Fachhochschule für Ökonomie und Management

Studienzentrum Berlin; Bismarckstr: 107, 10625 Berlin
Bachelor-Studiengang Wirtschaftsinformatik

Anwendung von Machinima-Methoden für die Datenreduktion beim Storytelling auf einer Mikrokonsole

verfasst und eingereicht von

Folker Linstedt

Bachelor-Thesis zur Erlangung des Grades Bachelor of Science – B.Sc.

betreut von Prof. Dr. Ralf Hötling

Matrikelnummer: 162081

Abgabedatum: 10.01.2019

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D, 3D	zwei-, dreidimensional
AHX	Hively Tracker Dateiformat
ASTC	Adaptive Scalable Texture Compression
AVI-RIFF	Audio Video Interleave-Resource Interchange File Format
BMP	Windows Bitmap
DDS	DirectX Texturformat mit DXT-Verfahren komprimiert
DVB-T	Digital Video Broadcasting Terrestrisch
DXT	S3 Texture Compression
ETC	Ericsson Texture Compression
FPSCC	First Person Shooter Creator Classic
HVL	Hively Tracker Dateiformat
JPEG	Bezeichnung für die 1992 vorgestellte Norm ISO/IEC 10918-1 bzw. CCITT Recommendation T.81, namensgebend war die Joint Photographic Experts Group
MP3	MPEG Audio Layer III
MP4	MPEG-4 Teil 12 und 14
MPEG	Moving Picture Experts Group
OBJ	Dateiformat für 3D-Objekte, Wavefront OBJ
OpenGL ES	Open Graphics Library for Embedded Systems
OpenGL	Open Graphics Library
PC	Personal Computer
PCM	Puls-Code-Modulation
PNG	Portable Network Graphics
SoC	System on a Chip
Tegra 3	Bezeichnung für SoC NVIDIA
VoD	Video On Demand
Vulkan	Next Generation OpenGL oder glNext, ‚Nachfolger von OpenGL
X	DirectX-Objekt-Format

1 Einleitung

In dieser Arbeit wird die Anwendung von Machinima-Methoden zur Datenreduktion beim Storytelling auf einer Mikrokonsole betrachtet. Als Machinima werden Filme bezeichnet, welche mit Game Engines erzeugt werden. Der Ausblick wird speziell auf das Storytelling gelegt, da hier in der Anwendung spezielle Anforderungen an Hardware und Software gestellt werden können, welche eine nähere Betrachtung erfordern.

Nach der Definition der in dieser Arbeit verwendeten Begrifflichkeiten wie Storytelling, Machinima wird anschließend wie folgt vorgegangen. Zunächst wird beschrieben, was kennzeichnend für Machinima-Methoden ist und inwieweit Machinima-Methoden bereits jenseits von Machinima selbst eingesetzt werden und in welchen Anwendungsfällen diese auch zu einer Datenreduktion führen können.

Anschließend wird der Einblick in die Datenreduktion vertieft. Übliche Kompressionsverfahren werden kurz erläutert und die mögliche Datenreduktion durch Machinimamethoden wird eingehend betrachtet. Insbesondere die gängige Videokompression für den sogenannten Stream wird im Vergleich zur Datenreduktion durch die Machinima-Methode untersucht.

Da der Begriff Machinima aus dem Bereich der Computerspiele stammt, welche um das Jahr 2000 herum vorzugsweise auf PC-Hardware gespielt wurden, wird weiterhin analysiert, wie Machinima-Methoden zur Datenreduktion auf Nicht-PC-Hardware, im Speziellen der Mikrokonsole, angewendet werden können. Die Unterscheidung zwischen mobilen Smartgeräten und stationären Mikrokonsolen dient der Einschränkung, da der Markt eine zu hohe Zahl an unterschiedlichen Geräten aufweist und nicht jede Kategorie berücksichtigt werden kann, auch wenn mögliche Gemeinsamkeiten vorhanden sein können.

Für den besseren Lesefluss wird auf eine Genderspezifizierung verzichtet. Als Anwender sind alle Personen gemeint, ungeachtet des Geschlechts. Ebenso ist „der Anwender“, „der Nutzer“ oder ähnliche Formulierungen in Bezug auf alle Geschlechter zu sehen.

In einigen Fällen wird bewusst der englische Begriff statt der deutschen Übersetzung zur Unterscheidung zwischen den unterschiedlichen Bedeutungen eingesetzt. In anderen Fällen sind deutsche und englische Bezeichnung entweder gleichermaßen vertreten oder als deutsches Wort nicht in Gebrauch.

Der Begriff „Spiel“ wird in dieser Arbeit möglicherweise für den Leser in einem anderen Zusammenhang genutzt als dem sonst üblichen. Zu diesem Zweck wird er an dieser Stelle näher beschrieben: In dieser Arbeit wird Spiel als Videospiel bzw. Computerspiel betrachtet. Game, Game Engine, Spieleumgebung, Netzwerk-Mehrspieler-Spiel und ähnliche Begriffe beziehen sich auf Spiele, welche an einem elektronischen Gerät gespielt werden können. Computer, Tablet, Smartphone, Mikrokonsole und andere Geräte können zum Spielen benötigt werden. Darüber hinaus ist ein Bildschirm oder anderes Anzeigegerät notwendig, wenn dies noch nicht im Gerät mit verbaut ist. Zum Spielen werden in der Regel Eingabegeräte benötigt. Dies können z.B. sein Maus, Tastatur, Gamepad, Joystick, digitaler Stift, eine Fernbedienung. Denkbar sind auch Spracheingabe oder durch eine Kamera aufgenommene Gesten oder aber Bewegungen des Spielers selbst. Bei einem Smartphone oder Tablet Erfüllen Finger in Kombination mit einem entsprechend ausgestatteten Display die Funktion eines Eingabegerätes.

Diese Seminararbeit richtet sich nach dem offiziellen „Leitfaden zur formalen Gestaltung von Seminar- und Abschlussarbeiten“ der FOM mit dem Stand vom 20.12.2016.

2 Storytelling mit Machinima-Methoden

Das Storytelling mittels Machinima-Methoden ist ein wesentlicher Bestandteil dieser Arbeit. In der Vergangenheit fanden Machinimas hauptsächlich in den Medien- und Filmwissenschaften Beachtung, in der Informatik wurden sie bisher kaum berücksichtigt. Vor einer Betrachtung der Machinima-Methoden zur Datenreduktion ist es jedoch erforderlich zunächst die Begrifflichkeiten zu klären. Welche Art von Storytelling wird berücksichtigt, was ist ein Machinima, wie unterscheidet es sich von anderen Medien und was ist letztendlich für eine Machinima-Methoden kennzeichnend, sind die zentralen Fragestellungen in diesem Kapitel.

2.1 Storytelling in Filmen und Computerspielen

Storytelling findet in verschiedenen Medien zu unterschiedlichsten Zwecken statt, um dies einzugrenzen, wird im weiteren Verlauf nur auf das wörtlich übersetzte „Geschichtenerzählen“ Bezug genommen. Zu welchen anderen Zwecken das Storytelling genutzt werden kann, ist in dieser Arbeit nicht relevant.[1] Diese Arbeit bezieht sich auf den durch Machinima-Methoden erstellten Film als Ausgangspräsentationsformat einer Geschichte. Auf andere Formen des Storytellings wie z.B. Literatur wird nicht eingegangen.

Computerspiele werden mittlerweile neben dem Film als eigenständige Kunstform anerkannt. Durch das Verschmelzen von Computerspielen und Methoden aus dem Film und umgekehrt, wachsen beide Technologien immer mehr zusammen. Zuletzt wurde dem Wechselspiel von Computerspielen und Filmen in einer Ausstellung des Deutschen Filmmuseums in Frankfurt am Main Rechnung getragen.[2] Der Film ist im Gegensatz zum Computerspiel nicht interaktiv. Er verändert seinen Ausgang nicht, die Geschichte entfaltet sich linear von vorn nach hinten. Selbst durch Rückblenden, Traumsequenzen und Schnitte, wird doch der Film als solcher von vorn nach hinten konsumiert. Computerspiele haben dagegen oft eine verzweigte Narration und somit eine nichtlineare Erzählstruktur.[3] Ein dramaturgisches Werkzeug, um die Handlung

[1] Storytelling beschreibt im Marketing eine Kommunikationsmethode zur Vermittlung von Informationen, Wissen, Werten, Meinungen etc. Dabei werden nicht emotionale Inhalte in Geschichten verpackt, um über die Geschichte Emotionen und Interesse bei Zuhörern, Lesern oder Betrachtern zu wecken. Vgl. *Online-Marketing-Praxis*, Storytelling, o.J.
[2] Vgl. *Deutsches Filmmuseum* (Hrsg.), Film und Games, 2015.
[3] Vgl. *Müller-Michaelis, J.*, Computerspiel Erzählform, 2006.

in Computerspielen voranzutreiben, ist die sogenannte „Cutszene“. In ihr werden wichtige Ereignisse des Spiels als filmische Zwischensequenz erzählt.[4] Im Gegensatz zur c*ut-scene* ist das Machinima nicht fest im Spiel integriert, sondern existiert außerhalb des Spiels als eigenständiges Medium. Hierzu soll im Folgenden genauer erläutert werden, was ein Machinima ausmacht und wie es sich zu verwandten Genres abgrenzt.

2.2 Was ist Machinima?

Machinima selbst ist ein Kunstwort, welches die englischen Wörter *machine, animation* und *cinema* miteinander verbindet und seit den 2000ern als Fachterminus etabliert ist.[5]

Das Phänomen Machinima wird in der Medienwissenschaft folgendermaßen definiert: „*Machinimas sind Filme, die in einer Echtzeit-3D-Umgebung produziert werden.*“[6] und „*Machinimas sind Filme, die mittels Echtzeit-3D-Spielen produziert werden.*“[7] Die aktuelle Filmwissenschaft grenzt den Begriff sogar noch stärker ein: „*Machinimas heißen die Filme, die mit beziehungsweise in digitalen Spielen in Echtzeit aufgezeichnet und im Anschluss mit einem Schnittprogramm bearbeitet werden.*“[8]

In dieser Arbeit wird Machinima wie folgt definiert: Machinimas sind aus Computerspielen aufgenommene Szenen zusammengestellt zu eigenständigen Filmen mit neuer Handlung, losgelöst vom Computerspiel und dessen Handlungsverlauf. Aus der Echtzeit Computerspieleaufzeichnung mit Game Engine entsteht eine neue Geschichte und somit ein Mehrwert. Eine Bearbeitung mit Videoschnittprogramm ist nicht zwingend notwendig, obwohl bei der Machinima-Erstellung in der Regel der Spielverlauf durch spezielle Software, so genannte Screencasting-Programme aufgezeichnet wird und anschließend diese Aufnahme mit einem Videoschnittprogramm

[4] Dieses Verfahren wurde erstmals von Ron Gilbert in Maniac Mansion eingeführt und gehört seitdem zum festen Repertoire der Erzähltechnik in Computerspielen, vgl. *Gilbert, R*., Maniac, 2011.
[5] Vgl. *Rodewald, V. M.,* Machinima, 2015, S. 195.
[6] *Schmitt, L*., Machinima, 2006, S. 18.
[7] *Schmitt, L*., Machinima, 2006, S. 19.
[8] *Rodewald, V. M.,* Machinima, 2015, S. 195.

nachbearbeitet wird. Allerdings können zur Erstellung des Machinimas auch Modifikationen am Spiel vorgenommen werden. Als Machinima werden Filme bezeichnet, welche mit Game Engines erzeugt werden.[9]

Das Machinima ist somit immer stark an die Vorgaben aus dem Spiel gebunden. Eines der bekanntesten Machinima Beispiele ist „*Red Vs Blue*“, welches sogar auf der Streaming-Plattform Netflix einer breiten Öffentlichkeit zugänglich gemacht wird. Bisher sind Machinima jedoch kein Mainstream geworden, vielmehr handelt es sich von Computerspielefans für Computerspielefans erstellte Filme. Allerdings werden Machinima-Methoden zur Anfertigung von Filmen des Öfteren Programm von Hochschulen und inzwischen sogar von Schulen,[10] weil es eine kostengünstige Methode darstellt, eigene filmische Werke zu produzieren. Der Vorteil von Machinima als pädagogisches Mittel ist, dass die Einstiegshürde sehr gering ist.

2.3 Verwandte Genres

Da mit Machinimas eine neue Geschichte erzählt wird, können sie von Demos, Let’s Plays und Walk Throughs abgegrenzt werden.

Die Demos sind datenreduzierte Animationen, welche aus den Crack-Intros, auch Cracktros genannt, entstanden sind und in ihrem Ursprung somit dem Machinima verwandt sind. „*Wenn Hacker den Kopierschutz eines kommerziellen Spiels deaktivierten und es in Umlauf brachten, dann fügten sie ihm zuvor oftmals ein Crack-Intro hinzu, um gegenüber anderen Spielern ihre Hackerfähigkeiten zu demonstrieren. Die Erstellung dieser Crack-Intros (…) wurde zu einem eigenen künstlerischen Wettbewerb und beeinflusste die Ursprünge der heutigen Demoszene*.“[11] In der Demoszene geht es darum, möglichst aufwendige audiovisuelle Animationen mit dem effizientesten Code zu erschaffen. Die Datenreduktion ist somit das Hauptmerkmal des Demos. Eines der bekanntesten Beispiele des Genres ist das Demo *debris* (Farbrausch 2007), welches nur 117 KB groß ist und somit die Größe der meisten Einzeltexturen bei Echtzeit-Renderings unterschreitet.[12] Bei Machinima wird der Code der Game-Engine im Unterschied zum Crack-Intro meistens nicht geknackt. Das Crack-Intro ist

[9] Vgl. *Nitsche, M*., Potenzial Machinima, 2015, S. 106-113.
[10] Vgl. *Rodewald, V. M.,* Machinima, 2015, S. 196-197.
[11] *Nitsche, F.,* Potenzial Machinima, 2015, S. 109.
[12] Vgl. *Nitsche, F.,* Potenzial Machinima, 2015, S. 110.

meistens nur eine kurze Animation des Hacker-Logos, während das Machinima einen eigenen Film mit Handlung darstellt.[13] Im Gegensatz zu Machinimas können Demos nur von hochspezialisierten Programmierern erstellt werden, während zur Erstellung von Machinimas in der Regel keine Spezialkenntnisse notwendig sind.

In den Walk Throughs bzw. Strategy Guides sowie Speed Runs kann man einen anderen Ursprung des Machinimas sehen. Schmitt erklärt den Speed Run so: *„Ein Spiel so schnell wie möglich durchspielen. Wobei cheaten (mogeln) nicht erlaubt ist. Die ersten Speedruns wurden mit der Demo-Funktion von Doom[14] aufgenommen. Besonders populär wurden sie mit Quake*."[15] Der optimierte Durchlauf zur Demonstration der Beherrschung des Spiels wurde zum Urmachinima: Spielen wurde zum Filminhalt.[16] Im Gegensatz zum Let's Play und zum Machinima darf beim Speed Run keinerlei Modifikation am Ausgangsspiel vorgenommen werden. Das Machinima ist eine Weiterentwicklung des Speed Runs. Es steht nicht mehr die Optimierung des Spielens im Mittelpunkt, sondern eigene Inhalte können vermittelt werden. Walk Throughs unterscheiden sich vom Speed Run dahingehend, dass der Hauptinhalt die Anleitung des rezipierenden Spielers ist und nicht die reine Demonstration des eigenen Könnens. Das Walk Through ist eine medial andere Umsetzung der ehemals in Printmedien abgedruckten Komplettlösung.

Let's Plays sind technisch identisch mit Machinimas, Walk Throughs und Speed Runs, unterscheiden sich jedoch inhaltlich.[17] „*Sinn eines Let's Plays ist (…), dem Zuschauer ein Erleben des Spiels zu ermöglichen. Durch das Voice-over, den Einsatz einer Kommentarstimme, geht es aber darüber hinaus. Das Voice-over (…) bildet einen eigenen Unterhaltungswert.*"[18]

2.4 Was sind Machinima-Methoden?

Im Folgenden soll erläutert werden, welche speziellen Methoden bei der Erstellung eines Machinima-Filmes Anwendung finden können und welche Bedingungen erfüllt

[13] Vgl. *Schmitt, L.,* Machinima, 2006, S. 10-11.
[14] Bei der Demofunktion von Doom können die einzelnen Bewegungsabläufe der Protagonisten „aufgezeichnet" werden, vgl. Kapitel 2.4.1 *Das Replay als Machinima-Methode*.
[15] *Schmitt, L.,* Machinima, 2006, S. 13.
[16] Vgl. *Nitsche, F.,* Potenzial Machinima, 2015, S. 110.
[17] Für ausführlichere Informationen zum Inhalt von Let's Plays, siehe *Klein, T*., Let's-Play-Videos, 2015, S. 199-203.
[18] *Klein, T*., Let's-Play-Videos, 2015, S. 199-200.

sein müssen. Ein Machinima wird mit Hilfe einer Game Engine, welche das Echtzeit-Rendern, also das Real-Time-Rendering, beherrscht, erstellt. Anders ausgedrückt, während der Laufzeit werden audiovisuelle Inhalte auf einem Anzeigegerät mit Hilfe von Grafik- und Sound-Hardware durch entsprechenden Programmcode dargestellt. Dieser Programmcode kann als Game Engine bezeichnet werden. Da die Game Engine für die Erstellung eines Machinimas essentiell ist, wird im Folgenden die Game Engine in ihrer Bedeutung näher erläutert.

2.4.1 Echtzeit-Rendering mit einer Game Engine

In einer Game Engine angewendete Verfahren zur Darstellung von 2D- und 3D-Objekten und weiteren visuellen Eindrücken sind von der Hardware in der Regel unabhängig angelegt. Beschreibungssprachen für eine solche Darstellung sind z.B. Standards wie OpenGL, DirectX und Vulkan. Auf mögliche Einschränkungen für bestimmte Hardware wird im weiteren Verlauf der Arbeit eingegangen.

Als Game Engine wird einmal der Echtzeit-Renderer bezeichnet, welcher mit dem Spiel zusammen an den Verbraucher ausgeliefert wird. Das fertige Spiel sozusagen, welches auf die weiteren Medien-Inhalte zur Darstellung und Vertonung zugreift. Auf der anderen Seite wird auch das „SDK“, als Game Engine bezeichnet. Das Framework mit seinen Bibliotheken und den Tools zur Erschaffung virtueller Welten und Optimierung von Texturen und 3D-Objekten. Diese Form der Game Engine kann sich stark in seinen Werkzeugen unterscheiden, deshalb besteht in dieser Arbeit kein Anspruch auf Vollständigkeit, welche Hilfsprogramme und Skripte oder Editoren und andere Module eine Game Engine im Sinne einer Software-Entwicklungs-Umgebung ausmachen müssen. Game Engines wachsen und verändern sich von Spiel zu Spiel. Es werden Erweiterungen im Verlauf der Entwicklung an Spielen hinzugefügt oder entfernt. Eine Game Engine und vor allem das „Game Engine SDK“ verändern sich ähnlich einer komplexen Warenwirtschaftssoftware, die sich an bestehende oder in Kraft getretene Gesetze anpassen muss oder an weitere Anforderungen. Eine klare Trennung zwischen Framework und SDK wird in dieser Arbeit nicht vorgenommen. Zur Vereinfachung wird die Game Engine im Sinne eines Programmes zur Erstellung des Spiels als Game Engine SDK bezeichnet und die Game Engine, welche im fertigen Spiel verwendet wird, als reine Game Engine. Weil sie quasi den „Motor“ des Spiels darstellt und dafür sorgt, dass es „läuft“. Die Game Engine ist im Vergleich

zum Brettspiel der Spielleiter, welcher sich an die Spielregeln hält und diese umsetzt.[19] Beispiele für angebotene Game Engine und Spiele-Frameworks sind z.B.: Unreal Engine, Unity Engine, App Game Kit, Game Maker Studio, Godot usw. usf.

Die aktuelle Versionsnummer variiert je nach Update des Game Engine SDKs. Fertige Spiele sind in der Regel von einer Versionsänderung nicht betroffen. Aktuelle Versionen sind im Internet auf den entsprechenden Seiten der Engine-Entwickler zu finden. Für ein Projekt muss eine Game Engine nicht lizenziert werden, sie kann auch Inhouse entstehen.

Für die Nutzung einer Game Engine können Lizenzgebühren anfallen. Meist betrifft dies vor allem die damit erstellten Spiele. Dies kann aber auch Teile der Engine betreffen, wenn dieser lizenzbedürftige Code verwenden. Zusatzmodule oder auch Teile für Physik und zum Abspielen von Audiodaten können unabhängig von der Game Engine betroffen sein. Hier sei der Unterschied zwischen App Game Kit und Game Maker Studio im Vergleich zu Unity, aber auch Unreal-Engine erwähnt. App Game Kit wird einmalig bezahlt, also mit einer einmaligen Lizenzgebühr ist die Nutzung unbefristet abgegolten. Bei Game Maker Studio ist dies ähnlich, sofern das damit erstellte Spiel nicht auf einer Videospiele-Konsole wie Xbox, Playstation oder Nintendo Switch ausgeführt werden soll. Bei Unreal Engine ist hingegen der mit dem Spiel eingenommene Verdienst ausschlaggebend, ob und in welcher Höhe mögliche Lizenzgebühren anfallen. Die Unity Game Engine bietet indes eine Art Mietmodell, wobei hier die Nutzung des Game Engine SDKs „vermietet“ wird. Eine genauere Betrachtung der Lizenzen erfolgt an dieser Stelle nicht. Nur muss dies eventuell bei der Erstellung von Anwendungen, in erster Linie von Spielen, mitberücksichtigt werden, da das Lizenzmodell eventuelle Folgekosten mit sich bringen. Vor allem bei dem Vertrieb, inklusive Verbreitung und der kommerziellen Ausnutzung der verwendeten Game Engine kann dies von erheblicher Bedeutung sein.

2.4.2 Verwendung von Assets

Als Assets werden alle digitalen Inhalte von Computerspielen bezeichnet, die zur Erstellung des Spiels benötigt. Darunter fallen Texturen, Audiodaten, Materialien, Skripte und auch 3D-Modelle.[20] Die Verwendung vorgefertigter Assets aus Spielen

[19] Vgl. *Post, U.,* Android, 2015, S. 141.
[20] Vgl. *Seifert, C., Wislaug, J.,* Unity, 2017, S. 31.

macht den Charme des Machinimas aus und ermöglicht eben auch diesen leichten Einstieg in die Filmproduktion, da nicht erst aufwändig eigene Assets in komplexen 3D-Modellierungsprogrammen wie 3ds Max oder Blender erstellt werden müssen. Assets bilden einen Teil der im weiteren Verlauf dargestellten Datenreduktion. Aus diesem Grund kann die Lizensierung von Assets nicht unberücksichtigt bleiben. Im Internet werden auf diversen Seiten kostenlose Assets angeboten, doch kann die Nutzung als solche später zu Komplikationen führen. Markenrechte könnten zum Beispiel verletzt werden. Oder aber der ursprüngliche Upload auf eine entsprechende Plattform für freien Inhalt war schon unberechtigt. Hier müsste jedes Objekt selbst auf seine Linzens geprüft werden.

Eine solche Webseite wäre z.B. die Seite poly.google.com von Google. Welche im Kern grundlegend zu begrüßen ist, jedoch sich selbst unbrauchbar macht, durch die Möglichkeit, urheberrechtlich geschützte Materialien als fälschlich gemeinfrei zu deklarieren. Die Einhaltung der Lizenzbedingungen muss daher auch bei der Erstellung einer möglichen Kompressions-Anwendung mit Machinima-Methoden berücksichtigt werden, da im Machinima die Assets eng mit dem resultierenden Werk verknüpft sind. Unabhängig von dem möglichen Eintreten des Vorschlags der EU-Kommission zur EU-Urheberrechtsreform speziell bezogen auf den Artikel 13, ist es wichtig, dass die Software selbst in ihrem Auslieferungszustand keinen Anlass für eine Urheberrechtsverletzung liefert.

Die Wichtigkeit der Einhaltung des Urheberrechts wird im Abschnitt zur Kompression verdeutlicht.

2.4.3 Replay-Aufzeichnungen

In dieser Arbeit wird ein Replay als Aufzeichnung von Abläufen in einem Videospiel betrachtet. Der Begriff „Aufzeichnung“ im Zusammenhang mit einem Video meint die Abspeicherung von komprimierten Pixeldaten wie sie in einem Video-Stream zu finden sind. Der Begriff „Aufzeichnung“ im Zusammenhang mit einem Replay meint die Abspeicherung von Spieledaten, eventuell Tasteneingaben, anderen sich verändernden Daten, ähnlich den Metadaten von Dokumenten, jedoch nicht die Aufzeichnung von komprimierten Pixeldaten. Der Begriff Replay wird bewusst genutzt, um es von „Videoaufzeichnung“ und dem „Videobeweis“ zu unterscheiden.

Die Möglichkeit der Aufzeichnung von sogenanntem Gameplay wird in Computerspielen zum Beispiel genutzt, um musterhaft Fehler im Ablauf zu protokollieren, die Bewegungen des Spielers nachzuvollziehen oder ganze Spielerunden später erneut wiedergeben zu können. Bei Rennspielen, Autorennen, etc. kann die Möglichkeit zur Speicherung von sogenannten Replays bestehen oder aber das Herausfordern einer Aufzeichnung zur Steigerung der eigenen Spieleleistung genutzt werden. Oft wird dies als „*Schatten*“ bezeichnet und es besteht die Möglichkeit, gegen einen früheren Ablauf des Fahrens auf einer Rennstrecke gegen sich selbst aus der Vergangenheit anzutreten. Im Bereich des eSports, also dem Wettkampf im größeren Maßstab mit Computerspielen, werden Matches, also Spiele oder Spielerunden, aufgezeichnet und können später analysiert werden. Ein Vertreter ist z.B. StarCraft 2. Die Aufzeichnungen werden auch hier als Replays bezeichnet.

Abbildung 1: Bildschirmaufzeichnung Installation Battle Net

Diese Replays benötigen das eigentliche Spiel selbst wieder zum Abspielen. Und es kann als eine Art Aufzeichnung der Eingaben während des Spielens betrachtet werden, ähnlich einem Keylogger. Inwieweit Eingaben protokolliert werden oder aber die Zustände von spielrelevanten Variablen selbst abgespeichert werden, kann auf Grund der Menge an unterschiedlichen Spielen und Spielemechaniken nicht vereinheitlicht werden. Als Gemeinsamkeit kann aber die geringe Datenmenge eines solchen Replays angenommen werden.

Diese Form der Kompression hat selbstverständlich Einschränkungen. Die Animationen und Spielszenen sind auf das Spiel und deren Inhalt beschränkt. Darüber hin-

aus muss jeder, der sich diese Art der Aufzeichnung anschauen will, das Spiel erworben haben. Selbst wenn der Erwerb möglicherweise finanziell kostenlos sein kann, sind dennoch die Kosten des Speicherplatzbedarfs zu berücksichtigen. 30 GByte werden bei der Installation laut Installationsbildschirm von StarCraft 2 veranschlagt. Damit ist der erste Film, welcher mit dieser Methode geschaut werden kann in seinem Datenverbrauch ~ 30 GByte groß. (Dateigröße des eigentlichen Films kann in dieser Ausprägung vernachlässigt werden). Der „Player“ wäre also 30 GByte groß, was bei einer Mikrokonsole mit ~8 GB Flashspeicher zu viel wäre. Darüber hinaus wären herkömmliche Streams wie sie auf Netflix oder YouTube zu finden sind bei gleicher Datenmenge über mehrere Stunden Spieldauer lang, oder sogar Tage, wenn man die 300 MB pro Stunde in SD Qualität von Netflix ansetzt.[21]

2.4.4 Machinima-Methoden in Netzwerk-Mehrspieler-Spielen

Eine ähnliche Umsetzung, wenn nicht sogar in einigen Fällen, dieselbe, ist in Multiplayerspielen zu finden, welche über Netzwerke funktionieren. (StarCraft 2 ist auch ein Netzwerk-Mehrspiele-Spiel) In regelmäßigen Zeitabständen werden hierbei mit den Mitspielern Veränderungen im Spielverlauf im Server-Client-Prinzip ausgetauscht. Jeder im Netzwerk-Spiel befindliche Spieler teilt via Client seine Eingaben mit und diese werden im Server auf möglichen Betrug abgeglichen und dann an die anderen Spieler nach Berechnung der resultierenden Veränderungen weitergeleitet.

Die detaillierte Funktionsweise von Netzwerk-Mehrspieler-Spielen wird in dieser Arbeit nicht tiefergehend behandelt. Es sollen lediglich Parallelen in der Funktionsweise dargelegt werden. Auch in Hinblick auf das Machinima, welches seinen Ursprung durch Netzwerk-Mehrspieler-Spiele erfahren hat.

Als aktuell relevante Ausprägung aus unterschiedlichen technischen und finanziellen Gründen kann das Videospiel „Fortnite“ von Epic Games angesehen werden. Der finanzielle Erfolg des Spiels wird in dieser Arbeit nicht betrachtet, jedoch ist zu erwähnen, dass technische Merkmale den finanziellen Erfolg begünstigt haben werden. Fortnite ist ein kostenloses Netzwerk-Mehrspieler-Spiel, welches auf unterschiedlichen Plattformen zur Verfügung steht.

---

[21] Beispiel einer Replay-Datei für StarCraft 2 https://lotv.spawningtool.com/replays/ 35:10 (mm:ss) Wiedergabezeit bei 213 kByte Dateigröße.

Merkmale des Videospiels Fortnite sind z.B. der leichte Einstieg in das Spielgeschehen und der vergleichsweise geringe Speicherplatzbedarf auf dem Spielgerät. Durch den Unterhaltungswert des Spiels sind auch Zuschauer motiviert, einfach nur dem Spielgeschehen ohne eigene Interaktion beizuwohnen.

Dies können Zuschauer auf zwei unterschiedliche Weisen. Einmal durch das Spielestreaming, welches durch einen Spieler selbst angeboten werden muss und einmal als Spectator, nach dem Ausscheiden aus einer Spielrunde.

Im zweiten Fall als Spectator, also als Zuschauer, benötigt der Zuschauer des Spielgeschehens selbst das Spiel und erhält nur wie sonst auch die Bewegungsdaten und andere benötigte Spieledaten des Spiels ähnlich wie bei Replays und kann so einem anderen Spieler über seine Schultern schauen. Der Datenstrom ist sehr gering, allerdings wird das Spiel selbst benötigt. Beim Spielestreaming zeichnet der Spieler seine Bildschirmpixeldaten als Stream auf und sendet diese an eine Plattform, welche diese dann weiterverbreitet. Hier unterscheiden sich Streaming und Replay grundlegend. Auf diesen Unterschied wird im weiteren Verlauf beim Vergleich der Kompressionsmethoden näher eingegangen.

2.4.5 Animationsfilmproduktion mittels Machinima-Methode

Für den Animationsfilm „Toystory“ von Pixar aus dem Jahr 1995 wurden noch Renderfarmen, also ein Computerverbund zur Berechnung von Filmdaten, verwendet. Die Berechnung eines einzelnen Bildes dauerte trotz des Verbundes noch mehrere Stunden. “*114,240 – Frames of animation in the final film, requiring 800,000 machine hours to render at 2-15 hours per frame.”*[22]

Im Jahr 2018 wird die Animationsserie „ZAFARI“ von Dreamworks mit der Unreal Game Engine in mit Hilfe von Echtzeit-Technologie erstellt. ZAFARI wäre damit ein Machinima-Film, da dieser mit einer Spiele-Entwicklungsumgebung erstellt wird. Allerdings hat diese Produktion nicht den Anspruch, auch Daten zu reduzieren, sondern die Darstellungsqualität auf ein qualitativ hohes Maß zu bringen. Die verwendeten Assets werden sicherlich durchaus mehrere Gigabyte an Speicherplatz allein benötigen. „*Producing feature film quality work at episodic television speeds and*

[22] *Bettinger, B.*, Pixar, 2012.

budgets is no simple task, but Digital Dimension was able to hit the mark with the help of Unreal Engine's real-time rendering technology"[23]

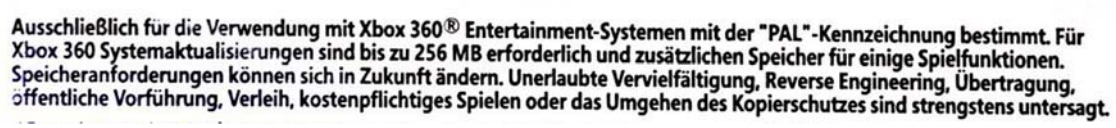
Das Echtzeit-Rendering mittels einer Game-Engine ist somit möglich, ohne größere Einbußen der visuellen Qualität hinnehmen zu müssen. Paradoxerweise verliert der Begriff Machinima durch Echtzeit-Rendering im Bereich des Films und der Animation immer mehr an Bedeutung, weil mittlerweile immer mehr Animationsproduktionen zum Echtzeit-Rendering tendieren und es kein Alleinstellungsmerkmal der Machinimas mehr darstellt.

Zusammenfassend kann gesagt werden, dass sich Machinima-Methoden zusammensetzen aus dem Echtzeit-Rendering mittels einer Game Engine sowie der Verwendung der bereits im Computerspiel enthaltenen Assets sowie dem Replay-Aufzeichnungsverfahren.

2.5 Exkurs: Problem des Urheberrechts bei Machinima-Methoden

Die sich in einem Spiel befindlichen Assets können Lizenzbedingungen unterliegen oder Markenschutzrechte enthalten. Das Abfilmen von Spieleinhalten kann untersagt sein und die unerlaubte Nutzung von Assets kann rechtlich geahndet werden. Bei Let's-Plays und Machinimas und ähnlichen Formen des Abfilmens wird dies oft geduldet, ist jedoch meist nicht explizit gestattet worden.

Abbildung 2: Hinweis auf der Verpackung von Minecraft Story-Mode, Xbox 360



Vor allem beim Machinima-Film kann dies für die weitere Verwendung zu Komplikationen führen. Aber auch die Erschaffung neuer Spiele mit Game Assets, welche Markenrechte verletzten ist problematisch.

Im Storytelling geht es rein aus wirtschaftlicher Sicht betrachtet immer um den Verkauf lizensierter Medien. Markenrechte an Titeln, Logos, Charakteren, und an anderen Dingen aus Universen wie Marvel, Disney, aber auch Harry Potter oder Star Wars sind denkbar. Diese Liste erhebt keinen Anspruch auf Vollständigkeit. Es soll

[23] *Pimentel, K*., ZAFARI, 2018.

verdeutlicht werden, dass Markenschutzrechte einigen Teilen eines Machinimas anheften können. Um dies zu vermeiden, müssten die verwendeten Assets in der genutzten Game Engine ebenfalls frei von Rechten sein oder eben dem Ersteller des Machinimas das entsprechende Recht zur Verwendung eingeräumt werden.

3 Methoden der Datenreduktion

In dieser Arbeit wird der Film als Ausgangsmedium in erster Linie technisch betrachtet. Damit eine mögliche Datenreduktion veranschaulicht werden kann, werden die Methoden zur üblichen Kompression von Filmmaterial beschrieben. Aufgrund der Komplexität von Kompressionsalgorithmen und der Vielfalt an Formaten werden nur markante Unterschiede explizit erwähnt. Darüber hinaus werden ausgewählte Methoden zur Datenreduktion von Game Assets beschrieben und in ihren Kontext zur Bedeutung in einer Game Engine gesetzt, damit die Reduktion der Daten bei Anwendung der Machinima-Methoden im Bereich des Animationsfilms im Vergleich zum üblichen Videostream verdeutlicht wird.

3.1 Kompression

Filme liegen bevorzugt als Videostream in einem Codec auf Datenträgern, wie z.B. Festplatte oder Flashspeicher, oder z.B. auf DVD und Blu-Ray vor oder befinden sich auf einer Streaming-Plattform. Die Feststellung einer erfolgreichen Datenreduktion von Ausgangsmaterial kann nur im Vergleich zu einem bestehenden Verfahren erfolgen.[24]

Eine Analyse eines speziellen Containerformates[25] und eines speziellen Codecs in seiner Kompressionsqualität und weiteren Merkmalen würde den Rahmen dieser Arbeit überschreiten.[26] Aber die grundsätzliche Funktionsweise wird im weiteren Verlauf der Arbeit im Vergleich zum Replay modellhaft dargestellt.

Die angewendete Kompressionsmethode sollte idealerweise plattformunabhängig sein, was aber nicht bedeutet, dass sie nicht an technische Voraussetzungen gekoppelt ist. Oft werden von Seiten der Hardware Kompressionsverfahren in z.B. *System on a Chip* implementiert, um so von Seiten der Software her weniger Rechenleistung des Hauptprozessors zu benötigen. Es sind also Spezialchips oder Rechenwerke in Prozessoren, die nur diese Algorithmen für entsprechende Kompressionen „kennen“

[24] Vgl. *Riegler, T.,* TV, 2011, S. 46.
[25] Containerformat und Kompressionsalgorithmus werden im Sprachgebrauch oft als Synonyme verwendet.
[26] Für detaillierte Informationen zu Bild- und Audiokompression, vgl. *Steppart, M*., Audio, 2014 und *Ohm, J.-R.,* Bildcodierung, 1995.

und bei Bedarf umsetzen. Eine Veränderung des Verfahrens in der Hardware ist damit oft unmöglich, dies ist der Fall, wenn z.B. kein Update einer Firmware oder des speziellen Chips möglich ist. Eine Softwareumsetzung ist jedoch mit erheblichem Rechenaufwand verbunden und würde andere Bereiche bei der Nutzung des Systems ausbremsen. So unterstützt der Tegra 3 SoC der OUYA z.B. nur eine ausgewählte Kompressionsstufe des H.264-Standards für MP4-Filme.[27]

Komplett neue Kompressionsverfahren werden daher eher auf komplexen Systemen wie dem Personal Computer entwickelt. Durch diese Einschränkungen sind Videostreaming-Plattformen mitunter dazu genötigt, ihre Streams in unterschiedlichen Formaten für unterschiedliche Geräte anzubieten, da nicht alle Geräte dieselben Codecs unterstützen und doch eine große Anzahl an Verbrauchern erreicht werden soll.

Diese Hardware-Unterstützung bzw. Nicht-Unterstützung liegt sowohl bei Videodaten als auch bei Audiodaten vor. So können einige Systeme MP3s[28] unterstützen, andere z.B. nur Linear-PCM.[29] Und ebenso verhält es sich mit Bilddaten, wenn diese nicht direkt von der Hardware unterstützt werden, muss dies in Software gelöst sein und verlangsamt dadurch unter Umständen das Anzeigen. Beispiele für oft unterstütze Formate sind JPEG[30] und PNG. Bei Grafikeinheiten kann sich dieses Phänomen auf die Unterstützung von komprimierten Texturen im Grafikspeicher beziehen.

Der Tegra 3 SoC unterstützt z.B. DXT, aber kein ASTC. Diese Einschränkungen können ebenfalls von Grafikeinheit zu Grafikeinheit in verschiedenen Systemen unterschiedlich sein und werden daher durch Standards wie z.B. OpenGL, OpenGL ES, Vulkan und DirectX spezifiziert. Wenn eine bestimmte Unterstützung vorliegt, so sind daran auch andere Unterstützungen gekoppelt.

Als Beispiel: Unterstützt eine Grafikeinheit den Standard OpenGL ES 2.0, so unterstützt diese Grafikeinheit auf jeden Fall die Texturkompression ETC. Unterstützt die

---

[27] H.264 (HP @ 40Mbps) High-Profile, vgl. *NVIDIA-Corporation*, Tegra 3, 2018. H.264 umgangssprachlich als MP4 bekannt und H.265/HEVC als DVB-T2-HD-Standard (Deutschland) bekannt.
[28] Vgl. *Steppart, M.*, Audio, 2014, S. 49 und *Ohm, J.-R.*, Bildcodierung, 1995, S. 446f.
[29] Vgl. *Steppart, M.*, Audio, 2014, S. 24ff und *Ohm, J.-R.*, Bildcodierung, 1995, S. 3 und 12.
[30] Vgl. *Ohm, J.-R.*, Bildcodierung, 1995, S. 430-433. Zwischen Kompressionsverfahren und der Datentyp-Bezeichnung finden sich im allgemeinen Sprachgebrauch mitunter Diskrepanzen. So wird JPEG als Bildformat oft für JFIF gebraucht.

Grafikeinheit den Standard OpenGL ES 3.1, kann davon ausgegangen werden, dass auch der Texturkompressionsstandard ASTC unterstützt wird.[31]

3.2 Vergleich der Kompression im Videostream und Computerspiel

Ein einfaches verständliches Container-Format ist AVI-RIFF.[32] Auf die konkrete Umsetzung und oder Unterschiede zu MP4 und anderen Containern wird nicht eingegangen, sondern es wird nur die Funktionsweise grob erläutert.

AVI, also *Audio Video Interleave*, in Kombination mit RIFF, also *Resource Interchange File Format*, spezifiziert eine bestimme Form, wie unterschiedliche Medien verschachtelt in einer Datei untergebracht werden müssten, damit sie dann korrekt wiedergegeben werden können. Diese Verschachtelung von unterschiedlichen Formaten und Datentypen geschieht mehr oder weniger in Datenblöcken, den sogenannten *Chunks*. Als einfache Annahme sind in dem Videostream Rasterpixeldaten in Form von Bildern und Teile von digitalisierten Audiowellen. Ein einfaches Format für Pixelbilder ist das BMP-Format und ein einfaches Format für Audiodaten ist das WAVE-Format[33].

Bilder können mit Verfahren wie JPEG z.B. komprimiert werden, dies kann verlustbehaftet oder verlustfrei geschehen. Dabei werden nebeneinanderliegende Pixel mit einander kombiniert und bei verlustbehaftetem Verfahren auch in ihrer Farbvielfalt reduziert.[34] Für die Videokompression würde dann auch die Möglichkeit bestehen, dass Bilder, welche aufeinander folgen, nur Unterschiede zum vorhergehenden Bild aufweisen und so wiederum Platz gespart werden kann. Bei der MP3 als Beispiel für die Kompression, werden für die meisten Menschen kaum wahrnehmbare Geräusche kombiniert, um so nur einen Eindruck „desselben“ Klanges zu erzeugen. Auch dies geschieht für ein paar Datensamples, also ein paar Abtastungsschritte hintereinander und dann werden diese Daten ähnlich wie in RIFF vorgegeben in Chunks aufgeteilt. In einem Videostream liegen demnach mehrere Streams unterschiedlicher Medientypen vor. So ist es z.B. möglich, den Sprachstream bei DVDs zu wechseln, um eine andere Audiospur für eine andere Sprache zu erhalten oder bei einem

[31] Zur detaillierten Beschreibung und Unterscheidung der unterschiedlichen Texturspeicherkompressionsverfahren, vgl. *Chait, D.,* ASTC, 2018.
[32] Vgl. *Steppart, M*., Audio, 2014, S.25.
[33] Vgl. *Steppart, M*., Audio, 2014, S. 26.
[34] Vgl. *Ohm, J.-R.*, Bildcodierung, 1995, S. 430-433.

Stream aus dem Internet zwischen unterschiedlichen Qualitäten und Datenraten zu springen.

Der grundlegende Ansatz der Kompression liegt aber darin, dass es Aufzeichnungen gibt, welche im Vorfeld in irgendeiner Form quantisiert worden sind, um so aus der bestehenden analogen Natur ein digitales Abbild zu schaffen und aus diesen digitalen „Rohdaten“ dann eine Verkleinerung zu erzeugen.

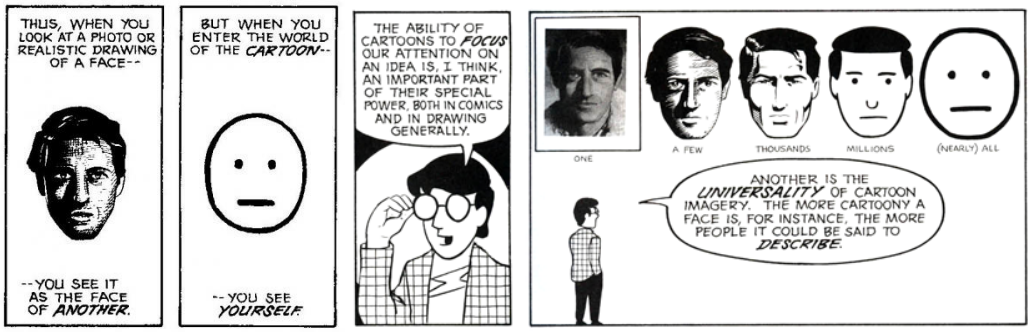
Die Spiele-Entwicklung verfolgt einen anderen Ansatz. Sie versucht innerhalb des Computers digitale Welten modellhaft nachzubilden. Ist aber nicht darauf angewiesen, dass diese erzeugten Welten in der Realität im Vorfeld existiert haben. Es ist also eher eine Nachbildung als eine Umwandlung. Der Begriff Synthese im Zusammenhang mit nachgebildeten akustischen Instrumenten beschreibt dies sehr treffend.

Wenn man so will, kann man sich der Kompression von zwei Seiten nähern. Einmal mit der maximalen Menge an Daten aus der Natur, welche dann durch Algorithmen verkleinert werden. Und das andere Mal, komplett synthetisiert und vom Kleinsten ausgehend so lange Daten hinzugeben, bis eine vergleichbare bzw. für den Zweck der Präsentation ausreichende optische oder akustische Qualität ähnlich der Natur erreicht wird.

In der Akustik, vor allem der Musik, mag die Synthese einfacher sein, da auch Instrumente selbst schon in gewisser Weise eine Nachbildung von in der Natur möglichen Lauten und Geräuschen sind. Also eine Klangsynthese schon im Vorfeld, noch vor der Digitalisierung, stattfand. Bei Filmen würde das Theater mit seinem Bühnenbild einer Illusion einer Szenerie entsprechen.

Beim Storytelling selbst, ohne den Anspruch auf literaturwissenschaftliche Korrektheit zu erhalten, werden ebenfalls Annahmen getroffen, welche erst beim Leser zu einem Gesamtbild im Kopf zusammengesetzt werden. Dies soll an dieser Stelle der Arbeit nicht den Anspruch einer filmwissenschaftlichen Abhandlung zur Umsetzung von Filmen genügen. Doch ist eine modellhafte Beschreibung der Parallelen zwischen Erzählungen und Spielen notwendig, um die benötigen Elemente für die Methode der Kompression klarer einschränken zu können.

Abbildung 3: Abstraktion im Comic[35]



Nehmen wir den Ansatz von McCloud, und betrachten die linke Seite als Realismus und die rechte Seite als sehr vereinfachtes Schema bzw. Modell, dann kann die Game Engine betrachtet werden als nähert sie sich mit ihren Methoden und Möglichkeiten von rechts nach links und der Videostream von links nach rechts dieser Darstellungsform. Und bei der Dateigröße bei gleicher optischer und akustischer Qualität wird es eine Gewinnschwelle geben, wo das eine Verfahren dem anderen dann überlegen sein würde. Wie McCloud erwähnt, würde der Mangel an Details in der Darstellung dem Storytelling keinen Abbruch tun müssen, denn je höher der Abstraktionsgrad des Protagonisten sei, desto höher sei auch der Identifikationsgrad des Rezipienten mit selbigen.[36]

3.3 Minimalste Datenmenge

Die minimalste Datenmenge für eine spezifische Animation mit Geräuschen und Musik wird in einer Demo umgesetzt. Der Aufwand für diese Demos ist speziell ausgerichtet auf das Zusammenspiel zwischen Animation, Bewegungen, den anderen visuellen Eindrücken und eben der Musik und den Geräuschen. Es existieren sogar Demos, welche Sprache in Form von Sprachsynthese enthalten.[37] Doch diese Demos sind nicht einfach zugänglich in der Erstellung wie Machinimas, welche auch mit üblichen Videoschnittprogrammen geschnitten und editiert werden können. Im Folgenden wird erläutert, was notwendig ist, um ausreichende Informationen für einen Film inklusive Musik und Story zu geben und dennoch gleichermaßen einfach editierbar zu sein.

---

[35] Aus *McCloud, S.*, Comics, 2001, S. 38 und S. 44 (jeweils Ausschnitt).
[36] Vgl. *McCloud, S.*, Comics, 2001, S. 32-67.
[37] Vgl. *Sixtus, M.,* Hollywood, 2008.

Denkbar wäre hier ein Drehbuch oder eine Partitur, vergleichbar ist auch ein Replay, welches dann die einzelnen Abläufe der im Spiel vorkommenden Protagonisten protokoliert.[38] Anhand des Beispiels von StarCraft 2 wird das Potential des Replays deutlich und sie sind in ihrer Dateigröße durchaus mit Demos vergleichbar, wenn die eigentliche Game Engine mit den für das Spiel benötigten Assets nicht mit in die Berechnung der benötigten Gesamtdatenmenge zur Darstellung des Films einfließen.

Um eine Datenreduktion mit Machinima-Methoden zu erreichen, ist es daher unabdingbar, auch die benötigten Assets auf ein Minimum an Datenverbrauch zu reduzieren. In einer Demo werden die benötigten „Assets“ während der Laufzeit erstellt. On-The-Fly im Speicher werden 3D-Objekte, Geräusche, Musik und weitere Effekte und benötigte Dinge angefertigt und zusammengefügt. Diese Assets sind generiert und wurden vorher nicht in üblichen Formaten wie JPEG, MP3 oder OBJ abgelegt. Sie existieren nur in Code-Form.[39]

Einige dieser Methoden sind auch in Spielen und Game Engines zu finden, so kann Musik z.B. in Form von Noten vorliegen, welche zur Laufzeit des Spiels in Echtzeit dann in Töne umgesetzt werden. Dadurch sind auch u.a. dynamische Veränderungen möglich, welche mitunter das aktuelle Spielgeschehen berücksichtigen und sich beispielsweise im Tempo oder der Lautstärke ändern können.

Aus Platzgründen und aus der Frühzeit der Computerspiele entstand schon auf dem Amiga das sogenannte „Tracker-Format“ oder auch bekannt unter der Bezeichnung „Module“. Durch den Soundprozessor bedingt, wurden Musikformate entwickelt, welche bis zu 4 gleichzeitige Geräusche abspielen konnten, die sogenannten „Stimmen“. Während bestimmter Zeitintervalle werden Töne, also zuvor digitalisierte Wellenformen, aus dem Speicher in je nach benötigter Ton-Höhe transponiert abgespielt. So entsteht Musik. Für die Musik benötige Instrumente werden als Wavesamples zuvor im Speicher abgelegt. Das Tracker-Verfahren ähnelt dem MIDI-Verfahren, nur dass bei MIDI sich die Instrumente in SoundFonds oder im Wavetable der Soundkarte befinden.[40]

---

[38] Vgl Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** ***Fehler! Verweisquelle konnte nicht gefunden werden.*** S. 9.
[39] Vgl. *Sixtus, M.,* Hollywood, 2008.
[40] Vgl. *Steppart, M*., Audio, 2014, S. 137 und 139.

Das Tracker-Verfahren hat den Vorteil, dass die benötigten Instrumenten-Samples im Format enthalten sind, die Klangqualität an die von der Audio-CD heranreichen kann und im Vergleich zur MP3 eine kleinere Dateigröße bei gleicher Lauflänge aufweisen kann und darüber hinaus zur Wiedergabe keinen komplexen in Hardware implementierten Algorithmus benötigt, um dennoch eine gute Wiedergabe auf schwächerer Hardware zu ermöglichen. Nachteil dieses Verfahrens ist die Limitierung auf Noten und Instrumente. So sind Gesang oder das gesprochene Wort zwar möglich, aber profitieren nicht von der Art und Weise der Speicherung. Im Vergleich wendet das Tracker-Verfahren aber keine psychoakustischen verlustbehafteten Verfahren zur Datenreduktion an. Es ist in gewisser Weise verlustfrei.

Beispiel mit kombinierter Instrumenten-Synthese:

Die komplette Bibliothek an Musik ist mit dem Hively Tracker abspielbar.[41] Datei Alltunes.zip wird mit 88K angegeben. Der XMPlay gibt eine Gesamtdauer aller in Alltunes.zip enthaltenen Musik mit 1:32:51 an. Also über 1,5 Stunden Musik in 88 kByte, wobei diese entpackt und neu mit 7Zip gepackt noch mal in der Dateigröße schrumpfen und zwar auf ca. 55 kByte.[42] Das ist auch das Prinzip bei der Audiokompression: Datenreduktion wird durch Vermeidung von Redundanzen erreicht.

3.4 Datenreduktion durch die Machinima-Methode

Der Ansatz, welcher in dieser Arbeit verfolgt wird, um Daten zu reduzieren, ist ein Kompromiss aus den aufwendig erstellten Assets via Quellcode, welche in der Demoszene angewendet werden und den aufwändig angefertigten Assets mit 3D-Modellierungsprogrammen in der Filmproduktion, welche kein Echtzeit-Rendering mehr auf Grund ihrer Komplexität zulassen. Zu diesem Zweck werden die vorher beschrieben Machinima-Methoden auf das Potential für die Datenreduktion hin betrachtet und erläutert. Um die Datenreduktion zu erreichen, dürfen nicht die fertig generierten Bildschirminhalte als Videodaten gespeichert werden, sondern müssten wie in Multiplayer-Spielen, unter anderem bei Fortnite, einfach durch die Player-Bewegungsdaten (Spieler in einem Multiplayer-Spiel) oder ähnlich wie in In-Engine-Schnittszenen in Echtzeit gerendert werden. (z.B. beim N64 aus Platzgründen häufig verwen-

[41] Vgl. http://www.hivelytracker.co.uk/tunes/
[42] Vgl. http://www.hivelytracker.co.uk/

det). Um dies zu ermöglichen, müsste ein Teil einer Game Engine nachgebaut werden. Das Erstellen einer lauffähigen Game Engine ist für sich genommen kein triviales Thema und kann hier auf Grund seines Umfangs nicht ausführlich behandelt werden. Es wird deshalb davon ausgegangen, dass die Game Engine als Motor für die Anwendung ausführbar ist. Deshalb liegt der Fokus bei der Reduktion der Daten auf den in der Game Engine verwendeten Assets, welche für ein Machinima benötigt würden. In erster Linie wird die Reduktion der Daten durch die Vermeidung von Redundanzen erreicht. Zu diesem Zweck kann sich der Ansatz der Reduktion wie folgt vorgestellt werden: Es handelt sich dabei um mehrere Kompressions-Phasen, welche unter anderem eine Abstraktion beinhalten und damit eine Verallgemeinerung, was gleichzusetzen ist dem Verschwinden von Details. Dies ähnelt von der Idee her der Farbunterabtastung (dem Chroma-Subsampling) bei beim JPEG-Verfahren, wird aber auf gänzlich anderem Wege erreicht.

Beschreibung:

Ein Objekt, z.B. ein Stuhl wird nur als Stuhl gespeichert, aber nicht als bestimmter Stuhl mit bestimmten Merkmalen oder Verzierungen. Um allerdings eine Szene ausreichend darstellen zu können, muss es eine definierte Menge an Objekten geben, welche in großer Häufigkeit im Sprachgebrauch auftreten.

Ähnlich wie in der deutschen Sprache der Vokal „E“ mit einer starken Häufigkeit in deutschen Texten aufzufinden ist.

Durch Nutzung der Methodik, wie sie in der Huffman-Kodierung Anwendung findet, könnten entsprechend die IDs auf benötige 3D-Objekte (Gegenstände) verteilt werden.

Dies würde dafür sorgen, dass oft benötige 3D-Objekte ebenfalls schneller geladen würden. Ob wie beim JPEG-Kompressions-Verfahren in Bezug auf ein Bild entsprechend auf eine Filmsequenz bezogen ein neues Wörterbuch erstellt werden muss, welches dann der Huffman-Kodierung unterzogen wird, muss überprüft werden.

Das entwickelte Verfahren geht von einer eindeutigen 3D-Objekt-Referenz-ID aus, damit Szenen und Filme miteinander einfacher kombiniert werden können. Ähnlich einer Farbpalette, welche sich allerdings nicht pro Bild ändert, sondern für alle Bilder einheitlich ist.

Ein üblicher Haushalt am Vorabend des ersten Weltkrieges soll ca. 180 Dinge umfasst haben, heute sollen es rund 10.000 sein. Marie Kondo beschreibt in ihrem Buch „*Spark Joy: An Illustrated Master Class on the Art of Organizing and Tidying*", dass es nicht alles an Dingen braucht, was der Mensch im Laufe seines Lebens angesammelt hat. Dieser Ansatz wird bei der Nutzung der Game Assets ebenfalls verfolgt.

Es wird nur referenziert. Zu diesem Zweck müssen es nicht 10.000 unterschiedliche Gegenstände sein. Eine Gabel wäre ein Asset, 6 Gabeln würden zwar 6 sichtbare Objekte sein, aber auf dasselbe Objekt referenzieren. Um die Objekte dann wieder voneinander unterscheiden zu können, bekommen diese im Speicher unterschiedliche IDs und damit sie auch sichtbar werden und nicht exakt auf derselben Stelle im Raum liegen und sich so überlagern und damit überdecken, würden sie auch unterschiedliche Koordinaten im Raum erhalten. Dies ist ein Standardvorgehen bei Game Engines und kann daher als Machinima-Methode angesehen werden. Bäume in einer Landschaft werden beispielsweise nur etwas gedreht oder in ihrer Größe verändert, bleiben aber grundsätzlich dasselbe Asset. Weitere Transformationen erfolgen während der Laufzeit mit minimalem Rechenaufwand durch die Grafikeinheit.

Diese Art der Kompression ist im Ansatz deshalb nur bedingt mit dem Streaming vergleichbar. Parallelen könnten sich in der Inszenierung und dem Szenenwechsel ergeben. Wenn im Videostream ein neues Schlüsselbild, das sogenannte I-Frame, benötigt wird, kann dies in einer Game Engine das Nachladen von Assets vom Festspeicher in den RAM bedeuten. Das Herzstück einer Kompression beim Stream wäre der Codec. Also Encoder und Decoder. Dies könnte mit der Game Engine verglichen werden. Und der Datenstrom, also der eigentliche Stream, sind dann nicht die komprimierten Pixeldaten, sondern eventuell wie im Replay nur Daten, welche Transformationen auf bereits vorhandene Assets darstellen. Auf Grund des Potentials einer Urheberrechtsverletzung durch den Ersteller, also dem Story-Teller, bei Machinimas, wäre der „Machinima-Methoden-Datenreduzier-Codec" idealerweise frei von Patenten und Rechten Dritter. Zusammengefast wäre der Kompressions-Codec eine freie Software, welche das Erstellen und Abspielen von Animationen ermöglicht und der Funktionsweise der kommerziellen Software Plotagon ähneln kann.[43]

[43] Vgl. *Dutt, A.,* 3D Motion Capture, 2018.

3.5 Ansatz der Kompression durch die Machinima-Methode

Auf der einen Seite existieren Kompressionsverfahren für den „Realismus“ in Form der komprimierten Pixeldaten, auf der anderen Seite existieren Demos, welche durch minimalsten und sehr optimierten Programmcode Synthese in reinster Form betreiben. Der Ansatz in dieser Arbeit ist es, durch die Vorzüge von Machinima-Methoden für das Storytelling eine Datenreduktion im Vergleich zum Streaming zu erreichen. Das Machinima besteht aus in Echtzeit gerenderten Game Assets. Das Replay ist die Aufzeichnung der Veränderungen dieser Game Assets. Doch anders als beim Demo, werden üblicherweise Game Assets nicht erst zur Laufzeit erzeugt, sondern liegen schon als Daten in Dokumenten vor. Zur Veranschaulichung dient hier das OpenSource-Projekt FPS-Creator-Classic, einem Spiele-Editor von TheGameCreators.

In *Abbildung 9: Netflix-Datenraten* sind die Dateien nach Dateityp-Häufigkeit und Datengröße sortiert dar-gestellt.[44] Zur Erstellung der Übersicht wurden die Daten entsprechend gezählt und im Anschluss mit Excel sortiert. Das dazu verwendete PureBasic-Programm „FPSC_Classic_DatenAnalyse.pb“ befindet sich im Dateianhang.[45]

Aus der Tabelle ist ersichtlich, dass der überwiegende Datenanteil auf die Assets mit der Endung DDS und mit der Endung X fällt. Bei DDS handelt es sich um Textur-Daten im DXT-Format. Und X-Dokumente sind 3D-Objekte. Als 3. Datentyp mit dem meisten Speicherplatzbedarf sind ausführbare Dateien auszumachen. Vermutlich handelt es sich hier um weitere Tools und Programme, welche im Game Engine SDK weitere Funktionalitäten zur Verfügung stellen. Die größte ausführbare Datei ist der Haupteditor, die Datei „FPSC-Game.exe“ mit über 24 MB. Des Weiteren folgen nun 374 WAV-Daten bzw. WAVE-Daten, welche Audio-Daten wie Geräusche und Sprache enthalten. Die verbrauchen ~ 70 MB.[46]

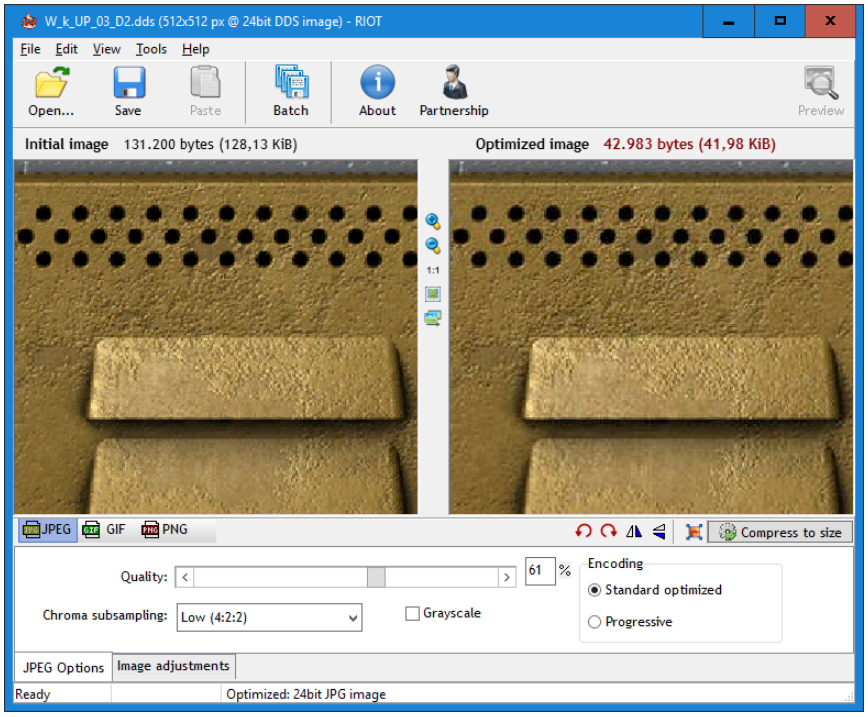
Des Weiteren ist im Auszug der Texturen ersichtlich, dass sich Texturen der Assets stark ähneln können. Ferner ist festzustellen, dass sich aus einer Textur andere ergeben können oder sich Muster in Teilen von einer Textur in anderen Texturen wiederfinden können. Da das DDS-Format schon das Abbild des späteren Grafikspeichers ist, kann die DDS-Datei zwar sehr schnell im Speicher verarbeitet werden, ist

---

[44] Siehe Tabelle 5: Auswertung FPSCC, S. 42.
[45] Quelle der Daten: https://github.com/TheGameCreators/FPS-Creator-Classic
[46] Vgl. Abbildung 11: FPSCC-Texturen-Auszug, S. 47.

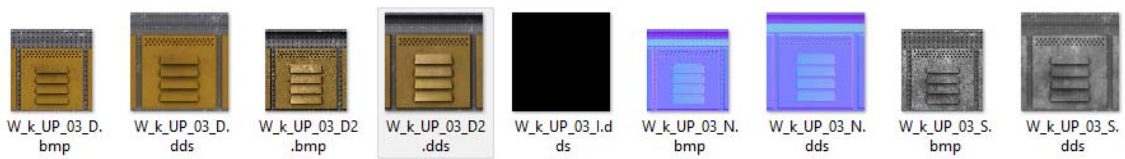
aber auch in ihrer Dateigröße zwingend festgelegt. Um hier einen Kompromiss zu finden, kann das JPEG-Verfahren angewendet werden, welches dann erst im Speicher in ein für die Grafikeinheit lesbares Format umgewandelt wird. Die Ladezeiten würden sich dadurch erhöhen, da eine Dekodierung des komprimierten JPEG-Bildes stattfinden muss, es würde aber durchaus Einsparpotential in der Dateigröße im Festspeicher mit sich bringen. Als Beispiel: eine DDS mit der Auflösung 512 x 512 Pixel ist immer exakt 128,13 KiB groß. Ein optisch vergleichbares JPEG kann 1/3 der Dateigröße sein.

Abbildung 4: Grafikvergleich



Darüber hinaus ist festzustellen, dass Texturen in unterschiedlichen Formaten zur Berechnung von Lichteffekten vorliegen.

Abbildung 5: Texturen



Die Dateien mit der Endung „bmp“ sind nur als Vorschaubilder gedacht und können bei der Betrachtung vernachlässigt werden. Jedoch sind die DDS-Dateien mit folgenden „Suffix“ erkennbar: _D, _D2, _I, _N, _S für eine visuell weniger imposante

Lichtstimmung in der verwendeten Game Engine reicht es aus, nur die Textur mit dem Suffix „_D2“ zu nutzen. Diese Textur ist eine Zusammenführung der anderen 4. Statt also 5 Texturen, würde sich der Bedarf auf eine reduzieren.

Tiefer soll die Datenanalyse dieser speziellen Game Engine mit ihren Assets an dieser Stelle nicht gehen, dies würde den Rahmen dieser Arbeit überschreiten. Die genauen Größen sind nicht ausschlaggebend. Veranschaulicht wird jedoch, dass eine Optimierung in den Assets für Texturen, 3D-Objekte, den Audiodaten und gegeben falls bei der auszuführenden Datei das meiste Einsparpotential auf Grund ihres prozentualen Datenverbrauchs in Bezug auf das Gesamtprojekt liegt.

Bei 3D-Objekten kann eine Polygon-Reduktion erfolgen, es kann auch auf ähnliche Modelle verzichtet werden, welche später durch Veränderung in den Proportionen wie-der neues Aussehen erhalten. Sound und Musik könnten als Kombination aus Geräuschsynthese bzw. Instrumenten-Synthese und Noten im Tracker-Verfahren abgespeichert werden. Bei Texturen könnte ein Stil gewählt werden, welcher weniger an den Realismus angelehnt ist und mehr an ein Cartoon erinnert. Dies würde dem Effekt entsprechen, den McCloud angesprochen hatte. Fortnite, einige Blizzard Spiele wie WoW und Overwatch, aber auch Dota 2 und Team Fortress 2 von Valve und einige Nintendo Spiele wie Super Mario Odyssee bedienen sich ebenfalls dieser Methode. Einige Spiele verzichten auch komplett auf Texturen und reduzieren zu dem die Anzahl der Polygone der verwendeten Objekte weiter, dies wird auch als Low-Poly-Stil bezeichnet. Als Ver-treter im Anwendungsbereich sei hier Plotagon erwähnt, welches auch das Storytelling umsetzt.

Das Storytelling-Animationsprogramm Plotagon reduziert z.B. die Handlungsoptionen des Nutzers und die Gestaltungsfreiheit der enthaltenen Assets in der Form, dass es Vorgaben zu bestimmten Kategorien aufzeigt. Darüber hinaus wird Sprachsynthese für gesprochene Dialoge eingesetzt. Dadurch werden keine vorher aufgenommenen Sprachsamples benötigt. Allerdings bedeutet dies, dass eine Sprachsynthese entweder mit in die auszuführende Game Engine implementiert werden muss, oder aber das System, auf welchem das Echtzeit-Machinima dann abgespielt werden soll, eine solche Synthese von Haus aus mitbringt. Nicht immer sind alle Stimmen und Sprachen auf den Systemen vorhanden. Eine gleichklingende Qualität kann also hier nur gewehrleistet werden, wenn eine Sprachsynthese durch die Abspielsoftware selbst, also die aus-führende Game Engine, mitgeliefert wird. Für eine

gute Sprachsynthese muss ebenfalls benötigter Speicherplatz berücksichtigt werden. Eventuell kann es sich dabei lohnen, nur wenige gesprochene Texte in OGG-Vorbis mit dem einem für Sprache optimiertem Kompressionsalgorithmus wie z.B. Speex zu wandeln. Eine gemeinfreie Sprachsynthese kann eventuell in Zukunft durch das folgende Mozilla-Projekt realisiert werden.

Abbildung 6: Gemeinfreie Sprachsynthese[47]

Es wäre nun denkbar, dass auch für ein Echtzeit-Machinima nur bestimmte ausgewählte Assets genutzt werden, welche anschließend immer wieder neu kombiniert werden können. Wenn diese nun nach genannten Verfahren verkleinerte Assets nutzen und als Replay aufgezeichnet würden, wäre eine sehr hohe Einsparung gegenüber dem Video-stream möglich. Allerdings hängt das Einsparpotential stark von der Minimierung der Assets ab und die Minimierung der Assets kann den Stil im visuellen als auch im auditiven Bereich beeinflussen.

Sollten im eventuell resultierenden Abspielprogramm ähnlich wie in Plotagon vordefinierte Assets zur Verfügung gestellt werden, damit Zuschauer und Filmemacher selbst Regie führen können, so sollte darauf geachtet werden, dass für diese Assets ebenfalls entsprechende Lizenzen vorhanden sind oder diese Assets gemeinfrei sind.

Eine Möglichkeit hierfür wären zum Beispiel die Assets aus dem angesprochenen Programm FPS Creator Classic. Da diese Open Source sind. Wenn das Erstellen solcher Assets selbst zu aufwendig ist, bietet die Seite kenney.nl z.B. gemeinfreie Assets im Low-Poly-Stil mit einfarbigen Texturen, welche einen geringen Speicherplatzbedarf haben. 141 Objekte zum Thema „Wohnen“, welche beliebig, je nach Grafikhardwarevermögen, in der Game Engine instanziiert werden können, um so den Eindruck einer Szenerie in einer Geschichte zu bilden, verbrauchen bei Kompression mit dem Programm 7Zip ~ 304 kByte. Hier besteht kein Anspruch auf eine genaue

[47] Siehe https://voice.mozilla.org/de

Datengröße und es kann auch davon ausgegangen werden, dass andere Kombinationen von Objekten andere Dateigrößen haben werden, es soll lediglich das Einsparpotential verdeutlicht werden.[48]

Abbildung 7: Assets[49]

Durch die Auswahl an vordefinierten „Stücken“ und die Instanziierung im Zusammenhang mit Low-Poly-Modellen und oder vereinfachten Texturen besteht starkes Reduktionsvermögen. Für die Musik könnte z.B. das Tracker-Verfahren genutzt werden.

Versuche der Standardisierung, unter anderem durch Pixar[50]

3.6 Vergleich zum Streaming

Beim Spielestreaming wird das sichtbare Bild des Spielenden als Videostream via entsprechender Internetplattform an seine Zuschauer übertragen. Bekannte Beispiele für solche Plattformen wären z.B. Twitch von Amazon oder YouTube von Google. Das Videosignal muss in Echtzeit in ein Videostream-Format konvertiert werden und möglichst ohne große Verzögerung auf die Plattform hochgeladen werden. Technisch bedeutet dies, dass die Erstellung der Videodatei beim Spieler geschieht und im Anschluss zur Videoplattform hochgeladen werden muss und von dort dann wieder an den einzelnen Zuschauer heruntergeladen werden muss. Da es sich um Streaming-Inhalte handelt, geschieht dies in Teilen, Chunks oder Paketen, damit das Gefühl einer Live-Übertragung aufrecht gehalten werden kann.[51]

[48] Vgl. Abbildung 10: Kenney-Assets, S. 46.
[49] Siehe https://www.kenney.nl/assets/furniture-kit
[50] Vgl. *Pixar Animation Studios*, USD, 2017.
[51] Siehe Abbildung 9: Netflix-Datenraten, S. 36.

Das Potential der Machinima-Methoden zur Datenreduktion ist bei optischen Einschränkungen erkennbar. Als Überschlag können folgende Annahmen getroffen werden:

1,5 Stunden Musik als Hively-Tracker AHX bzw. HVL verbraucht ca. 55 kByte

Das StarCraft 2 Replay der Veranstaltung „WCS Leipzig 2018“ in bei dem sich die Spieler Serral und ShoWTimE im 3. Spiel auf der Spiele-Karte Backwater LE gegenüberstanden ist knapp 36 min. lang bei einer Dateigröße von 683 kByte.[52]

Bei angenommener gleicher Datenrate wäre also ein 90 min. Replay unter 2 Mbyte groß. Im StarCraft 2 Spiel kann jeder Spieler mehrere hundert Einheiten steuern. So wäre es bei Szenen in Filmen mit weniger handelnden Personen durchaus denkbar, dass das Replay wesentlich geringer in seiner Datenrate daherkommt.

Mit den 3D-Objekten von Kenney und dem Verzicht auf aufwändige Texturen wäre weiteres Einsparpotential denkbar. Wenn es sich bei der Story z.B. um eine Sitcom handeln würde, würde alles „zu Hause“ spielen können. Das darstellte Asset-Pack „Furniture“ (Deutsch: Möbel) wäre dafür geeignet und kommt auf einen Verbrauch von 304 kByte bei zusätzlicher Kompression mit 7Zip.

Was für ein Storytelling noch von Nöten wären, sind die Protagonisten selbst und die Sprache bzw. Stimmen der handelnden Personen.

Setzen wir die maximale Datenrate vom patentfreien Speex-Codec an von 44 kbit/s würden dies bei 90 min. dann 5400 Sekunden sein, also 29700 Byte für 90 min. Sprache. Diese Datenmenge wird in der Regel allein dadurch unterschritten, dass nie in einer Geschichte 90 min. ununterbrochen auch nur ohne eine Sekunde eine Sprechpause zu machen, redet. Es wäre also eher eine Mischung aus Sprache und Musik. Wobei Geräusche ebenfalls mit dem Synthese-Verfahren ähnlich der Musik erstellt werden könnten. Es bleibt also die Frage, in wie weit die Protagonisten Platz benötigen. „aiko_L.X“, (siehe Tabelle 5: Auswertung FPSCC, S. 45) ist unter 4 MB groß. Durch geschickte Verformung, aber auch die Veränderung der Texturen, können daraus weitere Protagonisten werden. Ohne den Anspruch auf eine vollständige Analyse im Bereich der Filmwissenschaft an den Tag zu legen, wird hier von einer Anzahl an ~ 40 unterschiedlichen Darstellern ausgegangen, welche alle Animationen

[52] Vgl https://lotv.spawningtool.com/34103/

einer Hauptfigur in einem Film enthalten, wobei diese aus 10 Grundfiguren bestehen und sich eventuell in Größe und „Farbe“ unterscheiden. (siehe Anhang: Abbildung 12: Darstellung der verschiedenen Charaktere durch den Wechsel einer Textur und Abbildung 14: Darstellung eines Charakters mit verschiedenen Animationen, S. 48). Durch weitere Optimierung in der Datei ähnlich dem 7Zip-Verfahren kann unter Umständen nicht im hohen Maß ersichtlich in der Dateigröße eingespart werden, da diese 3D-Objekte oft im Vorfeld schon optimiert wurden. 10 x 4 ~ 40 MB. + ~ 30 MB maximale Sprache, + ~ 2 MB für den Ablauf der Geschichte selbst und noch einmal, im Vergleich fast zu vernachlässigen, 55 kByte also ~ 1 MB für die Musik. Was nun noch fehlt ist der Bedarf für den Abspieler selbst also den Teil der Game Engine, der dann das Replay, die Musik und die Sprache und die Animationen abspielt. Die „FPSC-Game.exe“, (siehe Anhang: Tabelle 5: Auswertung FPSCC, S. 45) muss mehr oder weniger all diese Funktionen beherrschen, sowie die Darstellung auf einer Grafikeinheit und benötigt ~ 26 MB.

Der 1. Film würde inklusive Player ohne spezielle weitere Optimierung damit auf ~ 99 MB kommen. Dieser Film hätte dann sogar die Länge von 90 min.

Möglicherweise sind Abstriche in der audiovisuellen Qualität im Vergleich zu anderen audiovisuellen Stilen denkbar. Das wäre dann aber Geschmackssache. Im Vergleich zum Videostream mit fester Datenrate wäre hier eine Einsparung erreicht.

4 Anwendung auf einer Mikrokonsole

Wenn das Verfahren auf der Mikrokonsole angewendet werden kann, dann kann es auch für PC und Smartphone bzw. Tablet genutzt werden. Mikrokonsolen bilden den kleinsten gemeinsamen Nenner.

Unterschied zum PC: weniger flüchtiger und fester Speicher und geminderte Rechenleistung im selben Anschaffungsjahr. In der Regel ist die Mikrokonsole deshalb auch kostengünstiger und arbeitet in den meisten Fällen mit einem stromsparenden ARM-Prozessor.

Zu klären ist also, ob diese Mechanismen bzw. Machinima-Methoden, welche in Computerspielen und Netzwerk-Mehrspieler-Spielen bereits eingesetzt werden, sich ebenso für die Nutzung zur Erzählung von Geschichten eben nicht nur auf Computern, sondern auch auf der eingeschränkten Hardware einer Mikrokonsole nutzen lassen. Im Folgenden wird daher erläutert, worin sich genau eine Mikrokonsole unterscheidet und welche möglichen visuellen und akustischen Abstriche dabei zu erwarten sein können.

4.1 Was sind Mikrokonsolen

Der Personal Computer und auch der Begriff des Multimedia-PCs ist in Zeiten von Smartphones, Tablets, Convertables und anderen Multimedia-Geräten nicht immer klar zu trennen. Die Mikrokonsole im Sinne dieser Arbeit steht stellvertretend für eine Gruppe an Geräten, die in erster Linie von der Leistung her zu schwach ist, um an einen aktuellen PC heran zu reichen. Beispielhaft werden im weiteren Verlauf Raspberry Pi, Amazon FireTV Stick und die OUYA näher erläutert werden. Die Mikrokonsole ist gegenüber dem Desktop-PC, einem aktuellen Notebook oder einer vollwertigen Spiele-Konsole und sogar einem aktuellen Smartphone in den meisten Fällen technisch unterlegen.

Die Mikrokonsole ist kein vollwertiger Desktop-Computer, welcher Maus und Tastatur für die Eingabe benötigt und einen Monitor zur Darstellung besitzt. Kann aber durchaus dazu verwendet werden, ähnliche Aufgaben wie das Schreiben oder das Browsen im Web zu meistern. Der Computer in dieser Arbeit ist leistungsstark und aufrüstbar. Er kann eine Workstation mit mehreren Prozessoren und mehreren Gra-

fikkarten sein. Die Abgrenzung kann über einen aktuellen Verkaufspreis neuer Geräte nicht eindeutig erfolgen. Eine Mikrokonsole ist in der Regel günstiger als ein Low-Budget-Smartphone.

Die Grenze zwischen einzelnen Hardware-Klassen verschwimmt Zunehmens. Daher bezieht sich diese Arbeit bei der näheren Betrachtung von Hardware-Einschränkungen auf eine Mikrokonsole.

Und zeigt auf, dass für bestimmte Software-Komponenten ein entsprechender Personal Computer wie beschrieben vorausgesetzt wird. In welcher Bauform dies letztendlich erreicht wird, ist nicht von Relevanz in dieser Arbeit.

Auf einem Computer können Komponenten in der Regel ausgetauscht werden. So kann z.B. eine Grafikkarte ausgetauscht werden oder via USB 3.0 kann z.B. an einen Laptop eine externe Grafikkarte angeschlossen werden. Mikrokonsolen sind meist nicht erweiterbar in irgendeiner Form, bis auf den Festspeicher durch Speicherkarten oder USB-Speicher und Software. Es müsste daher das gesamte Gerät ausgetauscht werden, wenn eine Komponente nicht mehr den Anforderungen entspricht.

Streaming-Boxen mit Android TV oder SmartTV-Aufrüstungen wie Amazon FireTV Stick, Singleboard-Computer wie Raspberry Pi 3, Kleine Spielekonsolen wie die OUYA mit Tegra 3 Grafikeinheit, und Geräte mit ähnlicher Hardware-Ausstattung z.B. Colibri T30, welcher laut Beschreibung mindestens bis ins Jahr 2025 unterstützt wird.

All diese Produkte haben folgende gemeinsame Eigenschaften:

ARMv7 kompatibler Prozessor, ca. 1 GB RAM, Grafikeinheit, welche OpenGL ES 2.0 unterstützt, aber nicht OpenGL ES 3.0. OpenGL ES 3.0-Geräte unterstützen auch OpenGL ES 2.0. Sie haben von Hause aus keinen Einschub für Disc-basierte Medien wie CDs, DVDs oder Blu-Rays.

Die Rechenleistung liegt nach Angaben von gflops.surge.sh zwischen 12 und unter 42 gflops.[53]

Die Mikrokonsole ist von der Grafikleistung her nicht mit einem aktuellen PC vergleichbar. Dieser kann durch Grafikkarten mit RTX-Technologie von z.B. nVIDIA zum Echtzeit-Raytracing betrieben werden und damit eine visuelle Qualität erreichen,

[53] https://gflops.surge.sh/

welche als Fotorealismus bezeichnet werden kann. Dies kann eine Mikrokonsole nicht leisten. Die Möglichkeiten von Echtzeit-Machinimas sind an dieser Stelle nach wie vor begrenzt.[54]

Die Mikrokonsole wird in erster Linie zum Streamen von Filmen genutzt, der Betrachter möchte sich entspannen, zurücklehnen und einfach abschalten können, dazu will er unterhalten werden.

Zur Darstellung von Grafiken und 3D-Objekten werden auf Endgeräten bestimmte Standards zum Ansprechen der Hardware benötigt. Aktuelle Standards sind meist nur in aktueller Hardware und oder in höherpreisiger Hardware zu finden. In der Regel sind Mikrokonsolen der kleinste gemeinsame Nenner, auf welchen sich geeinigt werden kann, bei welchem eine Softwareentwicklung noch sinnvoll sein kann.

Aus Hardwaresicht muss wenigstens vom offenen Grafikstandard OpenGL ES 2.0 ausgegangen werden. Darunter sind sogenannte Workarounds zu kostspielig im Vergleich zur Marktsättigung. Und darüber kann die Marktsättigung noch nicht gewehrleistet werden und damit würde eine Verbreitung nur langsam voranschreiten.

Tabelle 1: Verbreitung von der Unterstützung des OpenGL-Standards auf Endgeräten auf Android-Basis (Daten von 2017)

OpenGL ES Version	Distribution
2.0	38.3%
3.0	43.5%
3.1	18.2%

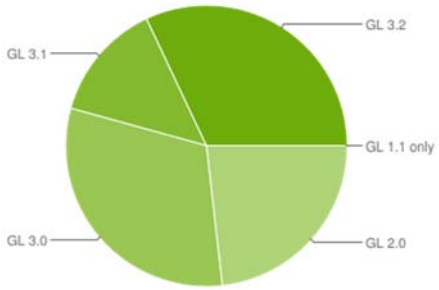
Data collected during a 7-day period ending on May 2, 2017

Tabelle 2: Verbreitung von der Unterstützung des OpenGL-Standards auf Endgeräten auf Android-Basis (Daten von 2018)

OpenGL ES Version	**Distribution**
2.0	21.1%
3.0	29.8%
3.1	13.6%
3.2	35.5%

[54] Vgl. *Schmitt, L.*, Machinima, 2006.

Abbildung 8: Grafische Darstellung von Tabelle 2



Achtung! Daten sind nicht zwingend repräsentativ, da viele chinesische AndroidTV-Streaming mitunter hier nicht Berücksichtigung finden, auch andere Geräte wie ältere AppleTV-Modelle, Ausführungen des Raspberry Pi oder aber auch Geräte aus dem Amazon-Ökosystem fehlen möglicherweise auf Grund eines nicht dargestellten Betriebssystems in dieser Auswertung. Darüber hinaus sind in der Grafik Smartphones und Tablets und andere Geräte eingeschlossen und stellen so ein verzerrtes Bild der tatsächlichen Verbreitung dar, da in der Regel Mikrokonsolen nicht denselben Neukaufzyklen unterliegen müssen wie z.B. Smartphones. Eine genaue Marktanalyse findet in dieser Arbeit nicht statt. Allerdings müsste vor einer konkreten Umsetzung, welche die speziellen Einschränkungen einer Mikrokonsole berücksichtigt, der daraus resultierende finanzielle Aufwand im Zusammenhang mit dem Verbreitungspotenzial berücksichtigt werden.[55]

Problematisch ist eine solche Clusterung des Marktes für Entwickler, da Chipsätze nicht einheitlich, teils technisch veraltet sind und Entwicklung für separate Technologien aufwändig ist. Deshalb wird meist auf den kleinsten gemeinsamen Nenner gesetzt, dies sind die OpenGL ES 2.0 kompatiblen Chipsätze.

Tabelle 3: Grafikeinheiten von Mikrokonsolen (Auszug)

Hersteller	Produktname	SoC	Grafikeinheit	max gflops
Amazon	FireTV Stick	MediaTek	Mali-450 MP4	41.8
Amazon	FireTV 4K	Amlogic	Mali-450 MP3	33.75
Rasberry Pi Foundation	Raspberry Pi 3B	Broadcom	VideoCore4	28.8
OUYA Inc	OUYA	Tegra 3	Geforce ULP	12.5

[55] Vgl. *Google Developers*, Distribution, 2018.

4.1.1 Streaming Box

Streaming Boxen ähneln der Set-top-Box und werden vorzugsweise zum streamen von VoD-Inhalten genutzt.[56]

4.1.2 Rasberry Pi und andere Single Board Computer

Single-Board-Computer sind unter Bastlern und Retro-Spieleliebhabern weit verbreitet. Als Beispiel wurde laut The Verge der Raspberry Pi bis 2017 ca. 12,5 Mio. Einheiten insgesamt verkauft, wobei mehr als 50% auf die ARMv7 kompatiblen Varianten fallen.[57]

Der Raspberry Pi 3B liegt mit 30% an der Spitze. Das entspricht ca. 3,75 Mio. Einheiten. Als Grafikeinheit ist der VideoCoreIV (oder Videocore4) verbaut. Dieser unterstützt OpenGL ES 2.0 vollständig. Der Raspberry Pi ist ab Modell 2 B mit ARMv7 kompatibel. Und bis auf das Modell 3 A+, welches 512 MB RAM besitzt, besitzen die ARMv7 kompatiblen Modelle 1024 MB Arbeitsspeicher.[58]

Andere Single-Board-Computer können ähnliche Merkmale aufweisen und liegen von der Leistung her meist in einer ähnlichen Region. Auf Grund der Vielzahl an Herstellern und Modellen, sowie Modellvarianten kann dies hier nur als Annahme getroffen werden. Einzelfälle müssen separat überprüft werden. So wird dem Raspberry Pi auf Grund seiner großen Community eine hohe Kompatibilität und Unterstützung von Seiten der Software nachgesagt. Bei anderen von der Hardware durchaus leistungsfähigeren Single-Board-Computern ist auch eine fehlende Software-Unterstützung entsprechender Hardware möglich. In dieser Arbeit wird daher bei einer Mikrokonsole davon ausgegangen, dass sowohl Hardware als auch Software entsprechend benötigte Standards unterstützen.

4.1.3 Ouya

Die OUYA bildet gleichermaßen eine Ausnahme wie auch ein repräsentatives Modell einer Mikrokonsole. Mit der Ankündigung des Kickstarter-Projektes im Jahr 2012 wurde der Begriff „Mikrokonsole“ für ähnliche Hardware geprägt. Mittlerweile sind Streaming-Sticks und Single-Board-Computer in der gleichen Leistungsklasse und der Begriff wird eher im Bereich der Low-End-Spiele-Konsolen verwendet. Die OUYA

[56] Vgl. *Riegler, T.*, TV, 2011, S. 123-125.
[57] Vgl. *Miller, P.,* Raspberry Pi, 2017.
[58] Vgl. *Miller, P.,* Raspberry Pi, 2017.

sollte energiesparend in einem kompakten Format Echtzeit-Computergrafiken darstellen können in Spielen und zum Streamen geeignet sein ähnlich der teureren Konkurrenz von Microsoft und Sony. Das Herzstück ist der Tegra 3 SoC mit all seinen Vor- und Nachteilen. Der verbaute Prozessor der OUYA entspricht nicht der Grafikleistung einer Xbox 360 oder einer Playstation 3. Kann aber durchaus mit der Leistung einer Playstation 2, der ersten Xbox und einer Nintendo Wii verglichen werden. Größter Unterschied zu den Spielekonsolen aus den Jahren 2006 und früher ist die Hardware-Unterstützung für MP4 und 1 GB RAM. Diese Konsole wird nur noch gebraucht verkauft und nicht mehr produziert. Dadurch unterscheidet sie sich von dem Raspberry Pi und dem Amazon FireTV Stick. Darüber hinaus wurde zu großen Teilen der Software-Support eingestellt. Als im Handel aktuell erhältlich kann das NVIDIA ShieldTV bzw. die Nintendo Switch angesehen werden. Allerdings unterstützten diese Geräte weitaus mehr Standards und entsprechen auf Grund ihrer hohen Grafikleistung nicht den hier beschriebenen Mikrokonsolen. Die OUYA läuft mit einem Vorgänger von AndroidTV, einer speziell modifizierten Version von Android 4.1.[59]

4.2 Anwendung auf einer Mikrokonsole

Welche speziellen Beschränkungen und Herausforderungen sich nun aus der beschriebenen Hardware ergeben, wird im Folgenden erläutert. Laut der Definition unterstützt eine Mikrokonsole mindestens OpenGL ES 2.0. Was die Anwendung insofern einschränkt, dass der Grafikspeicher nicht mit neueren Standards wie ASTC komprimiert werden kann. Dadurch ergeben sich für das Laden von Texturen Nachteile. Einerseits sind die Mikrokonsolen schon eingeschränkt im verfügbaren Arbeitsspeicher, andererseits verbrauchen sie bei gleicher visueller Qualität mehr Arbeitsspeicher. Darüber hinaus bedeutet dies für Texturen auch, dass sie unter Umständen erst umgewandelt werden müssen, weil sie vom Festspeicher in den Arbeitsspeicher bzw. Grafikspeicher gelangen müssen. Dies benötigt ebenfalls weitere Rechenzyklen und damit Zeit. Ein Möglicher Kompromiss, der in so einem Fall eingesetzt werden kann, ist die Verkleinerung der Ausmaße von Texturen. Also die Pixelauflösung. Dies hat natürlich visuelle Einbußen zur Folge, würde aber ermöglichen, dass das Echtzeit-Machinima dennoch abgespielt werden kann. Des Weiteren bedeutet eine schwächere Grafikleistung, also dem Unvermögen der hohen Anzahl an

[59] Vgl. Ouya, Ouya, 2012-13.

Fließkommaoperation, wenn man so will, dass die Anzahl der 3D Objekte eingeschränkt sein kann bzw. eingeschränkt werden muss. Bei der Erstellung des Machinimas müsste also darauf geachtet werden, dass z.B. nur eine beschränkte Anzahl an Protagonisten in einer Szene zu sehen ist oder aber auch, dass die Kameraperspektive so gewählt wird, dass keine hohe Sichtweite notwendig ist oder von der Geometrie her komplexe Objekte möglichst nicht gleichzeitig in die Sichtachse geraten. Hier ist also ein Zusammenspiel zwischen Inszenierung der Story und der 3D-Engine-Optimierung für einen optimalen „Filmgenuss“ von Nöten.

Je nach spezifischer Plattform kann es weitere Einschränkungen oder Hürden geben, wenn es um die Verbreitung des Echtzeit-Machinimas und deren Abspiel-Software geht. Der Abspieler sollte daher ebenfalls eine übliche Größe für eine Abspielsoftware nicht überschreiten. Auch um für mögliche potentielle Nutzer zusätzlich attraktiv zu sein im Sinne der Vermarktung.

Zum Vergleich sind ausgewählte VoD-Anbieter, VLC, K-Lite-Codec, sowie die Animationssoftware Plotagon in Bezug auf ihre Downloadgröße in der untenstehenden Tabelle 4 aufgeführt. Diese Auswahl hat keinen Anspruch auf Vollständigkeit. Da auf Android-Geräten die Installationsgröße laut Playstore variiert, wurden zum Vergleich die Dateigrößen aus dem itunes-Store in den meisten Fällen genutzt. Es wurde mit Windows 10 Pro 64 Bit durch einen Chrome-Browser am 09.01.2019 auf diese Daten zugegriffen.

Tabelle 4: Datengröße von Multimedia-Apps

Anbieter	**Store**	**Typ**	**Größe**
Netflix	itunes	VoD-Plattform	80,5 MB
VLC for Mobile	itunes	Player	140,1 MB
K-Lite-Standard	codecguide	Codec-Pack	37,9 MB
Plotagon Story	itunes	Anwendung	367,1 MB
Plotagon Story	Playstore	Anwendung	100,0 MB
Amazon Prime Video	itunes	VoD-Plattform	126,8 MB
YouTube	itunes	VoD-Plattform	241,8 MB
Chrome	itunes	Browser	107,6 MB

Idealerweise sind Download-Apps auf Android nicht größer als 100 MB.[60] Dies ist eine Empfehlung von Google. Weitere Daten können dann erst nach Erstinstallation nachgeladen werden. Dies kann bei Nutzern von mobilen Geräten ohne Internetzugriff zu Einschränkungen führen. Dies kann jedoch bei stationären Mikrokonsolen, welche in erster Linie auch zum Streamen anderer Inhalte den Zugriff auf das Internet benötigen, als zweitranging betrachtet werden.

4.3 Weitere Technologien und Ausblick

In der Unterhaltungsindustrie erzielen Computerspiele und Filme gleichermaßen hohen Umsatz. Darüber hinaus sind Film-Produktionen und die Erstellung von Computerspielen kostspielig. Streaming-Services wie Netflix gehen in die Eigenproduktion, auch für Animationsfilme. Unter anderem sind auch Filme und Serien wie „Red versus Blue“ zu finden, welches ein Machinima-Projekt war.

Ohne das G5-Netz werden neuartige Methoden zur Datenkompression benötigt, um diesen Ansprüchen Genüge zu tun. Die Anwendung durch Machinima-Methoden kann auf Mikrokonsolen gewährleistet werden. In wie weit sich diese Technologie etablieren kann, ist abzuwarten.

Ist eine weite Verbreitung des Players erwünscht und angedacht, muss eine hohe Kompatibilität vorhanden sein und eine geringe Dateigröße erreicht werden, um den Ersteinstieg weiter zu erleichtern.

Mögliche Strategie zur Verbreitung in Hinblick auf die Fortführung dieses Projektes:

Unabhängig von der Werbung, also Promotion in Social-Media, Funk und Presse ist schon in der Konzeption der Software zu berücksichtigen, dass sie eine möglichst geringe Hürde für den Endverbraucher bietet.

- Kleine Dateigröße (zwei Vorteile) schneller Download, wenig Grund zur Deinstallation aus Platzmangel auf dem Gerät
- Kompatibel mit älteren Geräten (z.B. Open GL ES 2.0)
- Frei von Rechten, damit weiter nutzbar, daraus erfolgt ein Mehrwert
- Für Mikrokonsolen optimiert (Gamepad-Navigation), dadurch Alleinstellungsmerkmal gegenüber z.B. Plotagon Story

[60] Vgl. Google Developers, APK, 2018.

5 Zusammenfassung

Diese Arbeit untersucht die Anwendung von Machinima-Methoden zur Datenreduktion beim Storytelling auf einer Mikrokonsole.

Als Erstes wurden hierfür in einer ausführlichen Einleitung die Vorgehensweise und der Aufbau sowie die Zielsetzung beschrieben.

Zunächst wurde der Forschungsgegenstand in Kapitel 2 exakt definiert. Ebenso wurden die historischen Hintergründe und die Entstehung sowie die Abgrenzung von Machinima zu verwandten Genres kurz erläutert. Es folgte eine genaue Beschreibung der Merkmale von Machinima-Methoden: d.h. von Echtzeit-Rendering, dem Anwenden einer Game-Engine und dem Gebrauchen der im Spiel bereits enthaltenen Assets. Insbesondere auf die Rolle des Replays als Machinima-Methode wurde detaillierter eingegangen. Auch auf die bereits bei Netzwerk-Mehrspieler-Spielen und anderen Animationsfilmproduktionen verwendeten Machinima-Methoden sowie dem Lizenzproblem von Machinima wurde in Kapitel 2 eingegangen.

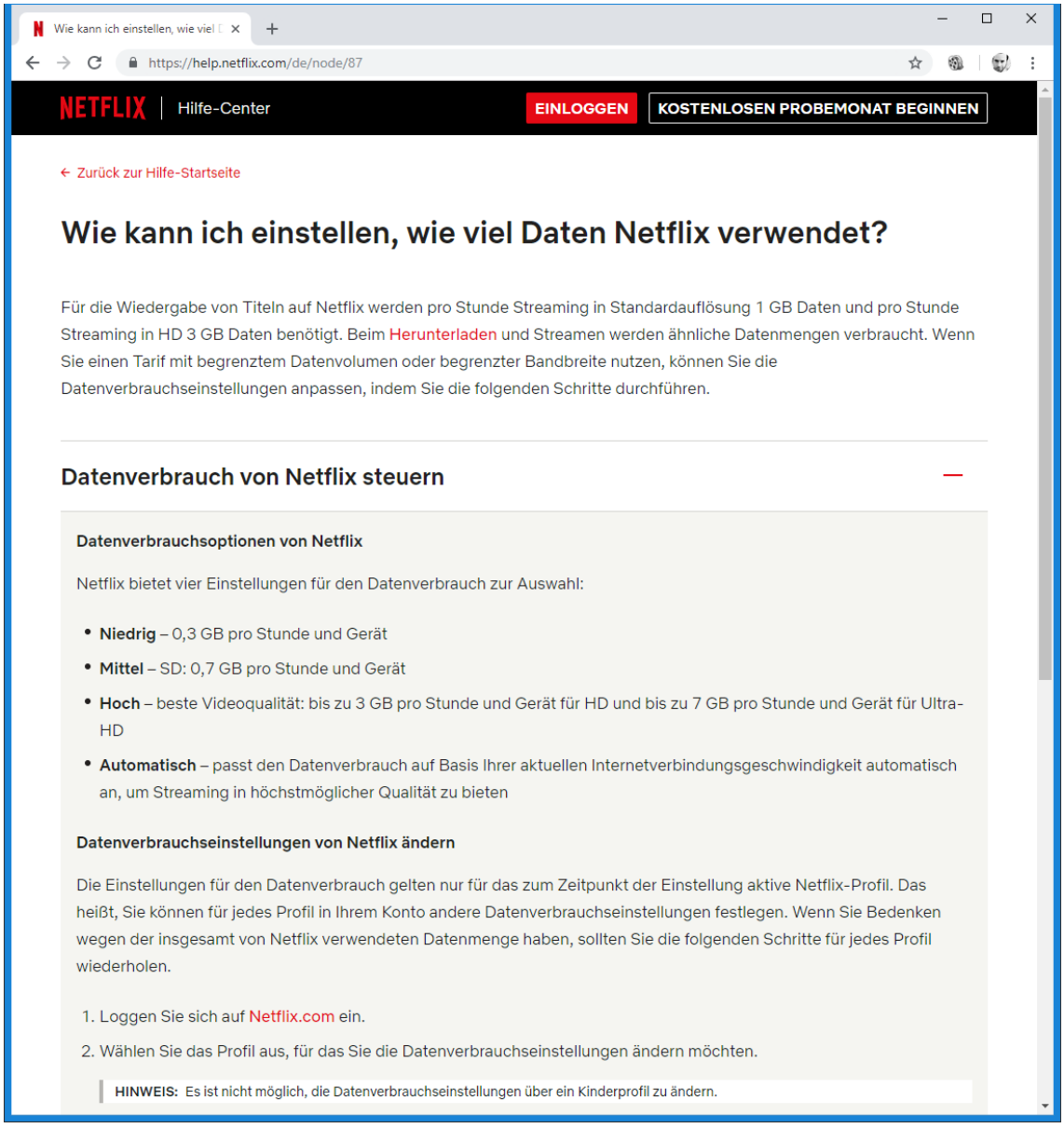
Den Schwerpunkt der Arbeit bildet die Bedeutung der Machinima-Methoden für die Datenreduktion, welche eingehend in Kapitel 3 behandelt wurde. Hier wurden Unterschiede und Vorzüge zwischen Machinima-Methoden und alternativen Kompressionsformaten dargelegt. Ausdrücklich der Unterschied zum Kompressionsverfahren für den Videostream wurde eingehender erklärt.

Im Anschluss wurde in Kapitel 4 die Anwendung der Machinima-Methoden zur Datenreduktion auf einer Mikrokonsole untersucht. Hierfür musste zunächst definiert werden, was unter einer Mikrokonsole zu verstehen ist. Auch Probleme bei der Umsetzung auf einer Mikrokonsole wurden berücksichtigt. Schlussendlich wurde ein Ausblick auf weitere Technologien und Möglichkeiten der Machinima-Methoden zur Datenreduktion gegeben. Eine vollständige Vermarktungsstrategie hätte allerdings den Rahmen dieser Arbeit überstiegen. Auch eine vollständige Kostenanalyse zur Erstellung war ebenfalls nicht Bestandteil dieser Arbeit.

Schlussendlich kann gesagt werden, dass das Anforderungsprofil an eine solche Software grob beschrieben wurde und diese Untersuchung als Idee für einen möglichen Prototypen verstanden werden kann.

Anhang und Anlagen

Abbildung 9: Netflix-Datenraten



Quellcode zur Zusammenfassung von Daten

```
; FL 2016-10-17

Structure OwnType
  type$

  count.l ; 4 Byte
  size.q  ; 8 Byte
  maxsize.q   ; bigest file of this type
  maxname$; name of the bigest file of this type

EndStructure

NewList folders.s()

NewList OwnFiles.OwnType()

Directory$ = GetHomeDirectory() ; Listet alle Dateien und Ordner im 'Home'-Verzeichnis auf

Procedure sucheMusikdateien(pfad.s, List Liste.s(), typ.s="")
   PathAddBackslash_(@pfad) ; Hängt Backslash an, falls nötig
   Protected dir=ExamineDirectory(#PB_Any, pfad, ""), n
   If dir
     While NextDirectoryEntry(dir)
       If DirectoryEntryType(dir) = #PB_DirectoryEntry_File
```

```
      ElseIf DirectoryEntryName(dir) <> "." And DirectoryEntryName(dir) <> ".."
        AddElement(Liste())
        Liste()=pfad+DirectoryEntryName(dir)
        sucheMusikdateien(pfad+DirectoryEntryName(dir), Liste(), typ)
      EndIf
    Wend
    FinishDirectory(dir)
  EndIf
EndProcedure


Path$="C:\F\The Game Creators\FPS Creator\"
Path$="C:\Users\InnoCampus\AppData\Local\AGKApps\FL_LoadAll\media\FPSC_MP_Extracted\"
Path$="D:\F\The Game Creators\FPS Creator\"
sucheMusikdateien(Path$,folders()) ;
AddElement(folders())
folders()=Path$
ForEach folders()
  Directory$ = folders()
  If ExamineDirectory(0, Directory$, "*.*")
    While NextDirectoryEntry(0)
      If DirectoryEntryType(0) = #PB_DirectoryEntry_File
        ;Type$ = "[File] "
        typ$=LCase(GetExtensionPart(DirectoryEntryName(0)))
```

```
 notinlist = 1
 ForEach OwnFiles()
  If OwnFiles()\type$=typ$
    notinlist = 0
    OwnFiles()\size=OwnFiles()\size+DirectoryEntrySize(0)
    If OwnFiles()\maxsize < DirectoryEntrySize(0)
      OwnFiles()\maxsize = DirectoryEntrySize(0)
      OwnFiles()\maxname$ = DirectoryEntryName(0)
    EndIf
    OwnFiles()\count = OwnFiles()\count + 1
    Break
  EndIf
 Next
  If notinlist
   AddElement(OwnFiles())
   OwnFiles()\type$=typ$
   OwnFiles()\size=DirectoryEntrySize(0)
   OwnFiles()\count = 1
   OwnFiles()\maxname$=DirectoryEntryName(0)
   OwnFiles()\maxsize = DirectoryEntrySize(0)
 EndIf

Else
 ; Type$ = "[Directory] "
 ; Size$ = "" ; A directory doesn't have a size (Add Size from all Files ...)
```

```
        EndIf
    ; Debug Type$ + DirectoryEntryName(0) + Size$
   Wend
   FinishDirectory(0)
 EndIf
Next

SortStructuredList(OwnFiles(),#PB_Sort_Descending,OffsetOf(OwnType\size),
TypeOf(OwnType\size))

 ForEach OwnFiles()
  Debug          OwnFiles()\type$+Chr(9)+Str(OwnFiles()\count)+Chr(9)+Str(Own-
Files()\size)+Chr(9)+Str(OwnFiles()\maxsize)+Chr(9)+OwnFiles()\maxname$

 Next
  Debug ListSize(OwnFiles())
 Debug "-------"

; IDE Options = PureBasic 5.62 (Windows - x64)
; CursorPosition = 52
; Folding = -
; EnableXP
; EnableUnicode
```

Tabelle 5: Auswertung FPSCC

Typ	Anzahl	Größe	Größte	Name
dds	2166	612.181.544	8.388.736	gun_D.dds
x	1330	135.537.471	3.895.413	aiko_L.X
exe	12	79.095.561	25.667.406	FPSC-Game.exe
wav	374	72.094.037	10.576.868	title.wav
pdf	3	49.813.082	37.987.217	FPS Creator Manual.pdf
bmp	1832	43.568.962	5.760.054	splash.bmp
tga	34	35.217.774	3.146.267	scope.tga
zip	4	32.277.310	22.621.610	Entity_Maker_v1.1.zip
dbo	25	10.046.863	4.281.376	universe.dbo
dll	6	5.020.672	3.280.896	GameCreatorStore.dll
fpm	38	4.464.127	129.253	scifi-2.fpm
avi	1	4.101.120	4.101.120	storyvideo.AVI
png	124	3.656.479	489.505	backdrop.png
jpg	18	3.490.557	769.397	gamemenu_background.jpg
ogg	27	1.216.739	886.832	TerrorStrike.ogg
fps	316	1.196.635	12.499	Chateau Study Room Full.fps
fpol	28	1.154.160	41.220	armoury large.fpol
dbu	1	1.085.488	1.085.488	universe.dbu
wmv	1	909.517	909.517	intro.wmv
dat	6	606.366	282.420	oldmap.dat
fpe	581	513.714	2.701	AI (Unarmed).fpe
fpmb	29	309.956	282.420	map.fpmb
fpmo	29	309.956	282.420	map.fpmo
fx	42	275.830	16.693	nightvision.fx
txt	130	221.866	32.341	GameCreatorStore.txt
ini	17	155.352	25.405	fpsci-050.ini
db	3	153.600	141.824	Thumbs.db
fpi	206	153.337	10.066	main-weapon.fpi
rtf	1	104.022	104.022	addendum.rtf
bin	11	99.297	17.541	Hanger Bay Full.bin
fpml	1	41.220	41.220	map.fpml
obs	20	29.630	8.254	map4.obs
3ds	2	25.570	13.543	bed_a.3DS
fpg	6	20.630	3.531	scifigame.fpg
ico	4	15.096	3.774	artillery_gun_d2.ico
loc	908	13.962	38	Bunker Passageway Corridor TJunction.loc
ele	2	12.514	6.257	universe.ele
fpp	28	6.225	530	chateau.fpp
	2	4.505	2.685	entries
svn-base	15	3.923	388	nightvision -toggle.fpi.svn-base
lipsync	12	3.286	658	thug.lipsync
manifest	6	2.268	378	CountPolygons.exe.manifest
lnk	1	2.113	2.113	wav2lipsync silent.lnk
fpid	5	2.051	731	setuplevel.fpid
cur	2	1.036	518	MOVE4WAY.CUR
cfg	2	250	125	cfg.cfg
bat	1	225	225	wav2lipsync.bat
seg	1	201	201	map.seg
log	1	200	200	memusedtable.log
ent	1	150	150	map.ent
gx9	1	138	138	script.gx9
lgt	1	52	52	universe.lgt
way	1	8	8	map.way
eff	1	6	6	universe.eff

Abbildung 10: Kenney-Assets

Abbildung 11: FPSCC-Texturen-Auszug

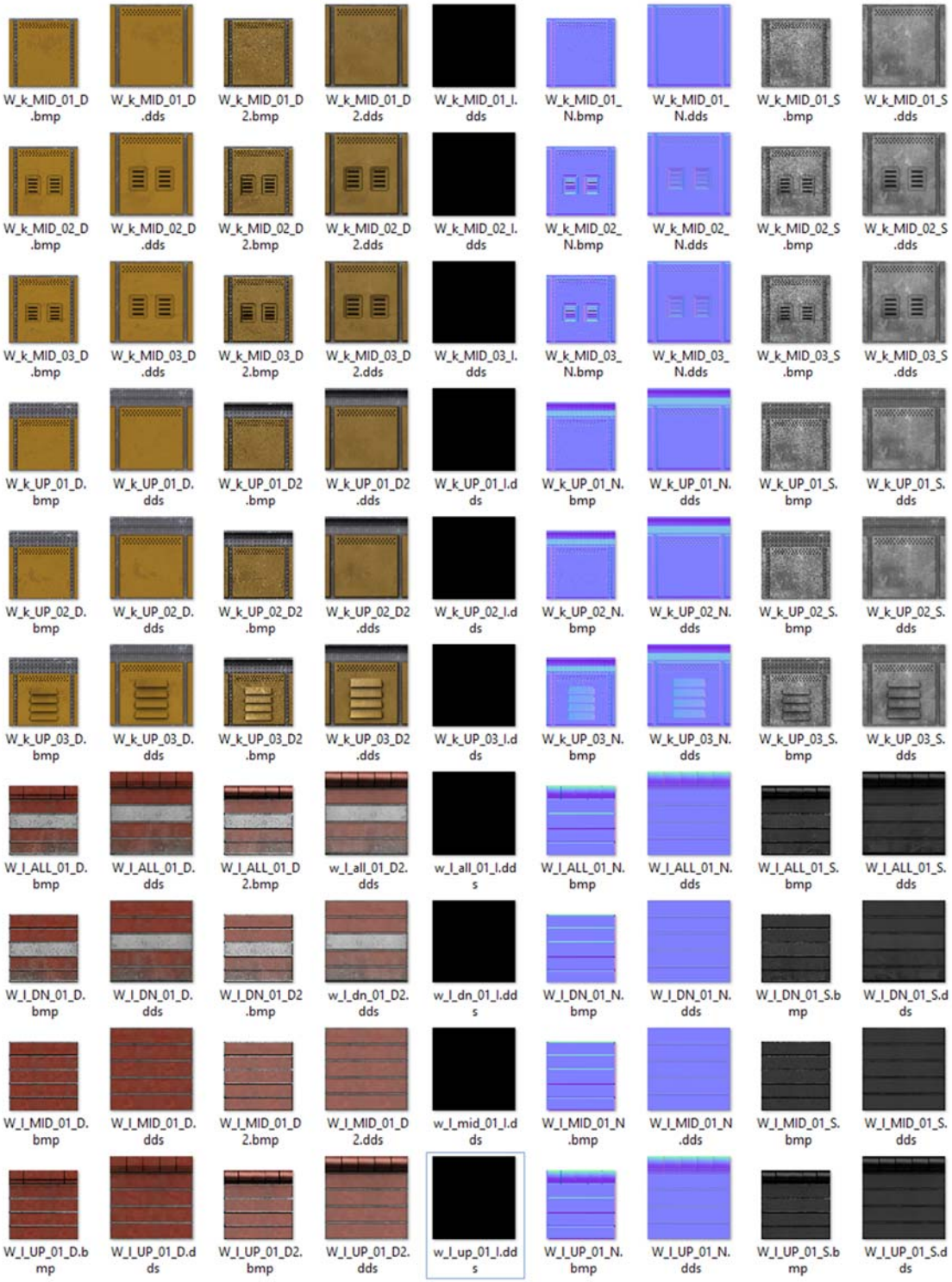


Abbildung 12: Darstellung der verschiedenen Charaktere durch den Wechsel einer Textur

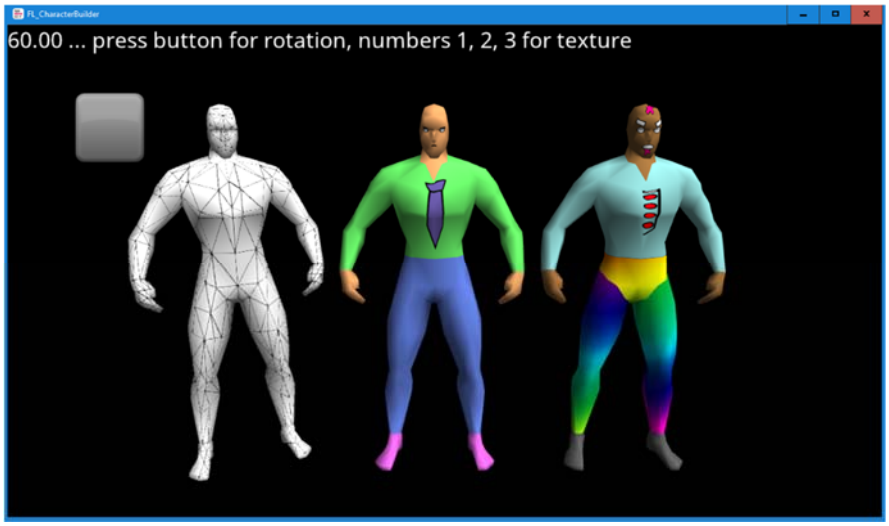


Abbildung 13: In Abbildung 12 verwendete Texturen

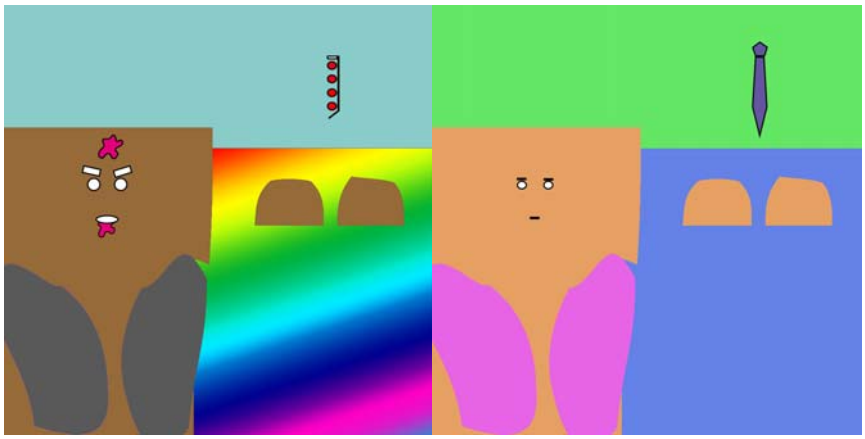
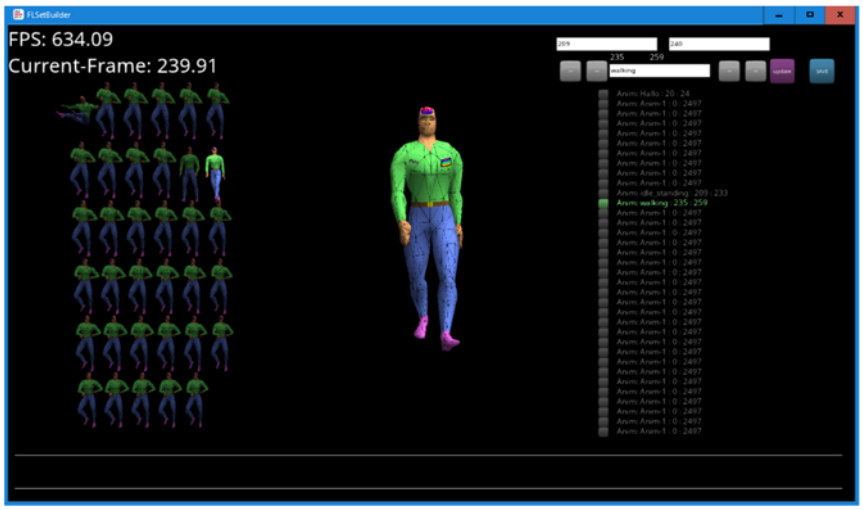


Abbildung 14: Darstellung eines Charakters mit verschiedenen Animationen



Literaturverzeichnis

Deutsches Filmmuseum (Hrsg.) (Film und Games, 2015): Film und Games: Ein Wechselspiel, Deutsches Filmmuseum, Frankfurt am Main 1. Juli 2015 bis 31. Januar 2016, Berlin: Bertz & Fischer, 2015

Klein, Thomas (Let's-Play-Videos, 2015): Let's-Play-Videos und Gaming Culture, in: *Deutsches Filmmuseum* (Hrsg.), Film und Games: Ein Wechselspiel, Deutsches Filmmuseum, Frankfurt am Main 1. Juli 2015 bis 31. Januar 2016, Berlin: Bertz & Fischer, 2015, S. 198-203

McCloud, Scott (Comics, 2001): Comics richtig lessen, Hamburg: Carlsen, 2001

Meyers Lexikonredaktion (Hrsg.), Claus, Volker, Schwill, Andreas (wiss. Bearb.) (Duden Informatik, 1997): Schülerduden Informatik, 3. bearb. Aufl., Mannheim u.a.: Dudenverlag, 1997

Müller-Michaelis, Jan (Computerspiel Erzählform, 2006): Das Computerspiel als nichtlineare Erzählform: Entwicklung und Umsetzung eines dramaturgischen Konzepts, Diplomarbeit im Studiengang Medientechnik an der Hochschule für Angewandte Wissenschaften Hamburg, 2006

Nitsche, Michael (Potenzial Machinima, 2015): Das Potenzial von Machinima, in: *Deutsches Filmmuseum* (Hrsg.), Film und Games: Ein Wechselspiel, Deutsches Filmmuseum, Frankfurt am Main 1. Juli 2015 bis 31. Januar 2016, Berlin: Bertz & Fischer, 2015, S. 106-113

Ohm, Jens-Rainer (Bildcodierung, 1995): Digitale Bildcodierung: Repräsentation, Kompression und Übertragung von Bildsignalen, Berlin, Heidelberg, New York u.a.: Springer, 1995

Post, Uwe (Android, 2015): Android-Appsentwickeln für Einsteiger, Bonn: Rheinwerk, 2015

Riegler, Thomas (TV, 2011): Cyber-TV: Hybridtechnik – Fernsehen und Internet, Baden-Baden: Verlag für Technik und Handwerk, 2011

Rodewald, Vera Marie (Machinima, 2015): Machinima: Das Computerspiel als Filmkulisse, in: *Deutsches Filmmuseum* (Hrsg.), Film und Games: Ein Wechselspiel, Deutsches Filmmuseum, Frankfurt am Main 1. Juli 2015 bis 31. Januar 2016, Berlin: Bertz & Fischer, 2015, S. 194-197

Schmitt, Lutz (Machinima, 2006): Machinima: Medium und Technologie, Diplom Nebenthema: Auseinandersetzung mit Game Engines als Mittel zur Erstellung von Filmen. 2005, veröffentlicht unter einer Creative Commons-Lizenz (cc-by-nc-sa 2.0)

Seifert, Carsten, Wislaug, Jan (Unity, 2017): Spieleentwickeln mit Unity 5: 2D- und 3D-Games mit Unity und C# für Desktop, Web und Mobile, München: Hanser, 2017

Steppart, Michael (Audio, 2014): Audio-Programmierung: Klangsynthese, Bearbeitung, Sounddesign, o.O.:Hanser, 2014

Internetquellen

Bettinger, Brendan (Pixar, 2012): Pixar by the Numbers – From Toy Story to Brave (24.06.2012), http://collider.com/pixar-numbers-toy-story-brave/ (Zugriff 06.01.2019, 13:05 MEZ)

Chait, David (ASTC, 2018): Using ASTC Texture Compression for Game Assets, (ohne Datumsangabe), https://developer.nvidia.com/astc-texture-compression-for-game-assets (Zugriff 03.01.2019, 18:00 MEZ)

Dutt, Arjun (3D Motion Capture, 2018): Radically Simple: AI Startup Makes 3D Motion Capture a Breeze for All, (08.05.2018), https://blogs.nvidia.com/blog/2018/05/08/radical-3d-motion-capture/ (Zugriff 09.01.2019, 10:00 MEZ)

Gilbert, Ron (Maniac, 2011): The Making of Maniac Mansion, Blogeintrag (09.01.2011), https://grumpygamer.com/making_of_maniac_mansion, (Zugriff 07.01.2019, 15:55 MEZ)

Google Developers (APK, 2018): APK Expansion Files, (20.12.2018), https://developer.android.com/google/play/expansion-files (Zugriff 02.01.2019, 17:00 MEZ)

Google Developers (Distribution, 2018): Distribution dashboard, (11.12.2018), https://developer.android.com/about/dashboards/ (Zugriff 10.01.2019, 01:13 MEZ)

Miller, Paul (Raspberry Pi, 2017): Raspberry Pi sold over 12.5 million boards in five years, (17.03.2017), https://www.theverge.com/circuitbreaker/2017/3/17/14962170/raspberry-pi-sales-12-5-million-five-years-beats-commodore-64 (Zugriff 05.12.2018, 15:00 MEZ)

NVIDIA-Corporation (Tegra 3, 2018): NVIDIA-Tegra-3-Mobilprozessoren (keine Datumsangabe 2018), https://www.nvidia.de/object/tegra-3-de.html (Zugriff 07.01.2019, 18:00 MEZ)

Online-Marketing-Praxis (Storytelling, o.J.): Definition Storytelling, (keine Datumsangabe), https://www.onlinemarketing-praxis.de/glossar/storytelling (Zugriff 01.10.2018, 14:00 MEZ)

Ouya (Ouya, 2012-13) Ouya: A New Kind of Video Game Console, (keine Datumsangabe), https://www.kickstarter.com/projects/ouya/ouya-a-new-kind-of-video-game-console?lang=de (Zugriff 05.12.2018, 16:00 MEZ)

Pimentel, Ken (ZAFARI, 2018): Animated Children's Series ZAFARI Springs to Life with Unreal Engine (10.01.2018), *https://www.unrealengine.com/en-US/spotlights/animated-children-s-series-zafari-springs-to-life-with-unreal-engine* (Zugriff 05.01.2019, 10:15 MEZ)

Pixar Animation Studios (USD, 2017): Universal Scene Description, (keine Datumsangabe), https://graphics.pixar.com/usd/docs/index.html (Zugriff 06.12.2018, 16:00 MEZ)

Sixtus, Mario (Hollywood, 2008): Hollywood in 64 Kilobyte, in: Elektrischer Reporter Staffel 02, Episode 02, Blinkenlicht Produktionen im Auftrag des ZDF 2008, https://www.youtube.com/watch?v=L42Xm-Zjv94 (Zugriff 06.01.2019, 20:00 MEZ)

Quellenverzeichnis

Spiele

Fortnite
Entwickler: Epic Games
Verlag: Epic Games
Veröffentlichungsjahr: 2017

Maniac Mansion
Entwickler: Ron Gilbert und Gary Winnick
Verlag: Lucasfilm Games
Veröffentlichungsdatum: 05.10.1987

Minecraft: Story Mode
Entwickler: Telltale Games, Mojang: Stephen McManus
Verlag: Telltale Games
Veröffentlichungsdatum: 13.10.2013 – 13.09.2016

StarCraft 2: Legacy of the Void
Entwickler: Blizzard Entertainment
Verlag: Blizzard Entertainment
Veröffentlichungsdatum: 12.03.2013

Filme

Red vs Blue
16 Staffeln, 318 Folgen und 34 Folgen Miniserie
Veröffentlichungsdatum: 01.04.2003 – heute
Produzent: Rooster Teeth Productions
Website: https://roosterteeth.com/series/red-vs-blue
Game-Engine: Halo 1 und 2
Bild: Game Artwork
Ton: Original Voices und Game Sounds

ZAFARI
Veröffentlichungsdatum: 05.02.2018 – heute
Produktionsfirma: Zafari Holdings
Direktion: David Dozoretz
Vertrieb: Nbc Universal Dreamworks
Engine: Unreal
Website: http://zafari.com/

debris
Veröffentlichungsdatum: 2007
Produzent: Farbrausch
Webseite: http://www.pouet.net/prod.php?which=30244

Software und Sonstiges

3D-Animationssoftware
Von Plotagon AB
https://plotagon.com/

Common-Voice-Projekt
Von Mozilla
Gemeinfreie Sprachsynthese
https://voice.mozilla.org/de

Furniture Kit
Von Kennedy
Asset-Sammlung
https://www.kenney.nl/assets/furniture-kit

https://gflops.surge.sh/

https://lotv.spawningtool.com/34103/

Tracker
Zum Abspielen von Musikbibliothek
http://www.hivelytracker.co.uk/tunes/

FPS Creator
Game Engine inklusive Assets
Von The Game Creators
Quelle der Daten für 45Tabelle 5
https://github.com/TheGameCreators/FPS-Creator-Classic

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit einverstanden/nicht ein-verstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Berlin, 10.01.2019 (Eigenhändige Unterschrift)