

Fig. 5. Unity3D User-Interface

Um den Einstieg für Game-Designer so einfach wie möglich zu gestalten, wurde das User-Interface einer 3D-Moddeling Anwendung nachempfunden, vgl. Figure 5. Dies soll eine möglichst einfache Übersicht der einzelnen Funktionen gewährleisten und einen Überblick über die aktive Szene ermöglichen. Des Weiteren beinhaltet Unity3D eine Physik-Engine, welches das implementieren von eigenen Algorithmen unnötig macht. Darüber hinaus verfügt Unity3D über eine Vielzahl an Optimierungsverfahren, wie z.B. Level of Detail oder Occlusion Culling. Dies ist besonders bei VR-Anwendungen enorm wichtig, da eine niedrige Bildrate zur Simulator-Sickness führen kann.

[5] Unity3D - Unity Technologies - http://unity3d.com; letzter Zugriff: 02.07.2017
[6] Direct3D - Microsoft - http://msdn.microsoft.com/en-us/library/windows/desktop/ff476876.aspx; letzter Zugriff: 02.07.2017
[7] OpenGL - http://opengl.org; letzter Zugriff: 02.07.2017
[8] Vulkan - http://www.khronos.org/vulkan/; letzter Zugriff: 02.07.2017

Eine weitere Eigenschaft von Unity3D ist das Skripting, welches zur Steuerung von Szenen, Ereignissen uvm., verwendet werden kann. Der Vorteil bei Unity3D ist, dass solche Skripts zum größten Teil Plattform unabhängig sind. Dies macht es dem Entwickler einfacher für mehrere Plattformen parallel zu entwickeln, da dieser die gleichen Skripts verwenden kann. Die verschiedenen Arten von Skripts und Bibliotheken werden in Abschnitt 4.3 erläutert.

Unreal Engine

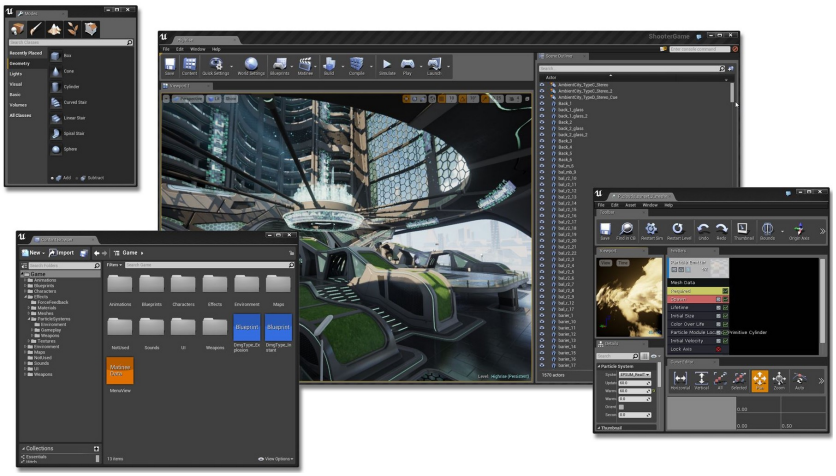

Fig. 6. Unreal-Engine User-Interface, vgl. [9]

Die Unreal Engine hat vom User-Interface Ähnlichkeiten mit Unity3D, vgl. Figure 5 und Figure 6. Der größte Unterschied aus der Sicht eines Entwicklers ist die Programmiersprache. Die Unreal Engine verwendet **C++**. Im Gegensatz zu der Unity3D-GameEngine, diese verwendet **C#** und **JavaScript**. Dies ermöglicht dem Entwickler Hardware nahe zu entwickeln. Jenes hat unter Umständen positiven Einfluss auf die benötigte Rechenleistung. Darüber hinaus stellt die Unreal Engine dem Anwender einen Material Editor zur Verfügung, vgl. Figure 7. Dieser ermöglicht es Materialien sehr genau zu bearbeiten. Hierdurch lassen sich realistische Texturen nachbilden. In Unity3D ist dies nur durch den Einsatz von zusätzlicher Software möglich, wie z.B. Substance Painter[9].

So wie Unity3D bietet auch die Unreal Engine die Möglichkeit für verschiedene Plattformen zu entwickeln, z.B. PCs, Spielkonsolen oder Mobile Endgeräte.

[9] Substance Painter - allegorithmic - http://www.allegorithmic.com/blog/substance-painter-2; letzter Zugriff: 02.07.2017

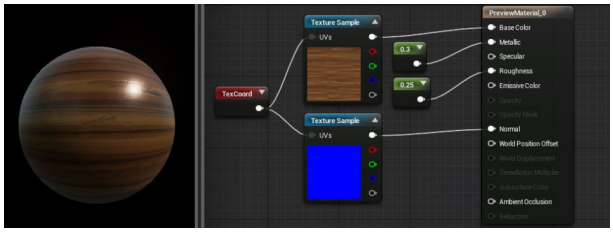

Fig. 7. Material Editor, vgl. [8]

Andere Game-Engines für graphische Anwendungen

Vollständigkeit halber sollen hier die Game-Engines genannt werden, die Open-Source sind, und für die Erzeugung von virtuellen Welten genutzt werden können. Die meisten Open-Source Game-Engines basieren auf der OpenGL Programmbibliothek. Ein Ausnahmebeispiel ist hierbei Ogre3D, jene unterstützt auch DirectX.

In einem Punkt unterscheiden sich die meisten Game-Engines jedoch nicht von einander. Die VR-Unterstützung seitens der VR-Headset-Hersteller ist sehr gering und wird daher größtenteils von der jeweiligen Community gepflegt. Somit sind solche Aussagen wie ”Ich mache mal schnell eine VR-Anwendung!” sehr unrealistisch!

Ogre (Object-Oriented Graphics Rendering Engine) ist eine in C++ geschriebene 3D-Engine, welche die Entwicklung von 3D-Anwendungen erleichtern soll. Diese wird seit 2001 unter der MIT License[10] veröffentlicht. Ogre ist jedoch keine klassische Spieleengine, da es keine typischen Elemente solch einer Engine besitzt, wie z.B. ein Sound-, Netzwerk- oder Physik-Plugin. Diese müssen vom Entwickler implementiert und angepasst werden. Der Grund hierfür ist die Ausrichtung von Ogre. Das Ziel war es eine Engine zu entwickeln, die ein breites Feld von Einsatzmöglichkeiten besitzt ohne den Entwickler einzuschränken [17]. Ein Beispiel hierfür sind die Anforderungen an eine Engine von verschiedenen Spiele-Genres. So bestehen bei einer ”Auto Rennsimulation” andere Ansprüche an die Physik-Engine als bei einem ”Jump and Run”-Spiel.

Um die gewünschten Anforderungen zu erfüllen stellt die Community von Ogre einige Programmbibliotheken zur Verfügung. Darüber hinaus lassen sich auch viele Anleitungen für die Entwicklung jener finden.

Ogre bietet dem Anwender jedoch auch einige Vorteile wie z.B.:

- Multi-Platform Unterstützung: Windows, Linux, Mac OSX, Android, iOS, Windows Phone & WinRT
- Unterstützung von OpenGL, DirectX 9 & 11 und WebGL
- Importierung von Objekten aus verschiedenen 3D Design-Tools, wie Blender, Maya und 3D Studio Max
- Level of Detail Unterstützung
- Importieren von verschiedenen Textur Typen

[10] MIT License - Open Source Initiative - http://opensource.org/licenses/MIT; letzter Zugriff: 02.07.2017

Ein Nachteil, für Game-Designer, bei einem Einstieg in die Entwicklung von Anwendungen mit Ogre ist das Fehlen einer grafischen Oberfläche. Softwareentwickler mit Erfahrungen in C++ sollten jedoch sehr schnell einen Einstieg in Ogre finden.

jMonkeyEngine

> It's a free, open source game engine, made especially for Java game developers who want to create 3D games using modern technology. The software is programmed entirely in Java, intended for wide accessibility and quick deployment. [13]

So wie Ogre besitzt die jMonkeyEngine keine grafische Entwicklungsoberfläche, dies erschwert den Einstieg. Die jMonkeyEngine beinhaltet jedoch die grundlegenden Eigenschaften einer Game-Engine, so wie z.B. eine Sound-, Netzwerk-, Physik-Engine und einigen mehr. Diese können, wie in Java üblich, simple erweitert oder überschrieben werden. Dies ermöglicht eine Anpassung der Game-Engine an die eigenen Anforderungen.

Ein Vorteil der jMonkeyEngine sind die angebotenen Utilities. Jene erlauben es dem Entwickler zusätzliche Funktionen zu implementieren oder bestehende leichter zu nutzen. Nachfolgend werden einige, häufig eingesetzte Utilities genannt und kurz erläutert [13].

- **Physics:** Die Physik-Simulation in jMonkeyEngine wird mit Hilfe von jBullet realisiert. Diese basiert auf der Bullet Physics Library, welche auch in aufwendigen, kommerziellen Anwendungen zu finden ist. Dadurch lassen sich Physikberechnungen leicht und größtenteils performant umsetzen.
- **Terrain:** Mit TerraMonkey kann man nicht nur eine Oberfläche erstellen oder generieren, sondern viel mehr eine ganze Umgebung nachbilden. Dazu gehören SkyBoxen, Beleuchtung einer Szene und auch eine automatische Optimierung von Level-of-Detail Mechanismen.
- **Cinematics:** Hierbei handelt es sich um die Implementierung eines Werkzeugs, welches dem Entwickler ermöglicht einen Film zu erstellen. Dieses wird aus den vorhandenen Szenen, Bewegungspfaden einzelner Objekte und vorhandener Sound-Events generiert.
- **Networking:** SpiderMonkey ist eine Netzwerk-Engine, welche für hohe Leistungsfähigkeit optimiert ist. Des Weiteren wird diese stetig weiterentwickelt, sodass immer mehr Funktionen hinzukommen.
- **GUI:** NiftyGUI ist eine eigenständige Bibliothek. Diese ermöglicht es Designern und Entwicklern eine Oberfläche für ihre Anwendung zu erstellen. Jene können in einem XML Format exportiert werden und mithilfe von Plugins in jMonkeyEngine eingelesen werden.

2.3 Interaktionsmöglichkeiten in der virtuellen Welt

Die Interaktion in virtuellen Welten geschieht, aus menschlicher Sicht, grundsätzlich über drei spezifische Sinnesarten. Die hierbei angesprochenen Sinne sind der visuelle, akustische und der haptische Sinn, vgl. [7, S.34]. Es können jedoch Diskrepanzen in

der Wahrnehmung eintreten. Vermehrt kann man das bei der Raumwahrnehmung beobachten und auch nutzen. Ein Beispiel für diesen Effekt ist das *Redirected Walking*. Hierbei lässt man eine Person im Kreis laufen, obwohl dieser das Gefühl hat in der virtuellen Welt geradeaus zu gehen, vgl. [11] (visuell) und [27] (auditiv). Die verschiedenen Wahrnehmungsarten können aber auch zur Interaktion genutzt werden. So kann man, wie in [12] beschrieben, die auditive Wahrnehmung zur Interaktion nutzen und andere Wahrnehmungsarten damit entlasten.

Die womöglich interessanteste Eigenschaft von Interaktionen in virtuellen Welten ist die Ähnlichkeit zu den Interaktionen in der realen Welt. Mit Hilfe einiger Technologien lassen sich virtuelle Objekte, welche angefasst werden können, auch erfühlen. Diese Eigenschaft der VR-Technologie weckt daher immer mehr das Interesse von anderen Disziplinen, wie z.B. die Psychologie, vgl. [3].

In diesem Abschnitt sollen jedoch die grundlegenden Interaktionsmöglichkeiten in virtuellen Welten erläutert werden. Erweiterungen dieser Technologien werden in Abschnitt 5.1 erwähnt.

Controller

Interaktionen mithilfe von Controllern sind in Unity3D recht einfach umzusetzen. Dies liegt an den begrenzten Möglichkeiten eines Controllers. So können simple Funktionen wie das Auslösen einer Eigenschaft mit einem Knopf verbunden werden. Jenes ist jedoch nur für eine kleine Zahl an Interaktionen immersiv.

Der Vorteil von Controllern ist jedoch immens. Aufgrund der bereits implementierten Schnittstellen für die HTC Vive und Microsoft Controller in Unity, lassen sich Interaktionen bereits im frühen Entwicklungsstand ausführen. Des Weiteren lassen sich auch eigens entwickelte Controller, vgl. Abschnitt 2.4, über native Plugins, vgl. Abschnitt 4.3, einbinden. Hierbei kann man zwischen zwei Typen, bezogen auf deren Darstellungsart, unterscheiden.

Mixed-Reality Controller sind jene, die in der virtuellen Welt mit einem virtuellen Objekt *überblendet* werden, sodass dieser eine andere Darstellungsform bzw. -art in der virtuellen Welt erhält. Somit lässt sich die Funktionalität eines Controllers erweitern, welches jedoch begrenzt ist auf die Form des Controllers.

Abbild eines realen Objektes in der virtuellen Welt. Hierbei wird ein virtuelles Abbild eines realen Objektes erstellt. Dieser dient gleichzeitig als Controller. In Figure 8 ist ein Beispiel solch eines Controllers dargestellt. Es handelt sich hierbei um einen Joystick, welcher zum steuern von z.B. Raumschiffen genutzt werden kann. Der Vorteil hierbei ist der sehr hohe Grad an Immersion, es wird genau das angefasst was gesehen wird. Der große Nachteil ist jedoch die sehr begrenzte Anzahl an Einsatzmöglichkeiten. Eine Erweiterung dieser Idee wird im nächsten Abschnitt beschrieben.

Leap-Motion Die Leap-Motion ist, wie im vorherigen Abschnitt erwähnt, eine Erweiterung des Konzepts der Abbildung von realen Objekten in die virtuelle Welt. Hierbei wird jedoch das wichtigste menschliche Werkzeug abgebildet, die menschliche Hand, vgl. Figure 9. Somit lassen sich nahezu unzählige Interaktionen umsetzen. Angefangen

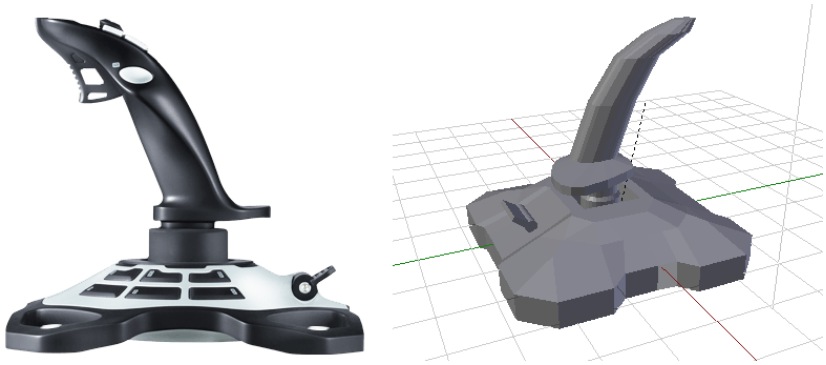

Fig. 8. Abbild eines realen Objektes in der virtuellen Welt

von der Modifikation von Objekten, bis hin zur Steuerung von virtuellen Displays. Dabei fühlen sich die meisten Interaktionen sehr immersiv an und erhöhen das Gefühl in einer Simulation zu sein.

Einen Nachteil hat diese Technik. Bei der Interaktion mit virtuellen Objekten wird kein haptisches Feedback an den Anwender weitergegeben. Somit erkennt der Anwender eine erfolgreiche Interaktion nur an dem visuellen Feedback. Dieses Problem ist jedoch bekannt, was viele Entwickler und auch Unternehmen dazu ermutigt Lösungen hierfür zu entwickeln, vgl. [26] und [21].

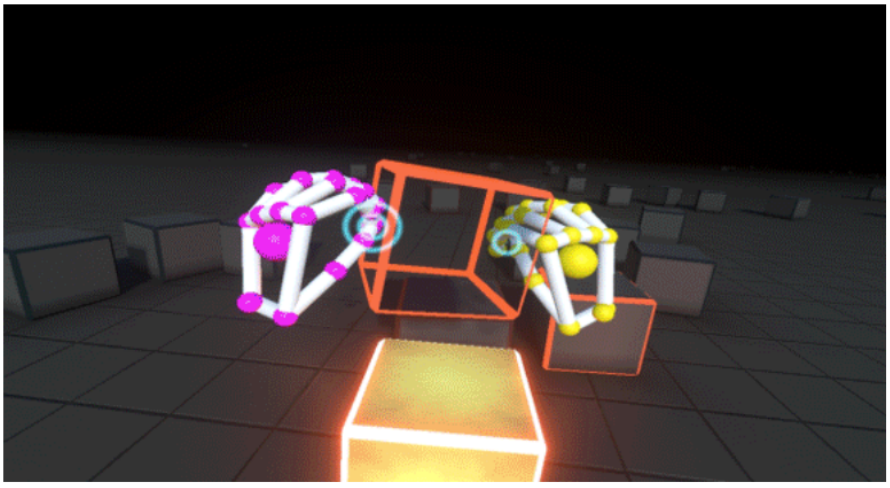

Fig. 9. Blocks Demo for Leap Motion [14]

2.4 Objekte in der virtuellen Welt

In allen Anwendungen, die eine Rendering-Engine verwenden, sind virtuelle Objekte ein zentrales Element. Je detaillierter diese Objekte sind, desto realer kann eine virtuelle Welt wirken. Zu beachten ist hierbei für welchen Zweck diese erschaffen werden. Eine sehr grobe Unterteilung kann mithilfe des Einsatzgebietes erreicht werden. So sind virtuelle Objekte, die für Computer animierte Filme erzeugt worden sind, meistens detailreicher, als jene für z.B. Spiele. Für Computer animierte Filme werden zudem andere Rendering-Verfahren eingesetzt. Diese brauchen unter Umständen viel Zeit für einen Frame, 1 Bild in einer Szene. Da HMDs jedoch auf hohe Frameraten angewiesen sind, sind solche Rendering-Verfahren hierfür ungeeignet. Daher sollte man bei der Wahl der Anwendungen zur Modellierung von virtuellen Objekten darauf achten.

Die bekanntesten Anwendungen für die Modellierung von virtuellen Objekten sind Blender, Maya und 3dStudioMax. Im folgenden werden Blender und Maya kurz vorgestellt.

Blender

Blender[11] ist ein Open-Source Software-Paket für die 3D-Modellierung. Jenes ermöglicht es dem Anwender nicht nur komplexe virtuelle Objekte zu erstellen, sondern auch diese mit einer Textur zu versehen, wie z.B. zu Animieren uvm. Aufgrund der sehr lebendigen Community wurden viele Erweiterungen für Blender entwickelt. Diese erleichtern nicht nur die Arbeit mit Blender, sondern erweitern die Funktionen von Blender enorm. Einige Funktionen von Blender werden in Abschnitt 3.1 näher beschrieben.

Maya

Maya[12] ist im Gegensatz zu Blender kostenpflichtig, ist jedoch sehr populär und wird daher von vielen Entwicklungsstudios verwendet. Auch mit Maya ist es möglich 3D Objekte zu designen, jedoch ist die Texturierung in Maya viel genauer als in Blender. Ebenso die Animation von Objekten ist in Maya möglich. Darüber hinaus lassen sich die erstellten Modelle mit Maya in andere Tools wie Mixamo[13] leichter importieren. Diese ermöglichen die Generierung von Bewegungsabläufen für Animationen oder Rendering-Engines, wie z.B. Unity3D.

Virtuelle Objekte übertragen in die reale Welt

Um möglichst genaue Controller für die virtuelle Welt zu erstellen ist es unter Umständen notwendig diese in der virtuellen Welt zu entwickeln und zu testen. Dies ermöglicht es dem Entwickler das gewünschte Objekt vorab zu testen und darüber hinaus ist jener nicht auf bestimmte Produkte angewiesen. Somit lassen sich für die einzelnen Anwendungsfälle *eigene* Controller designen. Jene können mittels 3D-Druck gedruckt und in die reale Welt übertragen werden, vgl. Figure 10.

[11] Blender - Blender Foundation - http://www.blender.org/; letzter Zugriff: 02.07.2017

[12] Maya - Autodesk - http://www.autodesk.de/products/maya/overview; letzter Zugriff: 02.07.2017

[13] Mixamo - Adobe - http://www.mixamo.com/; letzter Zugriff: 02.07.2017

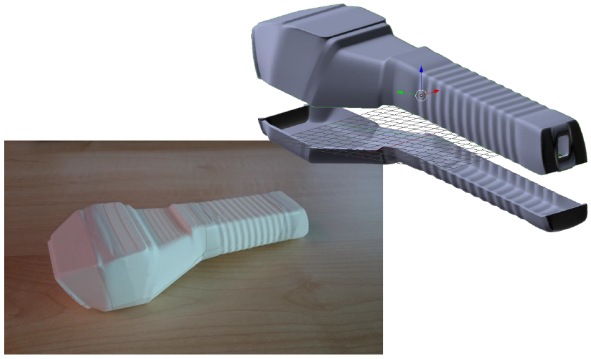

Fig. 10. Virtuelle Objekte übertragen in die reale Welt

2.5 Fazit

Wie in Abschnitt 1.1 erwähnt ist eines der Ziele dieser Ausarbeitung die Erstellung einer Toolchain, welche zum erstellen von virtuellen Welten benötigt wird. Anhand der zuvor erfolgten Analyse wird zum erstellen einer virtuellen Welt eine Anwendung benötigt, die virtuelle Objekte erstellen und animieren kann. Des Weiteren wird eine Rendering-Engine benötigt. Jene soll dem Entwickler dabei unterstützen die virtuelle Welt zu planen und zu erstellen. Abschließend soll diese in die virtuelle Welt übertragen werden, vgl. Figure 11.

Fig. 11. Geplante Toolchain

In Abschnitt 2.4 wurden zwei Anwendungen vorgestellt, die zum Erstellen von virtuellen Welten verwendet werden können. Auch wenn Maya im professionellen Bereich eingesetzt wird und in vielen Belangen etwas besser ist als Blender, ist dieser jedoch Open-Source und somit kostenfrei. Des weiteren ist er auch sehr leicht erweiterbar mit neuen Funktion. Somit fällt hier die Wahl auf Blender, welcher zudem auch für Einsteiger sehr freundlich ist, vgl. Figure 12.

Die Wahl der richtigen Rendering-Engine beruht auf der Analyse aus Abschnitt 2.2. Da die Kostenfrage bei allen genannten Rendering-Engines entfällt, ist hier die Einsteigerfreundlichkeit das wichtigste Auswahlkriterium. In diesem Punkt sticht Unity3D am meisten hervor, vgl. Figure 12.

Abschließend viel die Wahl des Anzeigegerätes auf die HTC Vive, da diese von vornherein die Möglichkeit anbietet sich frei im Raum zu bewegen. Darüber hinaus ist die Einbindung der HTC Vive durch ein vom Hersteller bereitgestelltes Plugin garantiert, vgl. Figure 12.

Fig. 12. Toolchain mit den geplanten Anwendungen

In den folgenden Kapiteln soll die Umsetzbarkeit dieser Toolchain nachgewiesen werden. Dies wird mithilfe eines Cowboys realisiert. Jener wird in Blender erstellt und mithilfe von Unity3D in die virtuelle Welt übertragen.

3 Blender

In diesem Abschnitt sollen die grundlegenden Funktionen von Blender erläutert werden. Darüber hinaus wird auf einige Arbeitsschritte hingewiesen, auf die geachtet werden sollte um zukünftige Mehrarbeit zu vermeiden.

3.1 Funktionen

Dieser Abschnitt soll kein Tutorial für die Benutzung von Blender sein[14]. Es werden jedoch einige Funktionen genannt, welche die Arbeit mit Blender erleichtern sollen.

Die meisten Funktionen in Blender sind über Tastaturkürzel zu erreichen, daher ist es sinnvoll zu Beginn diese Kürzel zu lernen. Hierfür kann man sich die Dokumentation von Blender zur Hilfe nehmen[15] oder auch nach Tastatur-Layouts.

Modellierung

Objekte lassen sich grundsätzlich auf zwei verschiedenen Arten modellieren, bzw. modifizieren. Im Editier-Modus können Objekte auf primitive Weise modifiziert werden. So lassen sich die einzelnen Knoten, Kanten oder Flächen verschieben, skaliert oder auf mithilfe der zahlreichen Funktionen modifiziert werden. Nachfolgend werden die wichtigsten Funktionen genannt, vgl **??**. Es existiert noch eine Vielzahl weiterer nützlicher Funktionen. Empfehlenswert ist es jedoch diese während eines Tutorials anzuwenden und dadurch die Funktionalität dahinter zu verstehen.

[14] Blender besitzt eine sehr aktive Community, daher lassen sich sehr gute Einsteiger-Tutorials auf der Videoplattform Youtube finden.

[15] Blender Wiki HotKeys - Blender Foundation - http://wiki.blender.org/index.php/Doc:2.4/Reference/Hotkeys/All; letzter Zugriff: 02.07.2017

- **Subdivide:** Diese Funktion ermöglicht das Unterteilen des ausgewählten Objektes. So wird durch ein einfaches Subdivide die Anzahl der Flächen eines Würfels von 6 auf 24 erhöht. Das erhöhen der Flächenanzahl kann für das erstellen von Objekten von Vorteil sein, dies erleichtert die Umformung eines Objektes in das gewünschte Objekt.
- **Subdivide Smooth:** Analog zu der vorherig erwähnten Funktion. Jedoch werden hier, wie der Begriff Smooth schon vermuten lässt, die Kanten geglättet, sodass ein Würfel nach x-Iterationen zu einer Kugel werden könnte.
- **Remove Doubles:** Eine sehr nützliche Funktion, da diese unnötige (übereinander liegende Knoten) entfernt.
- **Hide:** Versteckt das ausgewählte Objekt.
- **Reveal:** Zeigt alle versteckten Objekte wieder an. Hierbei ist es von Vorteil das Tastaturkürzel im Kopf zu behalten (ALT + H)!
- **Flip Normals:** Während der Modellierung kann es dazu kommen, dass die Normalen einer Fläche nach innen zeigen. In der Regel fällt das während der Simulation auf. Diese Funktion kann dabei helfen den Fehler schnell zu korrigieren.
- **Select All:** Markiert alle Elemente. Diese Funktion hat verschiedene Auswirkungen im Editier-Modus und Objekt-Modus. Im Editier-Modus wird das editierte Modell komplett selektiert, im Objekt-Modus werden alle Objekte der Szene ausgewählt.
- **Border Select:** Alle Elemente (Objekt-Modus) oder alle Knoten/Kanten/Flächen (Editier-Modus) die im, mithilfe der Maus gezogenen, Bereich liegen werden ausgewählt. Hierbei sollte man jedoch darauf achten ob der Solid- oder der Wirefrane-Modus aktiv ist. Bei dem Solid-Modus werden nur die sichtbaren Elemente markiert und im Wireframe-Modus alle Elemente die im Bereich liegen.
- **Area Select:** Analog zum Border Select lassen sich Elemente mithilfe eines *Pinsels* selektieren. Es gilt hierbei die selbe Eigenschaft wie zuvor. Bei dem Solid-Modus nur die sichtbaren Elemente, beim Wireframe-Modus alle im Bereich liegenden Elemente.
- **Extrude:** Eine sehr wichtige Funktion beim modellieren bzw. modifizieren von Objekten. Hierbei wird das Objekt um den ausgewählten Teil erweitert.
- **Scale:** Skalieren des ausgewählte Elements. Hierbei gibt es mehrere Varianten der selben Funktion (S-Taste, Alt+S, Shift+Alt+S). Empfehlenswert ist das ausprobieren der verschiedenen Varianten um die richtige für den gewünschten Effekt zu finden.

Position in der Toolchain

Anhand der zuvor benannten Funktionen wird im nächsten Abschnitt der Cowboy modelliert, vgl. Figure 14. Dieser soll die Machbarkeit demonstrieren, das in Blender virtuelle Objekte erstellt, texturiert und animiert werden können.

Blender Cowboy

Die im vorherigen Abschnitt genannten und einige weitere Funktionen wurden dazu genutzt um ein einfaches Model eines Cowboys in Blender zu erstellen. Dieser wurde mithilfe eines Tutorials [16] erstellt und teilweise animiert.

Im ersten Schritt wurde das Modell mithilfe einer 2D-Vorlage erstellt. Blender ermöglicht das importieren von Bildern als Hintergrund, dies erleichtert das modifizieren

Fig. 14. Erster Schritt: Blender

des eigenen Modells anhand einer Vorlage. Darüber hinaus ist es möglich verschiedene Vorlagen für die unterschiedlichen Ansichten (Vorne, Hinten, Rechts, Links, Oben und Unten) auf das Modell zu definieren. Das Ergebnis von Schritt Nr. 1 ist in Figure 15 zu sehen.

Anschließend wurde das erstellte Modell mit einer Textur versehen. Hierfür musst zunächst sichergestellt werden das die Normalen aller Flächen nicht nach innen zeigten. Dies konnte durch die Flip Normals Funktion getestet werden. Darauf folgend wurde eine UV-Map erstellt. Diese bildet die Flächen eines 3D-Models auf einer 2D-Fläche ab. Dies ermöglicht ein *bemalen* der erstellten UV-Map, welche anschließend über das 3D-Model *gestülpt* wird.

Die UV-Map für den Cowboy ist in Figure 16 rechts dargestellt. Um die Übersicht zu verbessern wurden die Gitternetze des fertigen Meshes in die UV-Map integriert. Hierbei ist es empfehlenswert detailreiche Bereiche eines Objektes sehr groß darzustellen, dabei nimmt die Skalierung der einzelnen Elemente in der UV-Map keinen Einfluss auf das 3D-Modell.

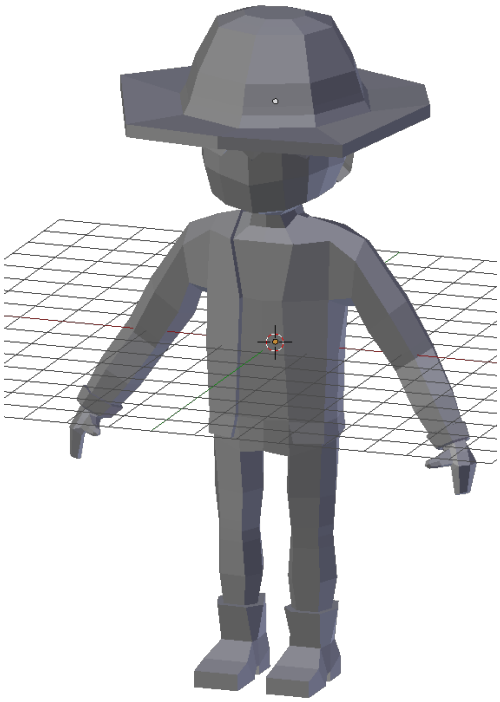

Fig. 15. Fertiges Mesh eines Cowboys

Nach dem erstellen der Textur musste der Cowboy animiert werden. Hierfür benötigt ein Objekt im Regelfall ein Skelett. Dieses gibt vor welche Elemente bei einer bestimmten Bewegung beeinflusst werden. Dabei muss das erstellte Skelett nicht dem menschlichen entsprechen. Nachdem erstellen der Skelettes sollten die sogenannten Control-Bones hinzugefügt werden. Diese erleichtern, wenn jene richtig konfiguriert

sind, die Positionierung des Modells in die gewünschte Pose. Mithilfe dieser Control-Bones lässt sich auch ein Bewegungsablauf erstellen, welcher in Unity3D importiert werden kann. Der fertig modellierte, mit einer Textur und einem Skelett versehendem, Cowboy wird in Figure 16 links dargestellt.

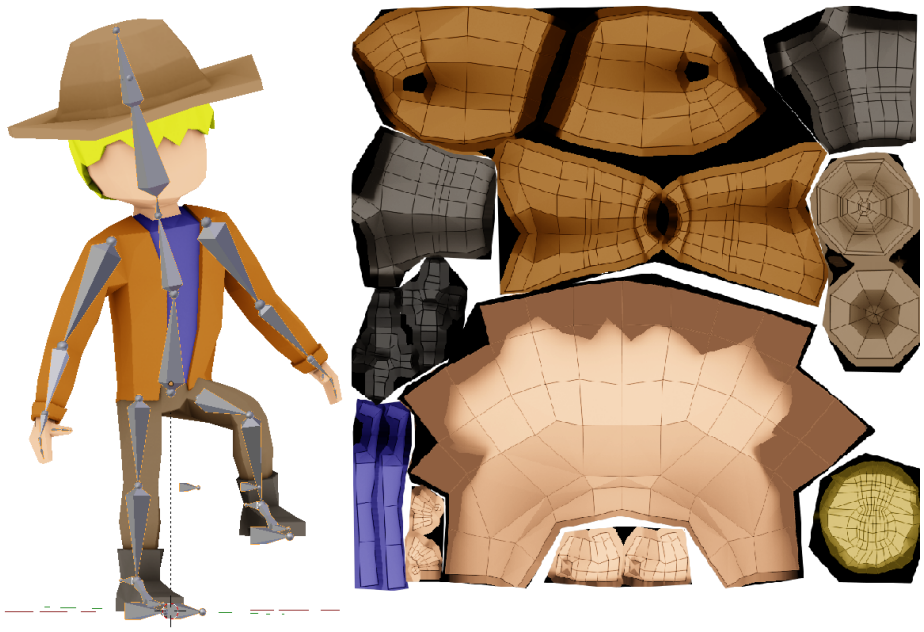

Fig. 16. Fertig texturiertes und mit Knochen versehenden Mesh

3.2 Tipps bei der Arbeit mit Blender

In diesem Abschnitt wird auf einige Arbeitsschritte hingewiesen. Diese sollen nicht nur die Arbeit mit Blender erleichtern, sondern auch die Implementierung der Blender-Modelle in Unity3D.

Wie bei jedem Projekt ist es sinnvoll sich vorher zu überlegen welches Ziel man verfolgt, z.B. erstellen eines Modells für Unity3D, 3D-Druck oder einen Computer animierten Film. Bei Modellen für Unity3D sollte die Szene vorher aufgeräumt werden, sodass nur das gewünschte Objekt sich in der Szene befindet. Da Unity3D beim Import von Blender-Dateien alle Elemente, außer der Kamera und der Beleuchtung, importiert. Für den 3D-Druck ist es notwendig die Datei zu exportieren, in der Regel ist es das STL-Dateiformat. Normalerweise ist das Export-Format aktiviert. Sollte dies jedoch nicht der Fall sein, so kann das in den *Blender-User-Preferences* nachgeholt werden. Diese werden mit der Tastenkombination *CRTL+ALT+U* aktiviert. Das erstellen von Computer animierten Filmen erfordert nicht nur eine gut strukturierte Szene, sondern

auch gute Kenntnisse über die verschiedenen Blender-Renderer. Da die Beschreibung der verschiedenen Renderer die Rahmen dieser Arbeit sprengen würden, empfiehlt es sich einige Tutorials in Bezug auf die Renderer durchzuarbeiten.

Die folgende Checkliste beschreibt die wichtigsten Punkte, auf die geachtet werden sollte beim vorbereiten des Blender-Modells für Unity3D.

- **Ein Objekt in der Szene**
- **Position des Objektes:** Das Objekt sollte im Nullpunkt positioniert werden, bei z.B. einem Haus sollte dieser am untersten Punkt in der geometrischen Mitte sein. Dies erleichtert die Positionierung in Unity3D
- **Rotation:** Da Blender und Unity3D verschiedene Koordinatensysteme besitzen ist es in einigen Fällen sinnvoll das Objekt vor dem Import in Unity3D zu rotieren.
- **Skalierung:** Die Skalierung der Objekte sollte in Blender durchgeführt werden, sonst führt es zu verfälschten und unter Umständen unterschiedlichen Werten in Unity3D. Daher sollte ein Standartmaß für Objekte verwendet werden.
- **Objekte nur mit UV-Map exportieren:** Sollte das nicht der Fall sein, kann es zu sehr großen Leistungseinbrüchen führen.

3.3 Fazit

In diesem Kapitel wurde gezeigt, dass es möglich ist mit Blender virtuelle Objekte zu erstellen, texturieren und zu animieren. Darüber hinaus wurden einige Stolpersteine genannt die beim Erstellen von Objekten auftreten könnten. Anschließend wurden einige Schritte genannt um diese Stolpersteine zu vermeiden.

Abschließend wurde ein Controller modelliert und mithilfe eines 3D-Druckers ausgedruckt und somit die Machbarkeit einer Übertragung von virtuellen Objekten in die reale Welt nachgewiesen, vgl. Figure 10.

4 Unity3D

4.1 Einleitung

In Abschnitt 2.2 wurden die grundlegenden Eigenschaften von Unity3D beschrieben. Dieses Kapitel beschränkt sich daher auf die Modellierung, Plugin-Entwicklung und Assets. Um jedoch den Umfang der Ausarbeitung nicht zu sprengen, werden die einzelnen Themengebiete nur grundlegend erläutert. Mit den Einzelheiten jener werden sich zukünftige Arbeiten beschäftigen.

4.2 Modellierung

Mit steigender Erfahrung steigt meist der Umfang der Projekte. Deshalb sollte sich jeder Entwickler eine Struktur für seine Projekte überlegen und diese umsetzen. Hierfür bietet Unity3D nicht nur eine Ordner-Struktur, sonder auch eine Objekt-Hierarchie für jede Szene an. Dabei erbt jedes Objekt die Eigenschaften seines Mutterobjektes. Dieses Konzept ist für Entwickler nicht neu, könnte jedoch für Designer gewöhnungsbedürftig sein.

Das Erstellen einer Szene erfolgt meistens nach dem selben Schema. Hierbei werden Objekte der Szene hinzugefügt und positioniert. Dabei können die Objekte entweder im Szeneansicht oder im Inspektor positioniert werden. Beim letzteren werden XYZ-Werte eingegeben um die Position zu bestimmen. Darüber hinaus können im Inspektor die Eigenschaften eines Objektes verändert werden, z.B. die Größe, und auch neue Eigenschaften hinzugefügt werden, wie z.B. Physik.

Das hinzufügen von Blender-Modellen in Unity3D, solange diese sich im Assets-Ordner befinden, ist trivial. Dazu wird das gewünschte Objekt entweder direkt in die Szeneansicht per Drag&Drop abgelegt oder auf dem selben Weg in die Hierarchieansicht. Aufgrund der unterschiedlichen Koordinatensysteme von Unity3D und Blender, rotiert Unity3D das Objekt automatisch. Die Rotation sieht man anschließend im Inspektor. Es ist jedoch ratsam mit Objekten zu arbeiten die in Unity3D im Null-Punkt richtig dargestellt werden. Deshalb sollten die Blender-Modelle wie in Abschnitt 3.2 vorbereitet werden.

4.3 Plugin-Entwicklung

Die Plugin-Entwicklung in Unity3D ist in vielen Bereichen sehr hilfreich. Zum einen können Schnittstellen von Anwendungen oder Sensoren angesprochen werden und zum anderen können dadurch Bestandteile der Anwendung ausgelagert werden. Dies ist in der Regel bei aufwendigen Berechnungen sinnvoll.

Um den Rahmen der Ausarbeitung nicht zu sprengen wird an dieser Stelle auf nachfolgende und auf dieser Arbeit aufbauende Arbeiten verwiesen, da diese sich ausgiebig mit diesem Thema befassen werden.

4.4 Assets

Assets sind Unity-Container die entweder nur ein Skript zur Lösung einer speziellen Aufgabe beinhalten können oder eine komplette Bibliothek inklusive Modelle und Beispiele. In der Regel werden Assets dazu verwendet wünsche der Entwickler oder Designer zu befriedigen.

Eine Vielzahl von nützlichen Assets lassen sich im Asset-Store finden. Diese sind jedoch größtenteils kostenpflichtig. Abschließend erfolgt auch hier der Verweis auf zukünftige Arbeiten, die sich genauer mit diesem Thema befassen.

4.5 Cowboy in Unity3D

Position in der Toolchain

Die Implementierung des Cowboys in Unity3D erfolgt wie in Abschnitt 4.2 beschrieben. Hierbei sollte die Machbarkeit der Übertragung von Blender-Modellen in Unity3D überprüft werden, vgl. Figure 17. Dabei soll zunächst der Cowboy importiert und mit einer Animation versehen werden.

Fig. 17. Zweiter Schritt: Unity3D

Unity3D Cowboy

In diesem Abschnitt wird die Implementierung des Cowboys in Unity3D beschrieben. Hierbei wurde der Cowboy zu der Szene hinzugefügt. Auch wenn Unity3D das Blender-Dateiformat unterstützt, wurde das Modell in Blender zunächst im FBX-Dateiformat exportiert. Dies ist durch die hinzugefügte Animation begründet. Statische Objekte können jedoch bedenkenlos im Blender-Dateiformat importiert werden.

Nach der Skalierung und Positionierung des Cowboys, und weiterer Objekte, wurde die Animation an das Cowboy-Objekt angehängt, vgl. Figure 18. Anschließend wurde der Animation-Controller hinzugefügt. Dieser beinhaltet alle Animationszustände von genau diesem Objekt. Jene sind direkt nach dem Starten der Anwendung aktiv und können entweder durch interne Ereignisse oder durch externe Einflüsse ausgelöst werden. Welche Aktion eine Animationen aktiviert, wird im Unity3D Animator definiert. In Figure 19 ist der Unity3D-Animator dargestellt. Jener kann auch mit einem Zustandsautomaten verglichen werden. Die in Figure 19

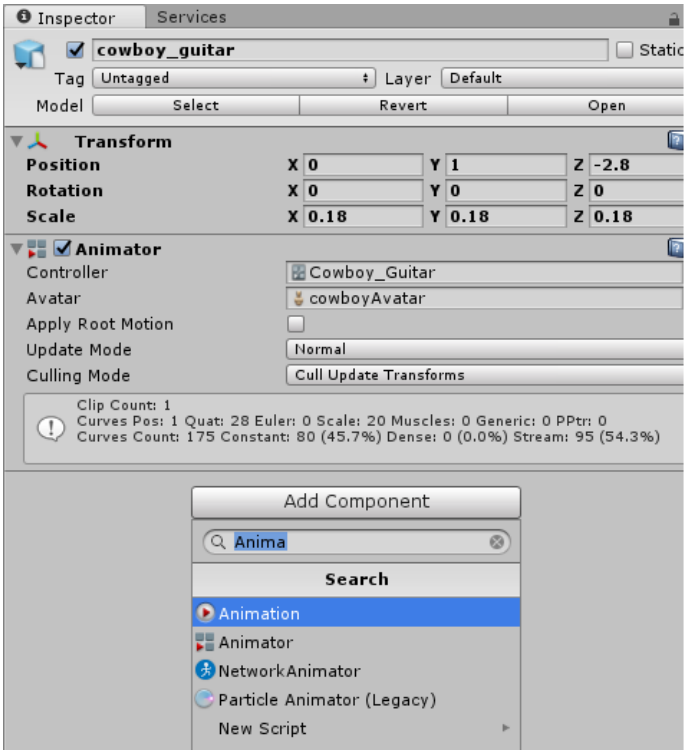

Fig. 18. Animation Komponente in Unity3D

konfigurierte Controller beinhaltet nur eine Animation, diese wird in Dauerschleife ausgeführt.

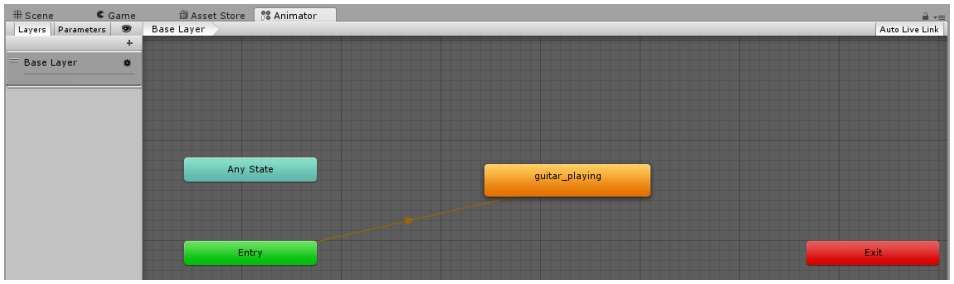

Fig. 19. Unity3D Animator

Abschließend wurden einige Effekte wie Partikel-Effekte und Licht hinzugefügt. Das Ergebnis ist in Figure 20 zu sehen.

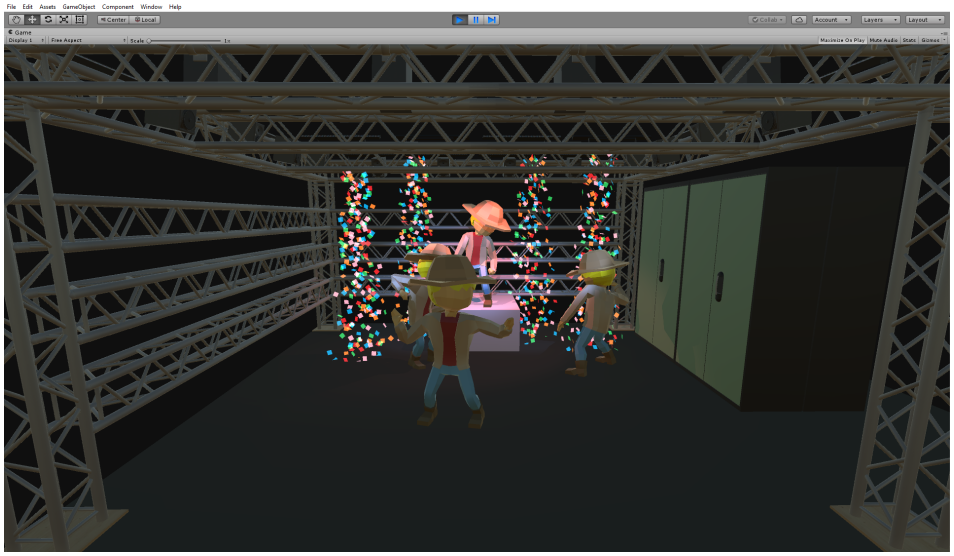

Fig. 20. Tanzende Cowboys

4.6 Fazit

Im vorangegangenen Abschnitt wurde das Cowboy-Modell in Unity3D importiert und somit die Machbarkeit bewiesen. Darüber hinaus konnte die Anwendung auf einem

Windowssystem gebaut und ausgeführt werden. Aufgrund des bereitgestellten Plugins von HTC und Unity3D, welches die HTC Vive unterstützt, konnte auch der letzte Schritt dieser Toolchain überprüft werden. Hierbei wird lediglich das Plugin als Asset zum Projekt hinzugefügt und in den Einstellungen von Unity3D die VR-Unterstützung aktiviert. Anschließend kann die selbe Anwendung in der virtuellen Welt gestartet und auf der HTC Vive dargestellt werden. Dies weist die Machbarkeit des letzten Schrittes nach, vgl. Figure 21.

Fig. 21. Letzter Schritt: HTC Vive/Virtual Reality

5 Fazit

Die vorangegangenen Kapitel haben die Machbarkeit der Toolchain schrittweise nachgewiesen. Dies ermöglicht einen erfolgreichen Projektabschluss. Des Weiteren konnte das Zusammenspiel der einzelnen Anwendungen beispielhaft dargestellt werden, sodass zukünftige Arbeiten auf den Erkenntnissen aufbauen können. Somit könnten Fehler vermieden und Zeit gespart werden.

5.1 Ausblick

Diese Ausarbeitung hat die Machbarkeit dieser Toolchain an einem einfachen Beispiel nachgewiesen. Das an diese Arbeit anschließende Hauptprojekt soll die Machbarkeit jener Toolchain auch bei größeren Projekten nachweisen. Darüber hinaus sollen die Entwicklung von Unity3D-Plugins und Assets näher betrachtet werden.

SpaceFlight

Hierfür soll eine Raumschiffssimulation entwickelt und getestet werden. Jene soll verschiedene Möglichkeiten der Interaktion ermöglichen und dafür unterschiedliche Technologien einsetzen. Für die Umsetzung des Projektes wird ein virtuelles Modell eines Cockpits inklusive Joystick zum steuern des Raumschiffs. Die genaue Planung erfolgt im anschließenden Projekt.

References

1. Amos, E.: Google-cardboard (2015), http://upload.wikimedia.org/wikipedia/commons/thumb/a/ad/Google-Cardboard.jpg/1165px-Google-Cardboard.jpg
2. Bronsch, J.: Vergleich von virtual-und augmented-reality in bezug auf deren gemeinsamkeiten und probleme. Tech. rep., Hochschule für Angewandte Wissenschaften Hamburg, http://users.informatik.haw-hamburg.de/ ubicomp/projekte/master2015-gsem/bronsch/bericht.pdf (2016)
3. Carvalho, C.A.S.: A theoretical inquiry into the role of enaction in cybertherapy. In: Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality. pp. 45–51. TEEM '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2808580.2808588
4. Cheng, L.P., Roumen, T., Rantzsch, H., Köhler, S., Schmidt, P., Kovacs, R., Jasper, J., Kemper, J., Baudisch, P.: Turkdeck: Physical virtual reality based on people. In: Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology. pp. 417–426. UIST '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2807442.2807463
5. CSTI: Csti-raumplan (2016), http://csti.haw-hamburg.de/projekt/
6. (DeSF), D.S.: Holodeck (2006), http://www.desf.de/images/Holodeck_6.gif
7. Dörner, R., Broll, W., Grimm, P., Jung, B.: Virtual und Augmented Reality (VR / AR). Springer Vieweg, 1 edn. (2013)
8. EpicGames: Unreal engine material editor (2017), http://docs.unrealengine.com/latest/images/Engine/Rendering/Materials/IntroductionToMaterials/WoodFloor.jpg
9. EpicGames: Unreal engine ui (2017), http://docs.unrealengine.com/latest/images/Engine/UI/InterfaceOverview/layout.jpg
10. Gu, X., Zhang, Y., Sun, W., Bian, Y., Zhou, D., Kristensson, P.O.: Dexmo: An inexpensive and lightweight mechanical exoskeleton for motion capture and force feedback in vr. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. pp. 1991–1995. CHI '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2858036.2858487
11. Hodgson, E., Bachmann, E., Waller, D.: Redirected walking to explore virtual environments: Assessing the potential for spatial interference. ACM Trans. Appl. Percept. 8(4), 22:1–22:22 (Dec 2008), http://doi.acm.org/10.1145/2043603.2043604
12. Jan, M.: Spatial auditory displays (2015), http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/martin.pdf
13. JMonkeyEngine: About jmonkeyengine (2017), http://jmonkeyengine.org/
14. Kip: Blocks demo for leap motion (2016), http://developer-archive.leapmotion.com/gallery/blocks
15. Kunz, M.: Datenmenge berechnen bei digitalen videodaten (2017), http://www.mediencommunity.de/content/datenmenge-berechnen-bei-digitalen-videodaten
16. Lague, S.: Blender character creation: Modelling (2015), http://www.youtube.com/watch?v=DiIoWrOllRw
17. Ogre: About ogre (2017), http://www.ogre3d.org/about
18. RiftInfo: Comparison of development kit 1 and development kit 2 of oculus rift (2016), http://riftinfo.com/oculus-rift-specs-dk1-vs-dk2-comparison
19. Sutherland, I.: The ulimate display. pp. 506–508. IFIP Congress (1965), http://worrydream.com/refs/Sutherland%20-%20The%20Ultimate%20Display.pdf
20. Sutherland, I.: Sutherland hmd (2016), http://resumbrae.com/ub/dms423_f08/06/sutherland-hmd.jpg
21. Systems, C.: Cyberglove (2017), http://www.cyberglovesystems.com/cybergrasp/
22. VRNERDS: Die geschichte der virtuellen realität (2017), http://www.vrnerds.de/die-geschichte-der-virtuellen-realitaet/

23. VRNERDS: Htc vive 2: 4k display und ohne kabel (2017), http://www.vrnerds.de/htc-vive-2-4k-display-und-ohne-kabel/
24. VRODE: Vr-brillen im technikcheck (2017), http://vrodo.de/vr-brillen-vergleich/
25. Wikipedia: Replikator (2016), http://de.wikipedia.org/wiki/Star-Trek-Technologie#Replikator
26. Wortmann, H.: Iterative konstruktion eines individualisierten force-feedback handschuhs mittels rapid-prototyping-techniken (2017), http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/wortmann.pdf
27. Wortmann, N.: Acoustic redirected walking with auditory cues by means of wave field synthesis (2015), http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/nogalski.pdf
28. Yu, S.: Ieee 802.11ad specification (2017), http://standards.ieee.org/news/2013/802.11ad.html