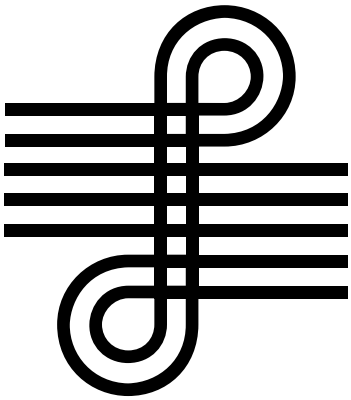
Hochschule für Musik Karlsruhe

IMWI - Institut für Musikinformatik und Musikwissenschaft



Bachelorarbeit zum Thema:

Interaktion mit Audio-Daten in VR mithilfe von Unity, Head-Mounted-Display und Motion-Controllern

Zur Erlangung des Grades Bachelor of Arts

Vorgelegt von:
Manuel-Philippe Hergenröder, 12085
mail@manuelhergenroeder.de

Betreuender Dozent:
Prof. Dr. Damon T. Lee

Studiengang:
Musikinformatik (Hauptfach) / Musikwissenschaften (Nebenfach)

Abgabe:
xx.07.2020

1 Inhaltsverzeichnis

2 Einleitung 3

3 Grundlagen 3

3.1 Virtual Reality 3

3.1.1 Definition, Historie und Abgrenzung 3

3.1.2 Anforderungen an VR-Software und -Hardware 4

3.2 Diskrete Fourier-Transformation 7

4 Vorstellung VrAudioSandbox 8

4.1 Architektur und externe Bibliotheken 8

4.1.1 Unity als Laufzeitumgebung 9

4.1.2 SteamVR / OpenVR 9

4.1.3 NAudio Bibliothek 9

4.1.4 DFT mit FFTW3 10

4.1.5 FFTWSharp 11

4.2 Implementation 11

4.2.1 Audio Engine und FFT 11

4.2.2 Visualisierung der Spektrum-Daten 12

4.2.3 User Interface und Steuerung 13

4.2.4 Hardwareanforderungen und Performanz 13

5 Kritische Betrachtung der Ergebnisse 13

6 Fazit 13

7 Literatur 14

8 Sonstige Quellen 15

9 Abbildungsverzeichnis 16

10 Eidesstattliche Erklärung 17

11 Anhang – Programmcode 18

2 Einleitung

Virtual Reality (VR) bietet das Potential unserem Geist direkten und erlebbaren Zugriff auf digitale Daten – losgelöst von der Umgebung in der wir uns aufhalten – zu geben. In dieser Arbeit sollen die Möglichkeiten der Interaktion mit Audio-Daten im virtuellen Raum am Beispiel der Darstellung und Manipulation von FFT-Audio-Daten mithilfe eines *Head-Mounted-Display* (engl. für „Am-Kopf-befestigter-Bildschirm“) – oder kurz *HMD* - und Motion-Tracking-Controllern untersucht werden.[1] Gegenstand der Arbeit ist die praktische Implementation dieses Konzepts und <Herausforderungen, Bewertung, Ausblick>.

3 Grundlagen

3.1 Virtual Reality

3.1.1 Definition, Historie und Abgrenzung

Der Begriff *Virtual Reality* impliziert, dass eine virtuelle Realität mithilfe einer Simulation geschaffen wird, welche für den Nutzer als real wahrgenommen wird. Bereits 1965 beschrieb der Computergrafikpionier Ivan E. Sutherland mit dem „Ultimate Display“ einen Raum, in dem die Materie durch den Computer gesteuert wird. Auf einen Stuhl in diesem Raum könne man sich setzen, eine Pistolenkugel wäre tödlich.[2] Dieses Ziel der kompletten Immersion konnte bisher nicht umgesetzt werden. Die Geschichte der Entwicklung von Technologien zur Annäherung an diese Utopie geht weit zurück.

Gerade im Bereich der Hardware spielte die Entwicklung des sogenannten HMD eine wichtige Rolle. Ein *HMD* besteht aus einem oder mehreren Displays und stellt am Kopf befestigt eine Schnittstelle zwischen der visuellen Wahrnehmung des Menschen und der Grafikausgabe eines Computers her. Ivan E. Sutherland hat 1968 ein solches System mit dem Namen „The Sword of Damocles“ mithilfe seiner Studenten realisiert.[3] Dies legte die Grundlagen zum einen für die stereoskopische Darstellung des im Computer erzeugten Raums in Vektorgrafik, zum anderen für die notwendige schnellen Synchronisierung der angezeigten Bilder mit den Kopfbewegungen des Benutzers. Ein Sensor mit mechanischem Arm hat zunächst die Kopfposition ermittelt. Später wurde dies für bessere Bewegungsfreiheit mithilfe von Ultraschall-Emittern und -

[1] *Anmerkung: Für die Entwicklung wurden das im März 2015 vorgestellte und im April 2016 auf dem Markt verfügbare HMD HTC Vive und die dazugehörigen Motion-Controller verwendet.*

[2] Vgl. Sutherland, Ivan: „The Ultimate Display” – Konferenzband: Information Processing 1965: proceedings of IFIP Congress / Wayne A. Kalenich [Hrsg.]. International Federation for Information Processing, Amsterdam u.a., Washington u.a. (1965), S. 508 – http://worrydream.com/refs/Sutherland%20-%20The%20Ultimate%20Display.pdf, letzter Abruf: 09.06.2020

[3] Vgl. Sutherland, Ivan: „A head-mounted three dimensional display” – AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I, December 1968 Pages 757–764, S. 757 – http://www.medien.ifi.lmu.de/lehre/ss09/ar/p757-sutherland.pdf, letzter Abruf: 10.06.2020

Empfängern umgesetzt.[4] Dieses sogenannte *Head-Tracking* und die zeitnahe Abstimmung der angezeigten Bilder an Kopfposition und -bewegungen ist auch heute noch ein wichtiger Aspekt bei *HMDs*. Einerseits um das Präsenzgefühl überhaupt erst zu erzeugen zwischen virtueller Welt und dem physischen Körperempfinden, andererseits um Übelkeit (sogenannte *Motion-Sickness*), die in unserem Gehirn entsteht, wenn widersprüchliche Reize – in diesem Fall die visuellen Informationen unseres Sehapparates und die körperliche Wahrnehmung bezüglich der Bewegung und Stellung des Kopfes verarbeitet werden – beim Benutzer vorzubeugen.

<NASA Datenhandschuh, VR Hype 90er, 2. Welle Oculus Rift/Vive, Nintendo Virtual Boy, Gaming Industry, Professionelle Anwendungen Architektur, CAD, Ergonomie>

Ein mit der *Virtual Reality* verwandtes Feld ist die sogenannte *Augmented Reality* bei der die echte Realität mit zusätzlichen Informationen angereichert wird. Im Gegensatz zur VR werden reales Umfeld und virtuelle Bestandteile vom Computer vermischt – z.B. durch ein transparentes Display oder mithilfe eines eingeblendeten Live-Kamera-Bildes. Bei einer starken Interaktion zwischen Bestandteilen realer und virtueller Welt wird zudem auch der Begriff *Mixed-Reality* verwendet. AR ist nicht Gegenstand dieser Arbeit.

3.1.2 Anforderungen an VR-Software und -Hardware

Gängige etablierte Bedienparadigma, die auf der der Computer- und Medientechnik basieren und sich über die letzten Jahrzehnte entwickelt und etabliert haben, sind für VR-Anwendungen nicht in Gänze übertragbar. Insbesondere die Gestaltung in Form einer zweidimensionalen Bedienoberfläche (*User Interface*), welche auf einem zweidimensionalen Display dargestellt wird und mit Maus, Tastatur oder via Touch bedient wird, lässt sich nur mit Einschränkungen bzgl. der Ergonomie und Nutzerfreundlichkeit bei Virtual Reality anwenden. Auch die damit einhergehenden verschachtelten Menüstrukturen, die viel Feinmotorik zur Interaktion benötigen, sind nicht für die Bedienung mit *Motion-Controllern* ausgelegt. *Motion-Controller* sind in der Regel Teil eines Virtual-Reality-Konzepts und werden demzufolge häufig in Form eines Gesamtkonzeptes mit HMDs zusammen entwickelt und verkauft. Dies macht deutlich, dass neue Bedienkonzepte – zugeschnitten auf die Bedienung durch Handbewegungen und -gesten – eine große Rolle spielen.

[4] Vgl. Ebd. S. 760-761

Ein Beispiel zur Verdeutlichung sind die in vielen Computerspielen – aber auch in Software fernab des Unterhaltungssektors – vorhandenen *HUD-UI-Elemente*. *HUD* steht für *Head-Up-Display* und beschreibt das Platzieren von informativen Elementen oder Elementen zur Steuerung im sogenannten *Screen-Space* – d.h. überlagernd ohne Bezug zur dreidimensional dargestellten Szene. Durch die stereoskopische Darstellung in VR kann dieser fehlende Bezug zur Geometrie des virtuellen Raumes irritierend wirken. Auch dass der Benutzer sich im Raum orientieren muss und seine Blickrichtung dabei frei wählbar ist, kann dazu führen, dass UI-Elemente im Sichtbereich stören können. Das VR-Computerspiel „Half-Life Alyx“ der Firma Valve zeigt den Ansatz traditionelle HUD-Elemente durch virtuell-physische Elemente zu ersetzen. Gesundheitszustand des Spielers und Munitionsanzeige sind Teil der Spielwelt und können durch entsprechenden Blick des Spielers bei Bedarf erfasst werden ohne zu stören (siehe Abbildung 1 unten, oben die traditionell genutzte Methode durch HUD-Elemente).

Abbildung 1 – Vergleich HUD-basierte Darstellung (oben) von Gesundheit und Munition mit im virtuellen Raum als physische Objekte manifestierter Darstellung (unten)

Einzelquellen (von oben n. unten): Steam Workshop Half-Life: Alyx [HUD], Screenshot von „Alex“ – https://steamuserimages-a.akamaihd.net/ugc/1016065725020200218/B47282DABAFC22E8A6C5770FE46BCA8FCFFBF718/; IGDB Half-Life: Alyx Press kit – https://images.igdb.com/igdb/image/upload/t_original/sc7dad.png; Screenshot von SuperQGS, reddit – https://i.redd.it/yojg6f875rg41.jpg – alle Screenshots © 2019 Valve Corporation, letzter Abruf 17.06.2020

Dadurch, dass der Benutzer seine Blickrichtung jederzeit frei wählen kann bzw. muss, ergeben sich Anforderungen an die Steuerung der Aufmerksamkeit durch das Design der VR-Applikation. Das Cone-Of-Focus-Modell (siehe Abbildung 2) teilt dabei das Umfeld des Spielers in verschiedene Zonen ein. Alle wichtigen Ereignisse sollten sich im Zentrum der Aufmerksamkeit und Blickrichtung des Benutzers abspielen. Sollte es gewünscht sein, dass der Nutzer seine Aufmerksamkeit auf eine andere Zone lenkt, bieten sich visuelle Anhaltspunkte über den primären und sekundären peripheren Sichtbereich an, um den Nutzer zur Änderung seiner Blickrichtung zu motivieren.[5] Diese könnten ein Wechsel der Beleuchtung, Aufblinken bestimmter Elemente oder Animationen sein, die den Blick des Benutzers lenken.

Bei Szenen- oder Positionswechsel bieten sich Fade-Out und Fade-In an, um den Übergang weicher zu gestalten. Schnelle Bewegungen der virtuellen Figur sollten vermieden werden, wenn

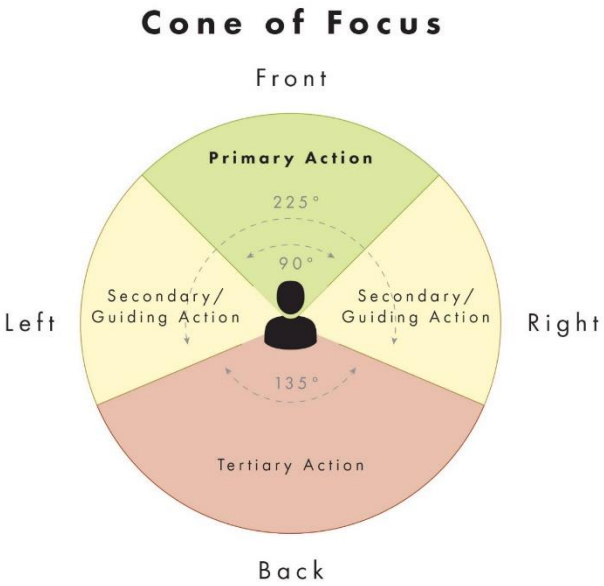


Abbildung 2 – Cone of Focus, UploadVR Copyright 2019 UVR Media LLC – https://mk0uploadvrcom4bcwhj.kinstacdn.com/wp-content/uploads/2016/07/cone-of-focus.jpg, letzter Abruf 17.06.2020

sie nicht mit den Bewegungen des Körpers in der realen Welt korrespondieren. *Room-Scale-VR* ist insofern unproblematischer, dass die virtuellen Bewegungen denen des Benutzers in der realen Welt entsprechen. Es bietet sich dabei ein Warnmechanismus z.B. in Form eines Gitters an, welches bei Annäherung an Hindernisse in der realen Welt eingeblendet wird oder das Einblenden der realen Welt als Echtzeit-Kamerabild.[6]

[5] Vgl. Jerald, Jason (Ph.d.): „The VR Book: Human-Centered Design for Virtual Reality", A publication in the ACM Book series #8, Association for Computing Machinery and Morgan & Claypool Publishers, San Rafael (California) (2016), S. 254
[6] Vgl. Ebd. S. 213

Ein weiterer Aspekt – bei dem insbesondere das Zusammenspiel zwischen Software und Hardware wichtig ist – ist die Latenz zwischen Bewegung (beispielsweise des Kopfes) und der entsprechenden Berechnung des angezeigten Bildes, welche die ausgeführte Bewegung berücksichtigt. Wichtig ist dabei Latenz des gesamten Systems bestehend aus Motion-Tracking, Verarbeitung im Computer (Applikation, Rendering) und die Ausgabe auf dem HMD. Hier spielen auch Faktoren wie die Frequenz des Trackings, Bildwiederholrate des HMD und die Verwendung von Framebuffern eine Rolle. Diese Latenz muss möglichst gering ausfallen, da ansonsten das Präsenz-Gefühl des VR-Nutzers leide und es schnell zu Übelkeit bei der Nutzung kommen könne.[7] Experimente am NASA Ames Research Center[8] legen nahe, dass einerseits Probanden unterschiedlich sensibel auf verschiedene Latenzen reagieren und teilweise bei einem VR-System mit 7.4 ms Latenz noch Unterschiede im Bereich von 3.2 ms wahrgenommen werden konnten. Gängige HMDs (Stand 2015) liegen laut Messungen von Kjetil Raaen und Ivar Kiellmo bei einer Latenz von etwa 40 ms.[9] Auch wenn die Fähigkeiten der eingesetzten Hardware vom Softwareentwickler nur bedingt beeinflusst werden können, sollte die Software auf stabile und hohe Frameraten hin optimiert werden, um Irritationen und Übelkeit zu vermeiden.

3.2 Diskrete Fourier-Transformation

Die diskrete Fourier-Transformation (DFT) ist ein wichtiger Algorithmus in der digitalen Signalverarbeitung und überführt Signal-Daten aus dem Zeitbereich in den Frequenzbereich. Der Frequenzbereich bietet Möglichkeiten Eigenschaften des Signals abzuleiten oder das Signal zu bearbeiten, welche im Zeitbereich nur schwer oder nicht möglich sind. Eine beliebige Wellenform kann in sinoidale Einzelbestandteile aufgespalten werden und zeigt so die Zusammensetzung eines Signals – metaphorisch ähnlich wie bei der Zerlegung von Licht mithilfe eines Prismas. Fast-Fourier-Transform (kurz. FFT) beschreibt dabei eine Untermenge von Implementationen der DFT, die besonders effizient und schnell zu berechnen sind. Die DFT bildet die Grundlage für die Visualisierung und Modifikation der Audio-Daten in VrAudioSandbox (siehe Kapitel 4.2.1).

[7] Vgl. Ebd. S. 183-184

[8] Vgl. Ebd. S. 184-185 – Adelstein et al. 2003 & 2006, Jerald 2009, Ellis et al. 1999, 2004. Mania et al. 2004

[9] Vgl. Raaen, Kjetil; Kjellmo, Ivar: „Measuring Latency in Virtual Reality Systems", 14th International Conference on Entertainment Computing (ICEC), Sep 2015, Trondheim, Norway. pp.457-462, 10.1007/978-3-319-24589-8_40. hal-01758473
Anmerkung: Stand 2015, zwischen 35-45 ms wurden bei der Ocolus Rift DK2 gemessen bei aktivierten V-Sync – d.h. Synchronisierung der Bildwiederholrate mit der Grafikkarte, um Tearing-Effekte zu vermeiden. Dazu wurde eine Unity Szene und eine lichtempfindliche Photozelle verwendet. Latenz durch Eingabegeräte ist nicht berücksichtigt.

4 Vorstellung VrAudioSandbox

4.1 Architektur und externe Bibliotheken

Im Rahmen dieser Arbeit wurde die Möglichkeit der Darstellung und Manipulation von Audio-Daten innerhalb der Virtual Reality praktisch implementiert. Dazu wurde neben den technischen Voraussetzungen ein Bedienkonzept zugeschnitten auf die Motion Tracking Controller der htc Vive entwickelt. Der Benutzer kann beliebiges vorhandenes Audiomaterial importieren, mithilfe des htc Vive HMD und der dazugehörigen Motion-Controller modifizieren und das Ergebnis probehören oder als Audio-Datei exportieren. Die Bedienung findet dabei vollständig in VR statt.

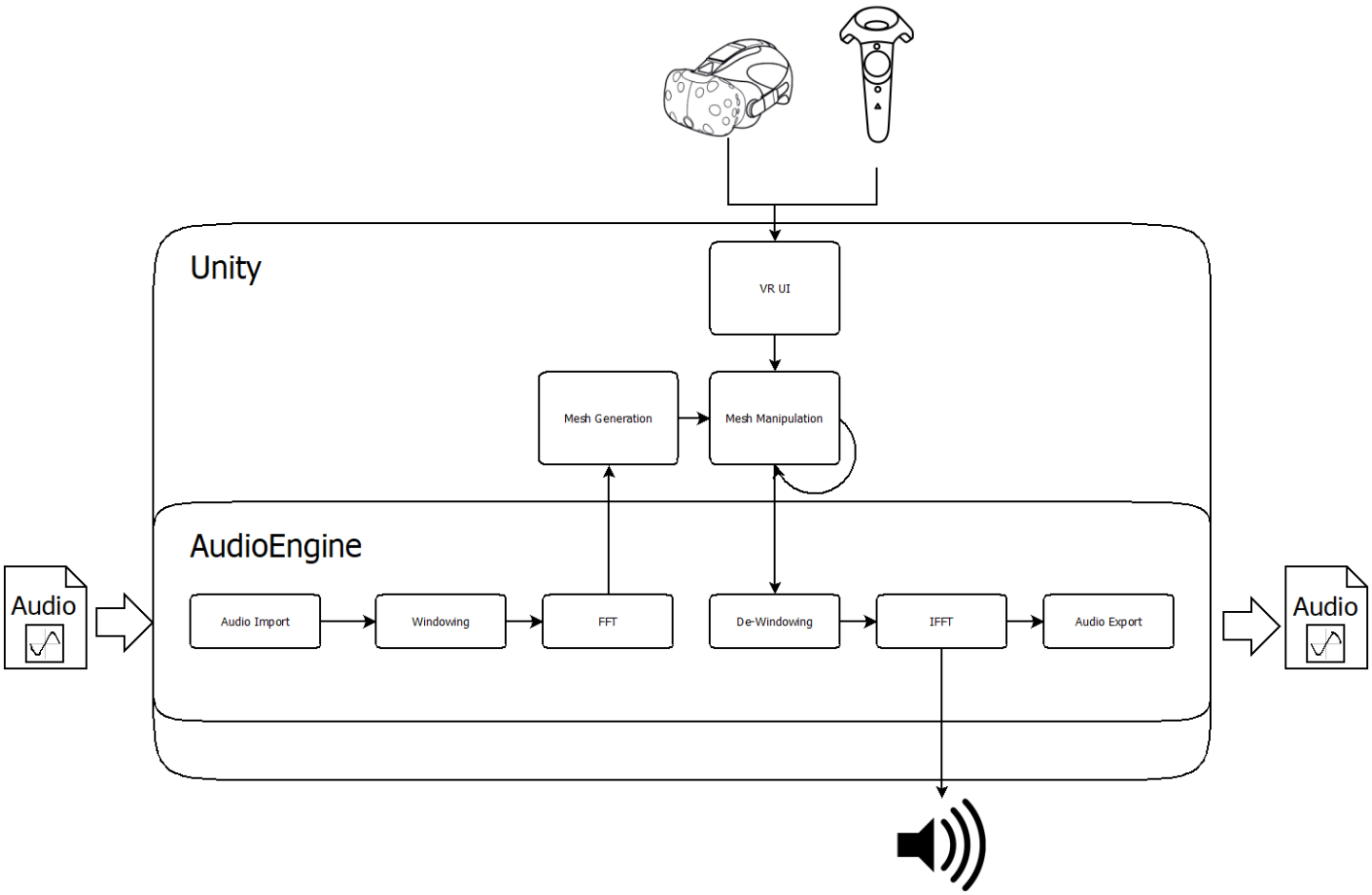


Abbildung 3 – Eine abstrakte Darstellung der Architektur - Eigene Darstellung

Im Mittelpunkt steht die Laufzeitumgebung Unity in der sämtliche Programmlogik in Form von eigenen Klassen implementiert ist. Darüber hinaus werden einige externe Bibliotheken genutzt, die im Folgenden vorgestellt und näher erläutert werden.

4.1.1 Unity als Laufzeitumgebung

Die Laufzeit- und Entwicklungsumgebung Unity ist eine weitverbreitete Spiele-Engine mit etwa 50% Marktanteil für den Bereich PC/Konsole/Mobile Games.[10] Das dazugehörige Unternehmen „Unity Technologies“ (ehemals „Over the edge“) wurde 2004 von David Helgason, Nicholas Francis und Joachim Ante gegründet. Auch außerhalb der kommerziellen Spiele-Entwicklung ist Unity sehr beliebt – u.a. auch da es eine für nicht-kommerzielle Zwecke kostenlose „Personal“-Lizenz gibt, welche den Hauptteil des Funktionsumfangs von Unity bereitstellt.[11] Momentan (Stand: Juni 2020) unterstützt Unity 18 verschiedene Zielplattformen.[12]

Unity bietet eine Vielzahl von vorgefertigten Komponenten im Bereich Grafik, Animation, Audio, Netzwerk, Physik und anderen Bereichen an. Außerdem liefert Unity viele Werkzeuge innerhalb des Unity-Editors mit, die bei der Spieleentwicklung hilfreich sind. Die Kernkomponenten von Unity sind in nativem C++ geschrieben. Die Logik und auch Erweiterung oder Modifikation der mitgelieferten Komponenten kann in der Sprache C# – ausgeführt innerhalb der freien, alternativen und quelloffenen Implementierung von Microsofts .NET Framework „Mono“ – implementiert werden.[13] Alternativ steht das Scripting-Backend IL2CPP (Intermediate Language To C++) für einige Zielplattformen zur Verfügung.

4.1.2 SteamVR / OpenVR

SteamVR ein Sammelbegriff für das Virtual Reality Konzept der Firma Valve und umfasst eine API-Spezifikation (Application Programming Interface), eine Runtime, um mit OpenVR entwickelte Applikationen auszuführen und wird auch als Markenbegriff im Zusammenhang mit eigener VR-Hardware genutzt. OpenVR ist dabei das dazugehörige Software-Development-Kit, das auch auch mit Produkten anderer Hersteller nutzbar ist. In diesem SDK sind bereits viele vorgefertigte Bausteine enthalten – etwa für das Motion-Tracking, Teleportation, etc.

4.1.3 NAudio Bibliothek

NAudio ist eine von Mark Heath entwickelte Open-Source-Audio-API und Bibliothek für .NET und ist in C# geschrieben.[14] Es wird für den Import der Audio-Dateien, die Wiedergabe und den

[10] Vgl. Unity Technologies: „Sie fragen sich was Unity ist? Entdecken Sie, wer wir sind, wo wir angefangen haben und wohin wir uns entwickeln | Unity“ – https://unity.com/de/our-company, letzter Abruf: 11.06.2020
[11] Vgl. Unity Technologies: „Powerful 2D, 3D, VR, & AR software for cross-platform development of games and mobile apps.” – https://store.unity.com/#plans-individual, letzter Abruf: 11.06.2020
[12] Vgl. Unity Technologies: „Multiplatform | Unity“ – https://unity.com/features/multiplatform, letzter Abruf: 24.06.2020
[13] *Anmerkung: In der Vergangenheit konnten alterrnativ die Skriptsprachen Boo und UnityScript verwendet werden. Diese sind aber als veraltet klassifiziert.*
[14] Vgl. Github – NAudio repository – https://github.com/naudio/NAudio, letzter Abruf: 23.06.2020

Export als Audio-Datei genutzt. Darüber hinaus würde NAudio auch die Möglichkeit bieten Audio-Streams zu manipulieren, zu mixen und MIDI-Daten einzulesen und auszugeben.

4.1.4 DFT mit FFTW3

FFTW (Akronym für „The Fastest Fourier Transform in the West") ist eine freie FFT-Bibliothek zur Berechnung der diskreten Fourier-Transformation und wurde von Matteo Frigo und Steven G. Johnson am Massachusetts Institute of Technology entwickelt.[15] Die in C und OCaml geschriebene Bibliothek ist im Quelltext verfügbar, sehr portabel und unterstützt daher viele Plattformen. Außerdem bietet sie laut den Autoren im Vergleich zu anderen Implementationen eine sehr gute Performance.[16]

FFTW bietet eine Vielzahl an Algorithmen und Funktionen an. Für die DFT und inverse DFT in VrAudioSandbox wird auf die „1d Discrete Fourier Transform (DFT)" aus der FFTW3-Biblitohek mit Double-Precision zurückgegriffen, welche durch folgende mathematische Summenformel beschrieben wird:[17]

$$Y_k = \sum_{j=0}^{n-1} X_j e^{-2\pi jk\sqrt{-1/n}}$$

Die Eingabe X ist dabei ein ein-dimensionales Array komplexer Zahlen der Größe n und Y das Ausgabe-Array, wobei das k-te Element der Frequenz k/n entspricht.

Bei der inversen Funktion handelt es sich um die gleiche Funktion ohne negatives Vorzeichen im Exponenten der DFT:

$$Y_k = \sum_{j=0}^{n-1} X_j e^{2\pi jk\sqrt{-1/n}}$$

FFTW berechnet eine unnormalisierte Transformation ohne Koeffizienten vor der Summe der DFT, d.h. eine vorwärtsgerichtete DFT gefolgt von der rückwärtsgerichteten DFT ergibt als Ergebnis die Eingabe multipliziert mit n.

[15] Vgl. Frigo, Matteo; Johnson, Steven G.: "The Design and Implementation of FFTW3" – Proceedings of the IEEE, Volume 93, Number 2, 2005, S. 216-231, S. 231 – http://www.fftw.org/fftw-paper-ieee.pdf, letzter Abruf: 13.06.2020
[16] Vgl. FFTW: „FFT Benchmark Results" – http://www.fftw.org/speed/, letzter Abruf: 13.06.2020
[17] Vgl. FFTW: 3.3.8 Manual, Kapitel 4.8.1 „The 1d Discrete Fourier Transform (DFT)" – http://www.fftw.org/fftw3_doc/The-1d-Discrete-Fourier-Transform-_0028DFT_0029.html, letzter Abruf: 13.06.2020

4.1.5 FFTWSharp

FFTWSharp ist ein Wrapper um die C-Bibliothek FFTW3 aus C# heraus zu nutzen.[18] Es vereinfacht und abstrahiert vor allem den Datenaustausch über „unmanaged“ Arrays (Arrays, die nicht vom Garbage-Collector der .NET-Laufzeitumgebung verwaltet werden) zwischen C# und C.

4.2 Implementation

4.2.1 Audio Engine und FFT

Nachdem der Benutzer eine Audio-Datei für den Import ausgewählt hat, wird diese mithilfe der NAudio-Bibliothek zunächst für die Weiterverarbeitung in ein Array bestehend aus den einzelnen Audio-Frames in Form von 32-Bit-Float-Werten überführt. Je nach Bit-Tiefe der Eingangsdaten werden dabei Bytes der PCM-Rohdaten zusammengefasst– bspw. bei 16-Bit Audio ergeben 2 Bytes ein Audio-Frame, bei 24-Bit-Audio 3 Bytes etc. Außerdem muss beachtet werden, ob die Daten in Big-Endian- oder Little-Endian-Reihenfolge gespeichert sind, d.h. ob das höchstwertige Bit an erster oder letzter Stelle positioniert ist. Da NAudio die Ausgabe von Rohdaten nur als 32-Bit-Float-Array unterstützt, findet beim Import von Audio-Daten mit 64 Bit ein Verlust der Genauigkeit bzw. der entsprechenden Nachkommastellen statt.

Bevor die Audio-Daten der FFT zugeführt werden, wird die FFT-Size abhängig von der Sampling-Rate festgelegt. Bei 44.100 kHz hat sich eine FFT Size von 1024 als guter Kompromiss zwischen temporaler Auflösung, frequenzbezogener Auflösung und Rechenleistung für Berechnung und Visualisierung herausgestellt. Dieses Verhältnis zwischen Sampling-Rate und FFT-Size wird auch bei anderen Sampling-Raten verwendet.

Aus der FFT-Size folgt die Anzahl der FFT-Bins[19]:

$$\text{Anzahl FFT-Bins} = \text{FFT-Size} / 2$$

Die Frequenz-Auflösung eines Bins beträgt[20]:

$$\text{Frequenz-Auflösung} = \text{Sampling-Rate} / \text{FFT-Size}$$

[18] Vgl. Github: C# wrapper for FFTW – https://github.com/tszalay/FFTWSharp, letzter Abruf: 23.06.2020

[19] Vgl. Ircam: AudioSculpt 3.0 User Manual „Introduction - FFT Size“ – http://support.ircam.fr/docs/AudioSculpt/3.0/co/FFT%20Size.html, letzter Abruf: 29.06.2020

[20] *Anmerkung: In der Literatur findet sich auch oft die äquivalente Formel: „Höchste darstellbare Frequenz / Anzahl der FFT Bins“. Der Quotient ist identisch, da Divident und Divisor bei dieser Definition jeweils halbiert sind.*

Beispielhaft beträgt bei 44.100 kHz Sampling-Rate und einer FFT-Size von 1024 die Frequenz-Auflösung eines Bins etwa 43 Hz – bei einer Anzahl von 512 Bins.

Die Audio-Daten werden nun in einzelne Teile entsprechend der FFT-Size aufgeteilt mit 50 % Überlappung. Der letzte Teil wird – wenn er nicht der FFT-Size entspricht – mit Nullen aufgefüllt.

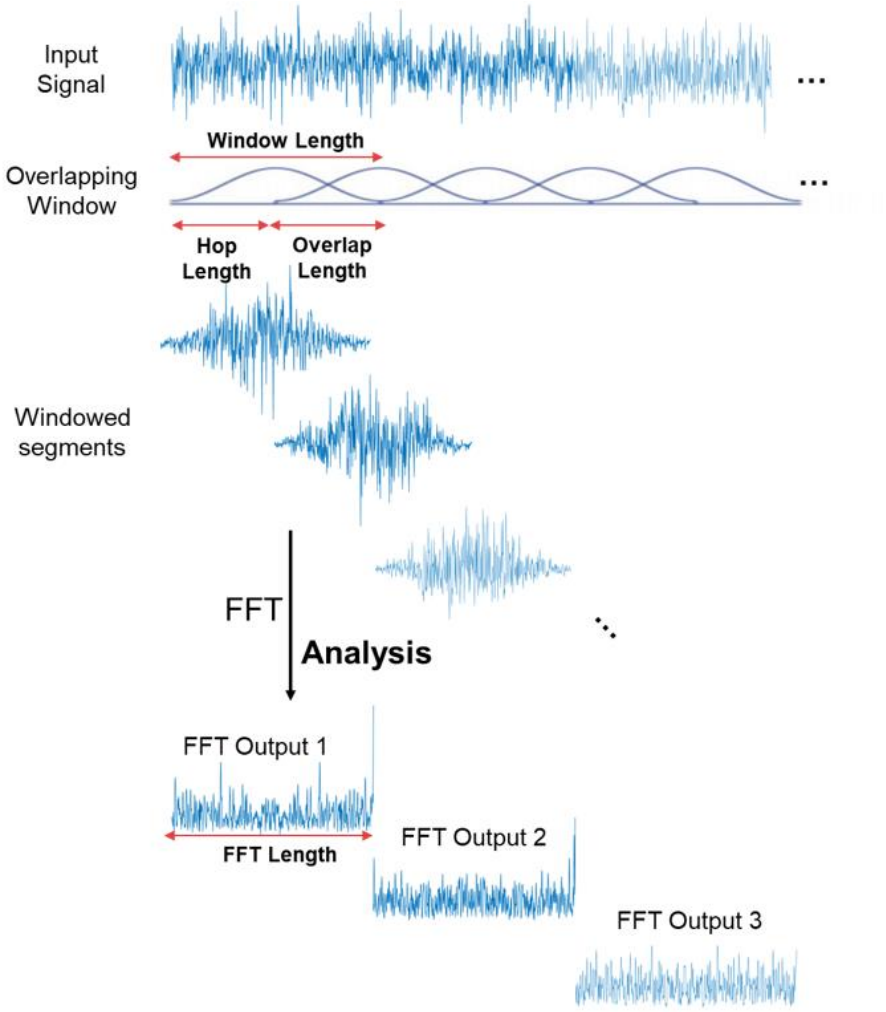


Abbildung 4 – Von-Hann Windowing mit 50 % Overlap – „Short-time FFT – MATLAB“, © 1994-2020 The MathWorks, Inc. – https://www.mathworks.com/help/dsp/ref/stft_output.png, letzter Abruf: 29.06.2020

<Spectral Leakage, Windowing Erklärung>

- Implementation, Windowing, Phase Amplitude, etc.
- Codebeispiele

4.2.2 Visualisierung der Spektrum-Daten

- Vertices, Triangles, Mesh
- Codebeispiele

- Unity Job System, Multithreading

4.2.3 User Interface und Steuerung

- Motion-Controller, Actions
- Konzept: Interaktion ohne verschachtelte 2D-Menüs, Werkzeugfarben, Ablesen im virtuellen Raum

4.2.4 Hardwareanforderungen und Performanz

Als Entwicklungssystem wurde ein Windows-PC mit Intel i9-9900K, 32 Gigabyte DDR4-3200 RAM und einer GeForce RTX 2070 mit 8 Gigabyte VRAM genutzt. Auf diesem System ist die Darstellung bei Audio-Dateien mit einer Länge unter einer Minute bei Sampling-Raten bis 96 kHz und 24 Bit flüssig möglich. Der Speicherbedarf ist hoch: VrAudioSandbox benötigt nach dem Laden einer x Sekunden langen Audio-Datei mit einer Samplingrate von … und … bit … etwa 18 Gigabyte Arbeitsspeicher.

5 Kritische Betrachtung der Ergebnisse

- Verbesserungsvorschläge
- Ideen für weitere Werkzeuge / Funktionen

6 Fazit

7 Literatur

Frigo, Matteo; Johnson, Steven G.: “The Design and Implementation of FFTW3” – Proceedings of the IEEE, Volume 93, Number 2, 2005, S. 216-231 – http://www.fftw.org/fftw-paper-ieee.pdf, letzter Abruf: 13.06.2020

Jerald, Jason (Ph.d.): „The VR Book: Human-Centered Design for Virtual Reality”, A publication in the ACM Book series #8, Association for Computing Machinery and Morgan & Claypool Publishers, San Rafael (California) (2016), ISBN 9781970001129

Norman, Donald A.: „The Design of Everyday Things”, Basic Books – Hachette Book Group company, New York (2002) (erstveröffentlicht als: „The Psychology of Everyday Things” [1988]), ISBN 9780465067107

Sutherland, Ivan: „A head-mounted three dimensional display” – AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I, December 1968 Pages 757–764, S. 757 – http://www.medien.ifi.lmu.de/lehre/ss09/ar/p757-sutherland.pdf, letzter Abruf: 10.06.2020

Sutherland, Ivan: „The Ultimate Display” – Konferenzband: Information Processing 1965: proceedings of IFIP Congress / Wayne A. Kalenich [Hrsg.]. International Federation for Information Processing, Amsterdam u.a., Washington u.a. (1965)

8 Sonstige Quellen

FFTW: „FFT Benchmark Results“ – http://www.fftw.org/speed/, letzter Abruf: 13.06.2020

FFTW: 3.3.8 Manual, Kapitel 4.8.1 „The 1d Discrete Fourier Transform (DFT)” – http://www.fftw.org/fftw3_doc/The-1d-Discrete-Fourier-Transform-_0028DFT_0029.html, letzter Abruf: 13.06.2020

Github: C# wrapper for FFTW – https://github.com/tszalay/FFTWSharp, letzter Abruf: 23.06.2020

Github: NAudio repository – https://github.com/naudio/NAudio, letzter Abruf: 23.06.2020

Ircam: AudioSculpt 3.0 User Manual „Introduction - FFT Size“ – http://support.ircam.fr/docs/AudioSculpt/3.0/co/FFT%20Size.html, letzter Abruf: 29.06.2020

Unity Technologies: „Multiplatform | Unity“ – https://unity.com/features/multiplatform, letzter Abruf: 24.06.2020

Unity Technologies: „Powerful 2D, 3D, VR, & AR software for cross-platform development of games and mobile apps.” – https://store.unity.com/#plans-individual, letzter Abruf: 11.06.2020

Unity Technologies: „Sie fragen sich was Unity ist? Entdecken Sie, wer wir sind, wo wir angefangen haben und wohin wir uns entwickeln | Unity“ – https://unity.com/de/our-company, letzter Abruf: 11.06.2020

9 Abbildungsverzeichnis

Abbildung 1 – Vergleich HUD-basierte Darstellung (oben) von Gesundheit und Munition mit im virtuellen Raum als physische Objekte manifestierter Darstellung (unten) 5

Abbildung 2 – Cone of Focus, UploadVR Copyright 2019 UVR Media LLC – https://mk0uploadvrcom4bcwhj.kinstacdn.com/wp-content/uploads/2016/07/cone-of-focus.jpg, letzter Abruf 17.06.2020 6

Abbildung 3 – Eine abstrakte Darstellung der Architektur - Eigene Darstellung 8

Abbildung 4 – Von-Hann Windowing mit 50 % Overlap – „Short-time FFT – MATLAB“, © 1994-2020 The MathWorks, Inc. – https://www.mathworks.com/help/dsp/ref/stft_output.png, letzter Abruf: 29.06.2020 12

10 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 nicht einverstanden, 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.

Oberboihingen, der xx.07.2020	Manuel-Philippe Hergenröder	
Ort, Datum	Vorname Nachname	Unterschrift

11 Anhang – Programmcode

Code Formatting Test #1

http://www.planetb.ca/syntax-highlight-word

```
1.  [BurstCompile]
2.  public struct FindPointsToUpdateJob : IJobParallelFor
3.  {
4.      [ReadOnly] public int meshIdx;
5.      [DeallocateOnJobCompletion] [ReadOnly] public NativeArray<Vector3> vertices;
6.      [DeallocateOnJobCompletion] [ReadOnly] public NativeArray<Vector3> points;
7.      [ReadOnly] public Vector3 direction;
8.      [ReadOnly] public float deformFactor;
9.      [ReadOnly] public float absoluteValue;
10.     [ReadOnly] public float radius;
11.     public NativeQueue<VertexChange>.ParallelWriter vertexChanges;
12.     public void Execute(int i)
13.     {
14.         for (int p = 0; p < points.Length; p++)
15.         {
16.             // Ignore y-axis when determing points to be affected
17.             var distance = (new Vector3(points[p].x, 0, points[p].z) - new Vector3(v
    ertices[i].x, 0, vertices[i].z)).magnitude;
18.             if (distance < radius)
19.             {
20.                 if (absoluteValue == -1)
21.                     // use factor for changing of y-value
22.                     vertexChanges.Enqueue(new VertexChange(
23.                         meshIdx,
24.                         i,
25.                         vertices[i].x + direction.x * deformFactor,
26.                         (vertices[i].y + direction.y * deformFactor >= 0) ? vertices
    [i].y + direction.y * deformFactor : 0f,
27.                         vertices[i].z + direction.z * deformFactor));
28.                 else
29.                     // use absolute value
30.                     vertexChanges.Enqueue(new VertexChange(meshIdx, i, vertices[i].x
    , absoluteValue, vertices[i].z));
31.             }
32.         }
33.     }
34. }
```