UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Virtueller Dirigent

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Marcel Bock

Erstgutachter: Prof. Dr.-Ing. Stefan Müller
Institut für Computervisualistik, AG Computergraphik

Zweitgutachter: M.Sc. Nils Lichtenberg
Institut für Computervisualistik, AG Computergraphik

Koblenz, im Januar 2016

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .

(Ort, Datum) (Unterschrift)

1 Zusammenfassung

Das Thema der vorliegenden Bachelorarbeit ist die Entwicklung einer Virtual-Reality-Anwendung, bei der der Benutzer in die Rolle eines Dirigenten versetzt wird und mithilfe von Handgesten das Orchester steuert. Als Hardware für die Gestenerkennung wurde die Microsoft Kinect verwendet und als Grundlage für die Entwicklung die Unreal Engine genutzt.

Ergebnis ist ein Prototyp, der einen Benutzer ein Musikstück mithilfe von Handgesten, in einem virtuellen Konzertsaal, dirigieren lässt. Es können einzelne Instrumente ausgewählt, die Lautstärke eingestellt und sich in der Szene umgesehen werden.
Mithilfe einer Evaluation des entwickelten Prototyps wurden Anforderungen festgehalten, die von Virtual-Reality-Anwendungen erfüllt werden sollten, um die Grundlage für ein immersives Erlebnis zu legen.

2 Abstract

The topic of the present thesis is the development of a virtual reality application which allows a user to become the conductor of a classical orchestra. With the aid of hand gestures the orchestra can be conducted. The Microsoft Kinect was used for gesture recognition and the Unreal Engine as basis for development.

As a result a prototype was developed which allows a user to conduct a composition with the aid of hand gestures in a virtual concert hall. It is possible to select single instruments, regulate the volume and to look around in the scene.
By means of an evaluation of the developed prototype, requirements were documented which should be used as a foundation for virtual reality applications to make an immersive experience possible.

Inhaltsverzeichnis

1 Zusammenfassung **ii**

2 Abstract **ii**

3 Einleitung **1**

4 Virtuelle Realität **2**
4.1 Anwendungen - Allgemein 2
4.1.1 Ozeanarium 2
4.1.2 Dunhuang Höhlen 3
4.1.3 Dom von Siena 3
4.1.4 Virtual Graffiti 4
4.2 Anwendungen - Musik 4
4.2.1 British Music Experience 4
4.2.2 Caruso - Ein Opernsänger 5
4.2.3 Conductor's Philosophy 6
4.2.4 Synse 6
4.3 Fazit 7

5 Konzept **8**
5.1 Grundlegende Idee 8
5.2 Klassenübersicht 9
5.3 Komponenten 11

6 Umsetzung **13**
6.1 Die Kinect 13
6.2 Die Unreal Engine 14
6.3 Notwändige Software 15
6.4 Das Hauptmenü 16
6.5 Die Spielumgebung 19
6.5.1 Der Konzertsaal 19
6.5.2 Die GUI 21
6.6 Der Sound 23
6.6.1 Die Sound-Dateien 23
6.6.2 Die Sound-Klassen 26
6.7 Der Highscore 27
6.8 Die Spieler-Klasse 28
6.9 Die Maussteuerung als Handbewegung 29
6.10 Die erste Geste: Orchester starten 32
6.11 Die zweite Geste: Lautstärke regeln 34
6.12 Instrumente dirigieren 37

7 Evaluation **41**
7.1 Durchführung 41
7.1.1 Der Fragebogen 41
7.1.2 Der Evaluationstest 42
7.2 Ergebnisse 43
7.2.1 Beschreibung 43
7.2.2 Analyse 51
7.2.3 Fazit 54

8 Fazit und Aussicht **55**
8.1 Ausgangslage 55
8.2 Fazit 55
8.3 Ausblick 56

Literatur **58**

Anhang **59**

3 Einleitung

Durch die schnelle Entwicklung der Hardware in den letzten Jahren sind die Möglichkeiten der 3D-Interaktion vielfältig geworden. Durch neue Anwendungen können Benutzer in eine Rolle versetzt werden, welche im normalen Leben nicht zugänglich wäre. Eine der Herausforderungen ist es, die Immersion für den Benutzer gut umzusetzen und den Benutzer in die Anwendung zu integrieren.
Ziel dieser Arbeit ist die Entwicklung einer interaktiven Anwendung, in der die Arbeit eines Dirigenten simuliert werden soll. Der Benutzer soll ein klassisches Orchester leiten, indem die Lautstärke dynamisch angepasst werden kann. Schwerpunkt der Arbeit ist die Gesternerkennung und Gestensteuerung, mit deren Hilfe die Lautstärke angepasst wird.
Hierfür soll entsprechende Hardware, wie zum Beispiel die *Microsoft Kinect*, verwendet und mit einer *Game-Engine* verbunden werden. Die graphische Repräsentation (Graphik und Animation) soll keinen Schwerpunkt darstellen. Optional sind weitere Gesten angedacht, welche dem Benutzer erlauben einzelne Instrumente auszuwählen und deren Einsätze zu steuern. Eine Verbindung mit der *Oculus Rift* gilt ebenfalls als optionale Erweiterung der Anwendung. Wünschenswert wäre die Möglichkeit dem Benutzer musikalisches Wissen, wie die Notenschrift, zu vermitteln und damit den Ansatz einer Lernsoftware in die Anwendung zu integrieren.

Die Ausarbeitung beginnt mit einer Einleitung in das Thema „Virtual Reality" und stellt unterschiedliche Anwendungen im Kapitel 4 vor. Danach wird im Kapitel 5 das Spielkonzept erläutert und erste Klassen und Komponenten vorgestellt. Im Kapitel 6 wird das implementierte Spiel und die verwendeten und erstellten Klassen beschrieben. Danach folgt im Kapitel 7 die Zusammenfassung der Evaluation der Anwendung. Zum Schluss, im Kapitel 8 , werden eine Zusammenfassung der Arbeit und ein Ausblick auf weitere Entwicklungsmöglichkeiten gegeben.

4 Virtuelle Realität

Die vorliegende Arbeit beschäftigt sich im Wesentlichen mit dem Thema „Virtuelle Realität". Dieser Begriff beschreibt Simulationsanwendungen, die es einem Benutzer ermöglichen virtuell eine Situation zu erleben. Diese Simulationen können sich beispielsweise mit Medizin, Automobil-Design oder Geschichte beschäftigen und haben in bestimmten Fällen, neben dem Faktor Unterhaltung, einen didaktischen Anspruch.
Seit den 60er-Jahren werden Geräte und Anwendungen entwickelt, die es einem Benutzer ermöglichen sollen in virtuelle Realitäten einzutauchen. Neben sogenannten „Head-Mounted-Displays", zu welchen auch die *Oculus Rift* gehört, wurden auch spezielle Peripherie-Geräte entwickelt, wie zum Beispiel Datenhandschuhe, damit die *Immersion* für den Benutzer maximal erlebbar ist.
Vor allem in der Industrie wurden die neuen Geräte benutzt, um beispielsweise Auto-Designs im Vorfeld begutachten zu können. Auch Operationen lassen sich in einer virtuellen Umgebung durchführen und Architektur-Entwürfe sind virtuell begehbar.
Die Vorteile der virtuellen Realität in der Industrie liegen hierbei in der Möglichkeit Produkte betrachten und in einem gewissen Rahmen testen zu können, um dadurch Kosten zu sparen, indem Fehler frühzeitig erkannt und behoben werden können. Für die Unterhaltungsindustrie liegt der Vorteil in dem Gefühl für die Benutzer nicht nur Betrachter einer Welt, sondern ein Teil von ihr zu sein. Darüber hinaus können entsprechende Anwendungen genutzt werden um Wissen auf neuartige Art und Weise zu vermitteln.

4.1 Anwendungen - Allgemein

Dieser Abschnitt widmet sich allgemeinen Anwendungen aus dem Bereich „Virtual Reality", um einen ersten Überblick zu schaffen und Anwendungsbereiche vorzustellen. Das Hauptaugenmerk liegt hierbei auf Anwendungen, die Wissen mithilfe neuer Technologien vermitteln wollen.

4.1.1 Ozeanarium

Das Fraunhofer Institut für graphische Datenverarbeitung (IGD) hat auf der EXPO des Jahres 1999 ein virtuelles Ozeanarium ausgestellt. Der Besucher sollte die Möglichkeit erhalten eine realitätsnahe Unterwasserwelt zu erkunden und dabei Meeresbewohner wie Fische, Haie und Rochen zu beobachten und die Pflanzenwelt zu entdecken.
Insgesamt vier Teilbereiche umfasste das Ozeanarium und simulierte die Biotope des Atlantiks, der Antarktis, des indischen Ozeans und des Pazifiks. Diese Biotope konnten von außen beobachtet werden, während ein sogenannter „Haupttank" dazu diente einen virtuellen Tauchgang durchzuführen. Hierbei konnte der Benutzer

in den Tank virtuell eintauchen und die Tiere und Pflanzen unter Wasser erleben.
Um eine möglichst hohe *Immersion* des Benutzers zu erreichen, beginnt das Programm mit einem Helikopter-Flug über das virtuelle Ozeanarium. Erst dann kann der Benutzer sich in dem Gebäude bewegen und die Biotope frei erkunden, wie in einem echten Ozeanarium. Für den Tauchgang wurden unter anderem spezielle Tauchgeräusche eingespielt beispielsweise das Atmen mit einer Gasflasche. Mit den Tieren der Unterwasserwelt konnte der Benutzer interagieren und Haie füttern oder Delphine reiten. Die virtuellen Fische verhielten sich nach bestimmten Verhaltensmustern, die in Echtzeit simuliert wurden. Hier haben sich die Entwickler an die Verhaltensforschung von Fischen gehalten und versucht die Instinkte und die Wahrnehmung der Tiere zu simulieren, um ein typisches Verhalten zu erreichen und zeigen zu können.
(Quelle: [Fro02])

4.1.2 Dunhuang Höhlen

Auch dieses Projekt wurde vom Fraunhofer-Institut für graphische Datenverarbeitung (Darmstadt/Rostock) geleitet und war ein Gemeinschaftsprojekt mit der Universität Hangzhou (China). Das Ziel war es ein detailliertes Modell der Dunhuang Grotten zu erstellen, welche ein altes Höhlentempel-System darstellen. Eine dieser Höhlen wurde in ein 3D-Modell überführt und mithilfe eines „CAVE" (*Cave Automatic Virtual Environment*) präsentiert. Ein „CAVE" ist ein begehbarer Raum, welcher Projektionsflächen verwendet, die eine virtuelle Umgebung simulieren.
Der Benutzer bewegt sich frei innerhalb der Höhle und hat eine Taschenlampe als Interaktionsschnittstelle. Die Kamera beleuchtet Teile der Höhle, Objekte wie Statuen oder Höhlenmalereien. Sobald ein Objekt längere Zeit beleuchtet wird, werden dem Benutzer Informationen eingeblendet.
Das Ziel war es einem Besucher der virtuellen Höhlen Wissen über diese zu vermitteln und dabei kulturelles Erbe, wie es die Dunhuang-Höhlen darstellen, nicht zu gefährden und zu bewahren.
(Quelle: [Lut09])

4.1.3 Dom von Siena

In den Jahren 1999 und 2000 ist der virtuelle „Dom von Siena", vom Fraunhofer-Institut für graphische Datenverarbeitung IGD, entwickelt worden. Unter anderem war es das Ziel eine Anwendung zu entwickeln, die immersiv Wissen vermittelt und Geschichte interaktiv erlebbar macht. Zur Erreichung dieses Zieles wurde der Dom von Siena als Kulturobjekt ausgewählt.
Der Dom wurde als 3D-Modell nachgebaut inklusive dem Vorplatz, den Kapellen und einer Bibliothek. Insgesamt 10 Stationen innerhalb der Kirche können angesteuert werden und ein virtueller Fremdenführer führt den Benutzer durch das Areal und vermittelt Informationen über den Dom.

Interagieren kann der Benutzer über einen Touchscreen. Dieser wurde als mittelalterlicher Foliant konzipiert und bietet die Möglichkeit mit dem virtuellem Fremdenführer zu sprechen und in der Kirche zu navigieren.
Die *Immersion* wird unter anderem durch realistische Körperbewegungen des Fremdenführers und eine sprachsynchrone Mimik unterstützt. Auf dem Domplatz selber steht zusätzlich ein Ingenieur, der Tauben füttert oder diese vertreibt, um eine lebendige Welt zu simulieren.
(Quelle: [BFK+09])

4.1.4 Virtual Graffiti

Im Rahmen einer Studien-Arbeit hat Moritz Gerl, ehemaliger Student der Universität Koblenz, im Jahr 2004 eine virtuelle Sprühwand entwickelt. Das Ziel war es einem Benutzer eine realitätsnahe Erfahrung von Graffiti sprühen zu bieten.
Hierfür wurde eine echte Sprühdose verwendet, welche mithilfe einer *Funkmaus* umgebaut wurde und somit ein digitales Signal für die Simulation des Sprühens an den Computer senden konnte. Darüber hinaus wurde auch ein entsprechender Sprühsound eingebaut.
Auf einer Leinwand konnten die Benutzer die Dose anwenden, unterschiedliche Farben und Dichten des Sprüh-Materials auswählen.
Auf diesem Wege erhielten die Benutzer die Möglichkeit Graffiti zu sprühen ohne eine Ordnungswidrigkeit zu begehen.
(Quelle: [Ger04])

4.2 Anwendungen - Musik

Im folgenden Abschnitt wird sich Anwendungen gewidmet, die speziell für die Vermittlung von musikalischem Wissen und das digitale Erleben von Musik entwickelt wurden. Bei den folgenden Anwendungen lag der Schwerpunkt auf dem spielerischen Umgang mit Musik, um Musik auch für Nicht-Musiker zugänglich zu machen und eine neuartige Erfahrung zu ermöglichen.

4.2.1 British Music Experience

Von 2009 bis 2014 fand die „British Music Experience" Ausstellung in Greenwich, London, statt. Die Ausstellung stellte eine musikalische Zeitreise von 1944 bis zur Gegenwart dar und leitete den Besucher durch mehrere Attraktionen, die die Musikgeschichte thematisierten. Viele der Attraktionen legten ihren Schwerpunkt darauf den Benutzer die Musik auf neue Arte und Weise betrachten zu erfahren.

Gibson Interactive Studio:
Das „Gibson Interavtive Studio" ermöglichte es den Besuchern der Ausstellung

Gitarre, E-Piano oder Schlagzeug zu erlernen. Hierbei wählten die Besucher einen Schwierigkeitsgrad und konnten interaktiv mit einem Musiklehrer ein gewähltes Stück üben. Der Musiklehrer agierte über ein Video mit dem Benutzer und zum Abschluss konnte der Benutzer mit der originalen Band des gewählten Stückes zusammen spielen.

Dance the Decades:
Mithilfe von „Dance the Decades" sollte Besuchern das Tanzen beigebracht werden. Hierfür wurde eine musikalische Epoche, zum Beispiel die Disko-Zeit, ausgewählt und eine virtuelle Vortänzerin zeigte die wichtigsten Tanzschritte. Um den eigenen Lernerfolg festzuhalten und zur Unterhaltung der Besucher wurde, nach erlernen des Tanzes, ein Video der Besucher aufgenommen.

The Beat Goes On:
Eine der letzten Attraktionen der Ausstellung versetzte die Besucher in ein Live-Konzert von unterschiedlichen Bands. Hierbei wurden hochauflösende Leinwände benutzt, die einen dreiseitigen Würfel bildeten. Mithilfe von Raucheffekten und speziellen Soundeinrichtungen wurden die Besucher in das Konzert versetzt und konnten von der Bühne aus das Konzert erleben. Unterstützt wurde die Immersion durch eine spezielle *3D-Layer-Technik*, so dass ein Tiefeneindruck für den Betrachter entstand.

(Quelle: [WC09])

4.2.2 Caruso - Ein Opernsänger

Jochen Feitsch und Marco Strobel, Masterstudenten der Fachhochschule Düsseldorf, haben unter Leitung von Prof. Dr. Christian Geiger einen virtuellen Tenor entwickelt, der 2013 auf der MuC-Messe („Mensch und Computer") ausgestellt wurde.
Ein Benutzer der Anwendung wurde in die Lage versetzt das Stück „Ave Maria" zu singen, ohne selber singen zu müssen. Mithilfe von Handgesten und Mundbewegungen konnte die Stimme eines Tenors gesteuert werden. Das Ziel war es dem Original so nah wie möglich zu kommen. Um ein immersives Erlebnis zu schaffen hatten die Benutzer eine spezielle Weste an, die die Vibrationen beim Singen simulieren sollte. Als Rückmeldung, wie gut der Benutzer sang, diente ein 3D-Avatar. Dieser veränderte sein Aussehen vom Benutzer hin zu einem echten Opernsänger, je nachdem wie gut der Gesang gesteuert wurde.
Auch die eigene Kreativität des Benutzers wurde gefordert, da unter anderem die Lautstärke frei gesteuert werden konnte und keinen Einfluss auf die Rückmeldung des Gesanges hatte. (Quelle: [PFSG14])

4.2.3 Conductor's Philosophy

Dieses experimentelle Projekt wurde ebenfalls von der TH Düsseldorf durchgeführt und beschäftigte sich mit dem Dirigieren eines virtuellen Pianisten. Ein realer Pianist und ein virtuelle Pianist wurden von einem Benutzer gesteuert, indem dieser unter anderem den Takt und die Lautstärke vorgab. Der Benutzer übernahm damit die Rolle eines Dirigenten.
Der virtuelle Pianist konnte mithilfe der Gesten des Dirigenten, welche über eine Kamera aufgenommen wurden, die Anweisungen des Dirigenten auf das Klavier übertragen. Es handelte sich hierbei um ein digitales Klavier, welches die Tasten selbstständig anschlagen konnte.
Darüber hinaus wurden die gespielten Töne auf einer Leinwand farblich dargestellt. Für jede angespielte Note entstand ein Punkt auf der Leinwand und Akkorde wurden mit Strichen miteinander verbunden. Damit wurde die Musik zusätzlich mit einem künstlerischen Aspekt verbunden.
(Quelle: [Gei13])

4.2.4 Synse

„Synse" ist eine Android-App, welche am 20. Oktober 2011 veröffentlicht wurde und von Christian Sander und Michael Becker entwickelt wurde. Es handelt sich um eine Musik-App, die es ermöglicht eigene, aufgenommene Videos mit Musik und speziellen Effekten zu unterlegen.
Der Benutzer nimmt ein beliebiges Video auf und kann mithilfe des entwickelten Editors verschiedene Instrumente und Melodien einbinden. Die zugrunde liegende Musik verändert daraufhin das Video, so wie es der Benutzer sich wünscht. Beispielsweise können Laternen einer Straße im Takt der Musik aufleuchten oder Pixel-Figuren durch das Video laufen.
Auf diese Weise können eigene Songs kreiert und visualisiert werden.
(Quelle: [ton11])

4.3 Fazit

Die vorgestellten Anwendungen zeigen, welche Möglichkeiten mit „Virtual Reality" bereits vorhanden sind. Sowohl die neuartige Benutzerinteraktion, wie beim virtuellen Graffiti beschrieben (siehe Abschnitt 4.1.4), als auch die Wissensvermittlung, wie sie beispielsweise bei den Dunhuang Höhlen umgesetzt wurde (siehe Abschnitt 4.1.2), sind komplexe Themenfelder. Anhand der Beispiel-Anwendungen von *Caruso* (siehe Abschnitt 4.2.2) und der *Conductor's Philosophy* (siehe Abschnitt 4.2.3) ist erkennbar, dass die neuen Techniken auch in Verbindung mit Musik funktionieren können.
Durch die neuen Techniken haben Entwickler die Möglichkeit die Benutzer mehr in die Anwendungen zu integrieren und neuartige Erfahrungen zu vermitteln.

5 Konzept

Auf Grundlage der im zweiten Kapitel vorgestellten Anwendungen (siehe Kapitel 4) sollte nun ein eigenes Konzept für eine Anwendung erstellt werden, die die Idee der *virtuellen Realität* aufgreift. Inspiriert wurde das grundlegende Konzept hierbei von den Anwendungen „Caruso - Ein Opernsänger" und „Conductor's Philosophy" und sollte sich dementsprechend mit Musik beschäftigen.

5.1 Grundlegende Idee

Die Anwendung sollte sich mit Musik beschäftigen und dabei den Benutzer aktiv einbinden. Neben den Möglichkeiten den Benutzer aktiv ein bestimmtes Instrument spielen zu lassen, wie ein Klavier oder eine Geige, wurde sich dazu entschieden den Benutzer in eine anleitende Rolle zu versetzen, wie einen Musiklehrer.
Aus der ersten Idee, ein einzelnes Instrument zu dirigieren, wurde ein Konzept ausgearbeitet, den Benutzer zum Dirigenten eines ganzen Orchesters zu machen.

Als Dirigent sollte der Benutzer vor einem virtuellen Orchester stehen. Hierbei stand vor allem die Authentizität im Vordergrund und das Orchester sollte in einem in 3D nachgebauten Konzertsaal stehen und mit einzelnen Musikern besetzt sein. Die Besetzung sollte sich nach einem klassischen Orchester richten.
Die graphische Benutzeroberfläche sollte nur aus einer Lautstärke-Anzeige bestehen und einer Anzeige für die Noten des aktuell gespielten Stückes. Der Spieler sollte mithilfe von Handgesten einzelne Instrumente oder Instrumentengruppen auswählen und ihre jeweilige Lautstärke senken oder anheben können.

Jede Instrumentengruppe sollte ihre eigenen Sound-Dateien haben, welche mithilfe von *Surround-Sound* einen realistischen Klang erzeugen sollten. Die klassischen Stücke würden mithilfe von Musikprogrammen wie *MAGIX Music Maker* [1] erstellt und abgemischt werden.

Für die Gestenerkennung standen die *Microsoft Kinect* [2], die *Leap Motion* [3] oder ein *Datenhandschuh* [4] zur Verfügung. Um die *Immersion* zu verbessern stand auch die Möglichkeit offen die *Oculus Rift* [5] zu benutzen. Dies sollte den Benutzer besser in die Szene integrieren.

Über dieses grundlegende Spielkonzept hinaus wurde überlegt dem Benutzer musikalisches Wissen zu vermitteln. Das Notenlesen und die Interpretation der Nota-

[1] Weiterführende Informationen: Magix Music Maker
[2] Weiterführende Informationen: Kinect
[3] Weiterführende Informationen: Leap Motion
[4] Weiterführende Informationen: Datenhandschuh - Erklärung
[5] Weiterführende Informationen: Oculus Rift

tion, wie zum Beispiel die Dynamik (*ppp* bis *fff*) boten sich an.
Als Anreiz dafür ein Musikstück öfter zu spielen wurde sich überlegt einen *Highscore* einzubauen. Für jedes gespielte Stück gäbe es eine Endwertung, die sich danach richtet, ob die gewählte Lautstärke der in der Partitur angegebenen Lautstärke entsprach.

5.2 Klassenübersicht

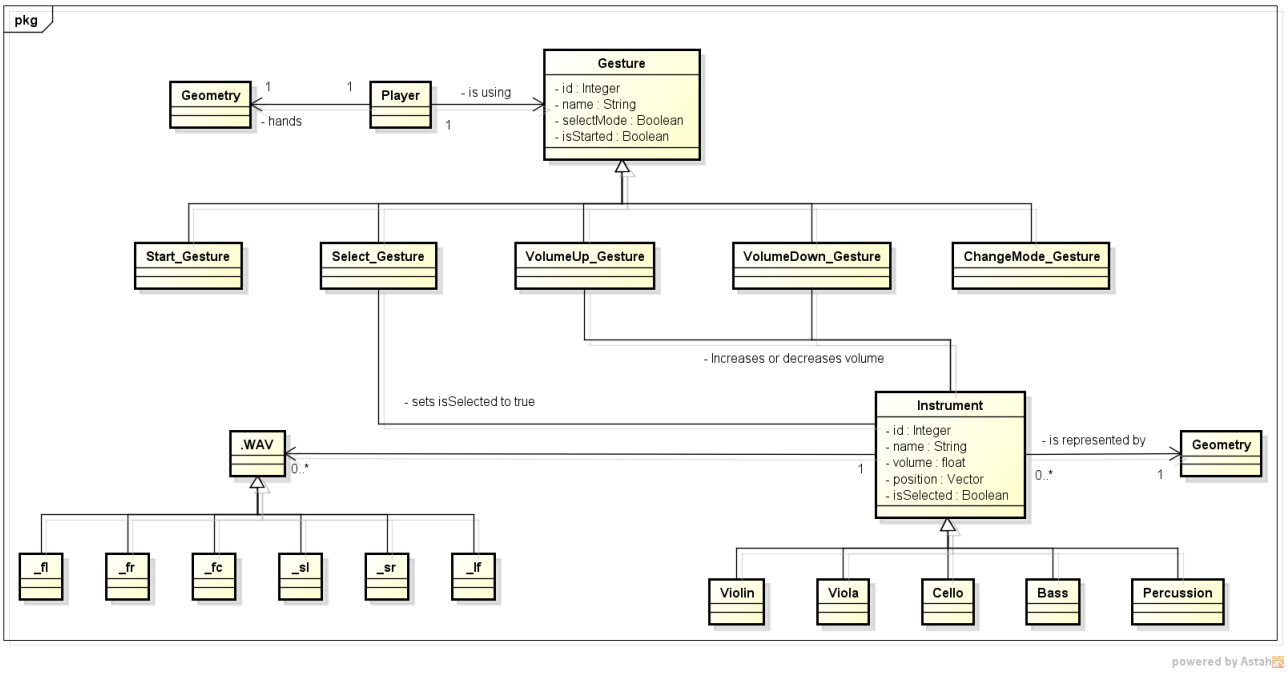

Abbildung 1: Konzept der Klassen und der Klassenzusammenhänge des späteren Spiels

Die Konzept-Idee aus Abschnitt 5.1 wurde um ein UML-Diagramm (siehe Abbildung 1) erweitert. Dieses UML-Diagramm sollte die Zusammenhänge im späteren Spiel vereinfacht darstellen und eine Übersicht über die beteiligten Komponenten liefern.

Der Spieler sollte durch eine abstrakte Geometrie dargestellt werden, vorzugsweise sollten Hände dazu dienen, den Spieler durch einen Avatar in der Spielwelt zu repräsentieren. Nur eine Ansicht aus Sicht der Spielfigur (*First-Person*) sollte bereitgestellt werden.
Auch die Instrumente des Orchesters sollten durch 3D-Objekte repräsentiert werden, jedoch ohne begleitende Musiker und ohne Animationen. Jedes Instrument sollte eine eindeutige *ID* zugeordnet bekommen, damit das jeweilige Instrument in der Spielwelt eindeutig definiert wäre. Darüber hinaus sollte jede Instrument-

Klasse eine Position in der Welt besitzen und sich neben der Lautstärke-Information ebenfalls speichern, ob die Instanz vom Spieler ausgewählt wurde oder nicht. Nur wenn der Spieler das Instrument ausgewählt hätte wäre es möglich, die Lautstärke zu verändern.
Damit auch Musik abgespielt werden würde sollte jeder Instrument-Klasse eine *.wav-Datei* zugeordnet werden, welche die Sound-Informationen bereitstellen würde. Die *.wav-Dateien* würden hierfür *Surround-Sound*-Informationen beinhalten und aus insgesamt sechs Kanälen bestehen.

Die Spieler-Klasse und damit der Spieler sollte Zugriff auf insgesamt fünf Gesten haben. Sobald das Spiel gestartet werden würde, wäre nur die *Start-Geste* benutzbar, damit der Spieler das Spiel starten könnte und die Musik erst losgehen würde, bevor weitere Gesten zur Verfügung ständen. Dafür hätten alle Gesten Zugriff auf die Information, ob die Musik schon gestartet wurde.
Nachdem der Spieler die Musik gestartet haben würde, befände er sich im *Selektier-Modus*. Jede Geste würde sich diese Information speichern, da im *Selektier-Modus* nur die *Select-Geste* verfügbar wäre. Mit Hilfe dieser Geste sollte der Spieler ein einzelnes Instrument auswählen können. Sobald dies geschehen wäre, würden die *Volume-Up*- und *Volume-Down*-Gesten verfügbar werden. Der Spieler sollte nun in der Lage sein die Lautstärke des ausgewählten Instruments zu verändern.
Mithilfe der *ChangeMode-Geste* sollte es möglich sein wieder zurück in den *Selektier-Modus* zu kommen, so dass das Instrument, das gerade ausgewählt war, nun nicht mehr ausgewählt wäre und die Lautstärke nicht mehr verändert werden könnte.

5.3 Komponenten

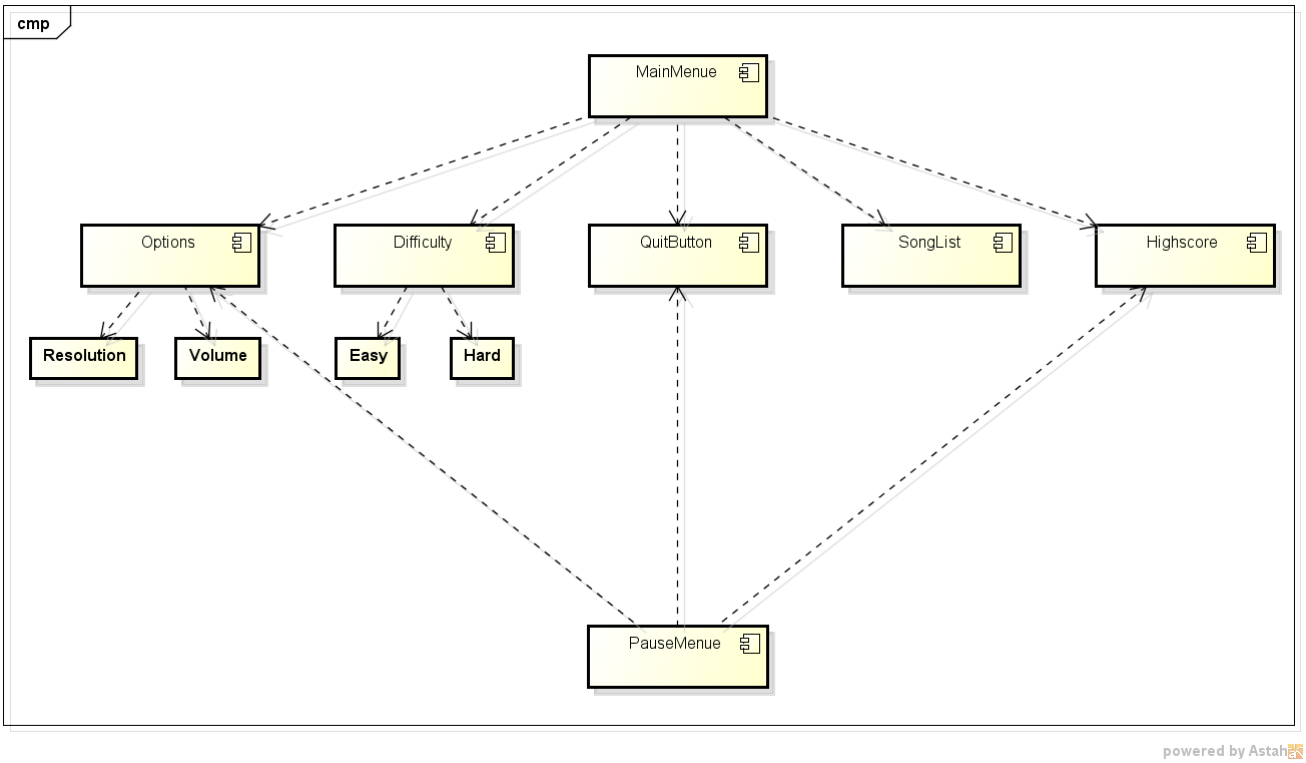

Abbildung 2: Konzept der Menü-Komponenten des späteren Spiels

Das Hauptmenü sollte dem Spieler vier interaktive Schaltflächen bieten, um folgende Untermenüs zu erreichen:

- Optionen
- Schwierigkeitsgrad
- Songauswahl
- Highscore

Darüber hinaus würde sich das Spiel mithilfe einer Schaltfläche *„Beenden"* wieder schließen lassen können (siehe Abbildung 2).

Um die Grafikqualität, Auflösung und Soundlautstärke an das eigene System anpassen zu können, sollte das Options-Menü dem Spieler bereitgestellt werden. Der Schwierigkeitsgrad würde sich in *„Leicht"*, *„Mittel"* und *„Schwer"* gliedern und gelte für alle Lieder, die im Menü *„Songauswahl"* ausgewählt werden könnten. Der in einem Spiel erreichte *Highscore* wäre im Hauptmenü abrufbar, damit eigene Erfolge mit denen anderer Spieler verglichen werden könnten.

Im Spiel selber würde sich das Hauptmenü auf die Optionen, den Zugriff auf den Highscore und die Möglichkeit das Spiel zu beenden reduzieren. Der Schwierigkeitsgrad sollte nicht während des Spiels verändert werden können.

6 Umsetzung

Das in Kapitel 5 vorgestellte Spiel-Konzept wurde innerhalb von ungefähr drei Monaten in einen lauffähigen Prototypen umgesetzt. Im folgenden Kapitel werden die umgesetzten Inhalte thematisiert und es wird erklärt, wie und mit welchen Hilfsmitteln das Spiel programmiert wurde.

6.1 Die Kinect

Zu Beginn der Bachelor-Arbeit standen drei mögliche Eingabegeräte zur Wahl: Die *Leap-Motion*, die *Wii-Fernbedienung* und die *Microsoft Kinect* [6]. Im Folgenden sollen alle drei Eingabegeräte kurz vorgestellt werden und es soll eine Begründung gegeben werden, warum sich am Ende für die *Microsoft Kinect* entschieden wurde.

Die *Leap-Motion* wurde vom Unternehmen „Motion Control" entwickelt und seit Mai 2014 ist die 2. Version der Software Entwicklern zur Verfügung gestellt worden. Bei der *Leap-Motion* handelt es sich um eine Hardware, welche die Hände eines Benutzers aufnimmt und die erhaltenen Informationen in Eingaben übersetzt, so dass eine Interaktion mit Programmen möglich wird. Unter anderem wird dieses Eingabegerät genutzt, um auf Webseiten zu navigieren, Gesten zur Veränderung der Größe auf Karten-Anwendungen zu ermöglichen oder mithilfe der Hand-Erkennung Zeichen-Programme zu bedienen.

Seit dem 2. Dezember 2006 ist die Nintendo-Konsole *Wii* auf dem Markt erhältlich. Mit der neuen Konsole von Nintendo wurde auch eine neuartige Steuerungsmöglichkeit vorgestellt, die *Wii-Remote*. Die Controller der *Wii* sind kabellos mit der Konsole verbunden und mithilfe eines Sensors (*Wii-Sensorleiste*) wird die Bewegung des Controllers verfolgt. So ist es möglich in den Menüs der grundlegenden Software zu navigieren, indem die Hand des Benutzers den virtuellen Mauszeiger steuert. Auch komplexere Gesten sind durch die Infrarot-Technik der *Wii-Remote* möglich und werden in Spielen eingesetzt.

Vom Unternehmen Microsoft wurde am 4. November 2010 das erste Mal die *Kinect* für die XBOX-360 veröffentlicht. Seit dem Jahr 2012 ist die *Kinect* auch mit Windows-Systemen kompatibel und im Jahre 2014 wurde der Sensor überarbeitet und liegt aktuell in der Version 2 vor.
Der Sensor benutzt Infrarot, um seine Umgebung zu erfassen und hat darüber hinaus eine Kamera in *Full-HD Auflösung* und ein eingebautes Mikrophon. Bis zu sechs Personen können gleichzeitig vom Sensor erfasst werden.

Am Ende wurde sich für die *Kinect* von Microsoft entschieden, weil der erfasste Bereich (*Field-of-View*) sehr groß ist und der ganze Körper des Benutzers erfasst

[6] Weitere Informationen: Wii-Remote, Kinect, Leap Motion

werden kann. Auch viele Videospiele, die für die Xbox-360 oder Xbox-One entwickelt wurden, unterstützen die Technik der Kinect, weshalb sich die Kinect ebenfalls anbot für ein eigenes Spiel, welches per Gesten gesteuert werden sollte.
Die *Leap-Motion* wurde sehr früh als Eingabegerät ausgeschlossen, da der wahrnehmbare Bereich im Gegensatz zur Kinect sehr klein ausfällt und nur die Hände des Benutzers erfasst werden anstatt der ganze Körper. Bei der *Wii-Remote* störte der Controller als solcher, da der Spieler des zu entwickelnden Spiels nur mit Gesten in dem Spiel interagieren und keine Eingabegeräte in den Händen halten sollte. Dies sollte dazu dienen die Immersion zu verbessern.

6.2 Die Unreal Engine

Um das Spiel als Bachelorarbeit zu realisieren standen drei Möglichkeiten zur Wahl:[7]

1. Unreal Engine
2. Unity Engine
3. selbst geschriebene Engine

Relativ früh wurde entschieden, dass keine selbst geschriebene *Engine* erstellt werden sollte, in der das Spiel entwickelt werden würde. Diese Entscheidung wurde aus zeitlichen Gründen getroffen und weil der Fokus der Bachelorarbeit auf der Entwicklung eines „Virtual Reality"-Spiels liegen sollte und nicht auf der Entwicklung einer Entwicklungsumgebung für ein solches Spiel.

Die *Unreal Engine* in der Version 4 wurde am 19. März 2014 vom Unternehmen „Epic Games" veröffentlicht. Seit 1998 ist die *Unreal Engine* für Entwickler verfügbar und seit 2009 auch, im Rahmen von nicht-kommerziellen Zwecken, kostenlos für die Öffentlichkeit.
Neben einer Grafik-Engine besteht die *Unreal Engine* aus einem Level-Editor und unterstützt Skriptsprachen und die Programmiersprache C++. Eine eingebaute Physik- und Sound-Engine ermöglichen physikalische Simulationen und das Einbinden von Sound-Dateien. Animationen können verwendet werden, visuelle Effekte und Netzwerkfunktionen sind ebenfalls integriert. Auch „Virtual Reality"-Hardware wie die *Oculus Rift* werden unterstützt.
Mithilfe von sogenannten *„Blueprints"*, welche Klassen in der *Unreal Engine* repräsentieren, ist es möglich visuell zu Programmieren (genannt *visual scripting*), auch ohne Kenntnisse in einer Programmier- oder Skriptsprache.
Unter anderem wurden Spiele wie *Mass-Effect*, *Borderlands* oder *Batman:Arkham Asylum* mithilfe dieser Engine entwickelt und konnten damit kommerziellen Erfolg

[7] Informationen zu den Engines: Unreal Engine, Unity Engine

erzielen. [8]

Neben der *Unreal Engine* stand auch die *Unity Engine* zur Wahl. Auch diese *Engine* steht kostenlos zur Verfügung, jedoch mit Einschränkungen im Umfang. Sie wurde vom Entwickler „Unity Technologies" im Jahre 2005 veröffentlicht und befindet sich aktuell in der Version 5. Die *Unity Engine* bietet ebenfalls eine Grafik-Engine, verfügt über ein Animationssystem und mithilfe von Skriptsprachen und der Programmiersprache C++ lassen sich eigene Inhalte schreiben. Zusätzlich wird unter anderem die *Oculus Rift* unterstützt.
Spiele wie *Angry Birds*, *Pillars of Eternity* und *Heartstone: Heroes of Warcraft* wurden mithilfe dieser *Engine* entwickelt und hatten kommerziellen Erfolg.[9]

Es wurde sich speziell für die *Unreal Engine* entschieden, da für die Kinect ein Plug-In angeboten wurde, mit welchem andere Entwickler im Internet bereits gute Erfahrungen gemacht hatten. Außerdem war es eine persönliche Vorliebe des Entwicklers, die eine Rolle gespielt hat, da dem Entwickler schon einige Projekte bekannt waren, die mithilfe der Engine entwickelt worden sind. [10]

6.3 Notwändige Software

Damit mit der Programmierung des Spiels begonnen werden konnte, wurde zunächst die benötigte Software installiert.
Für die *Kinect* wurde ein Desktop-PC mit Windows 8.1 benötigt, da auf älteren Systemen die Kinect nicht unterstützt wurde. Später wurde auch ein Desktop-PC mit Windows 10 genutzt, auf dem die Software ebenfalls ohne Probleme lief. Damit die Kinect vom System erkannt und benutzt werden konnte, wurde das *Kinect SDK 2.0* installiert. [11]
Die *Unreal Engine* in der Version 4.8.3 musste installiert werden und wurde später auf die Version 4.9.2 aktualisiert. Keine Version der Unreal Engine unterstützt die Kinect. Hierfür wird ein spezielles Plug-In der Firma „Opaque" benötigt: *Kinect4Unreal*[12].
Dieses Plug-In integriert die entsprechenden Funktionalitäten in die Unreal Engine.
Als Hilfsmittel stand ein Einführungsprojekt zur Verfügung, welches auf dem Unreal Engine Marktplatz kostenlos heruntergeladen werden konnte. In diesem werden verschiedene Räume präsentiert, in denen die Funktionen, die das Plug-In zur Verfügung stellt, vorgestellt werden. [13]

[8] Weitere Informationen zu den Spielen: Mass Effect, Borderlands, Batman: Arkham Asylum
[9] Weitere Informationen zu den Spielen: Angry Birds, Pillars of Eternity Heartstone
[10] Informationen zum Spiel Cubes
[11] Kinect SDK 2.0: Kinect SDK
[12] Kinect Plug-In für die Unreal Engine: Kinect4Unreal
[13] Einführungsprojekt: K4U Introduction

6.4 Das Hauptmenü

Das Hauptmenü wurde mithilfe eines „User-Interface-Blueprint" als „User Widget" umgesetzt [14]. Ein *Widget* ermöglicht es dem Entwickler eine graphische Oberfläche zu erstellen, die im Spiel zum Beispiel als *GUI* („graphical user interface") genutzt werden kann. Darüber hinaus kann der Entwickler die notwendige Logik für die graphische Oberfläche programmieren.
Im Hauptmenü des entwickelten Spiels wurde ein Hintergrundbild eingefügt, welches den Spieler auf das bevorstehende Spiel einstimmen sollte. Aus diesem Grund fiel die Wahl auf das Bild eines Konzertsaales (siehe Abbildung 3).

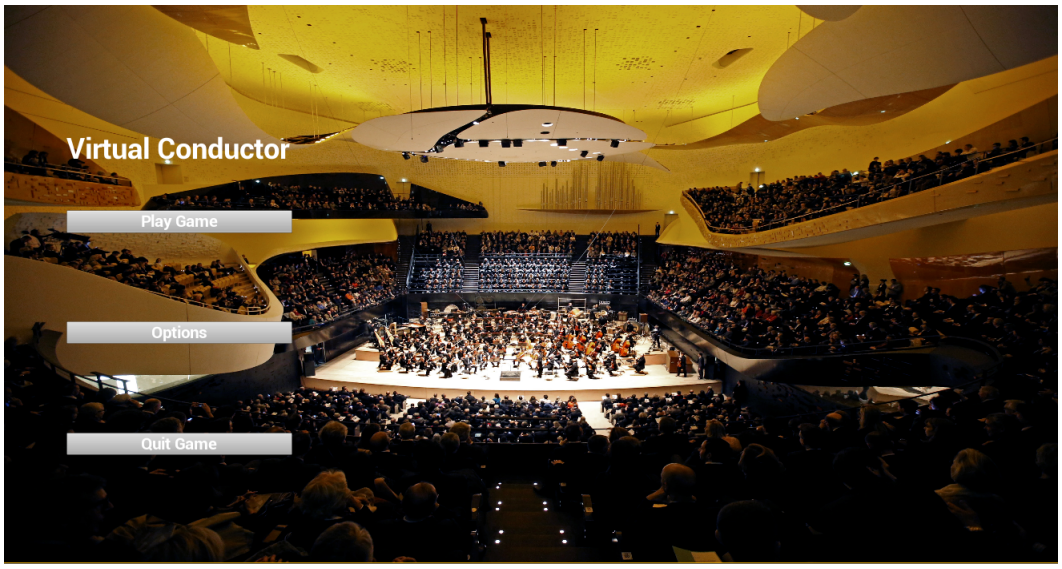

Abbildung 3: Das Layout des Hauptmenüs

Die Schaltflächen des Menüs wurden in den Farben Weiß und Grau gehalten und am linken Bildschirmrand platziert, damit das Hintergrundbild in den Vordergrund rücken würde.
Folgende drei Schaltflächen wurden umgesetzt:

- Spiel beginnen
- Optionen
- Spiel beenden

Zum Zeitpunkt der Erstellung des Spiels gab es nur ein Lied, das der Spieler spielen konnte, daher wurde auf eine *Songliste* verzichtet. Der *Schwierigkeitsgrad* entfiel, da mehr Fokus auf das virtuelle Erlebnis gelegt wurde, als auf die spielerische

[14] Unreal Engine Wiki [Gam15b]: Widget-Blueprint

Komponente. Aus diesem Grund gibt es auch keinen abgespeicherten *Highscore* und das Optionsmenü bietet ausschließlich Einstellungsmöglichkeiten für die Bildschirmauflösung.

Über den Blueprint lässt sich mithilfe der Schaltfläche *„Spiel beginnen"* das Level *„Game"* starten. Das Hauptmenü ist ein eigenständiger Level des Spiels. Während das Optionsmenü angezeigt wird, wird das Hauptmenü ausgeblendet.
Als Hintergrundmusik dient eine Dauerschleife, die die Geräusche vor einem Konzert abspielt (Husten, Gespräche, Stimmung der Instrumente). Die entsprechende Sound-Datei wurde von *„Youtube"* heruntergeladen [15]. Die Musik wird begonnen, sobald der Spieler im Hauptmenü ist und soll, wie das Hintergrundbild, den Spieler auf das Spiel einstimmen.

Sobald das Spiel gestartet wird, wird das Ereignis *„BeginPlay"* von der Engine aufgerufen. Im *Level-Blueprint* des Hauptmenüs wird das erstellte *Widget* dem *Viewport*, dem Anzeigefenster des Spielers, hinzugefügt und die Eingabe per Maus aktiviert (siehe Abbildung 4). Der Mauszeiger wurde hierbei explizit eingeblendet. Danach wird der *„MainMenueSound"*-Klasse mitgeteilt, dass die Hintergrundmusik gestartet werden soll.

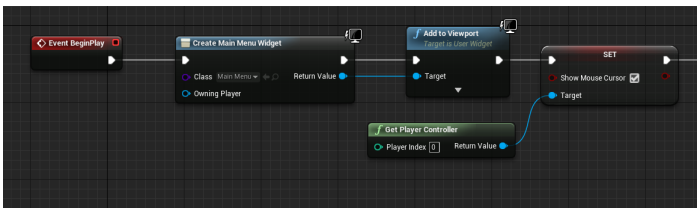

Abbildung 4: Die Logik des Hauptmenü-Levels

[15] Quelle: Hintergrundgeräusche

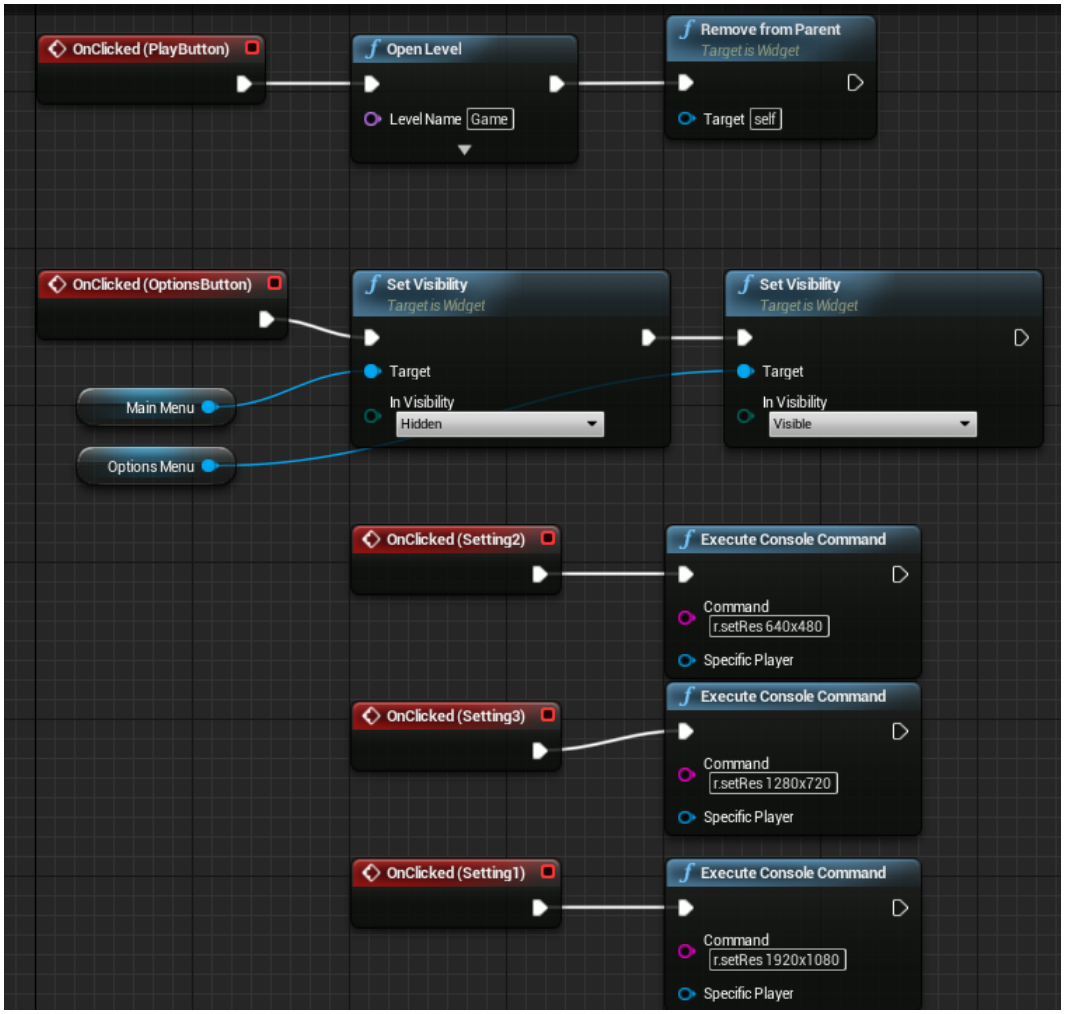

Abbildung 5: Auszug aus der Logik der Menüfunktionen

Mithilfe von *OnClicked*-Ereignissen können die Schaltflächen vom Benutzer angewählt werden und die programmierte Logik wird ausgeführt (siehe Abbildung 5). Wenn der Spieler das Optionsmenü aufrufen will werden alle Bereiche die zum Hauptmenü gehören ausgeblendet und alle Bereiche die zum Optionsmenü gehören eingeblendet. Möchte der Spieler aus dem Optionsmenü wieder hinaus und betätigt die *„Return"*-Schaltfläche, wird die Sichtbarkeit der jeweiligen Bereiche wieder umgekehrt.
Sowohl das Beenden des Spiels als auch die Einstellung der Auflösung funktionieren mithilfe von Windows-Befehlen, die von der *Unreal Engine* ausgeführt werden können.
Für die Erstellung des Hauptmenüs wurde ein offizielles Tutorial von den Entwicklern der *Unreal Engine* verwendet.[Gam15a]

6.5 Die Spielumgebung

In diesem Abschnitt wird sich dem Level-Design gewidmet. Hierbei wird auf die Level-Umgebung und das „Graphical User Interface" (*GUI*) eingegangen und die gefällten Design-Entscheidungen werden erläutert.

6.5.1 Der Konzertsaal

Als Level-Umgebung für das Spiel wurde ein Konzertsaal verwendet. Das 3D-Modell wurde selbstständig mithilfe von „Blender"[16] erstellt und texturiert (siehe Abbildung 7). Als Vorlage diente die Architektur der *Berliner Philharmonie* (siehe Abbildung 6).

Abbildung 6: Die Berliner Philharmonie als Vorlage [17]

Die Architektur sollte realitätsnah aussehen und auch realistisch dargestellt werden, um die Immersion zu maximieren. Damit die Szene nicht leer erscheint und der Spieler den Eindruck erhält in einem realen Konzertsaal, vor einem realen Orchester, zu stehen, wurden für das Orchester Stühle, Notenständer und 3D-Modelle der Instrumente (Violine, Cello, Kontrabass, Schlagzeug, Klavier) eingebunden und in der Szene platziert ([18]). Die Besetzung des Orchesters entspricht der eines klassischen Orchesters.

[16] Blender Webseite: Blender
[17] Quelle: Berliner Philharmonie
[18] Quelle für 3D-Modelle: Turbo-Squid, Archibase

Abbildung 7: Die Berliner Philharmonie - Modell

Der Spieler steht auf einem Podest, etwas erhöht über der ersten Reihe der Musiker. Das Podest selber wurde mit zwei geometrischen Würfeln („Cubes”), die von der *Unreal Engine* bereitgestellt werden, umgesetzt.
Für die Beleuchtung der Bühne wurden Scheinwerfer („Spotlights”) verwendet, die die Bühne aus vier Richtungen ausleuchten. Die Decke wurde mit Punktlichtquellen („Pointlight”) beleuchtet. Die „Spotlight”-Lichtquellen dienen auch der indirekten Beleuchtung, die von der *Unreal Engine* übernommen wird. So wurde die gesamte Architektur ausgeleuchtet.
Die Sound-Dateien, deren Lautstärke der Spieler mithilfe von Handgesten steuert, wurden innerhalb des Levels platziert und durch einen Namensschriftzug des zugehörigen Instrumentes visualisiert (siehe auch Abschnitt 6.6.2). Dies sollte der Orientierung des Spielers in der Szene dienen.
Der Spieler wird in der Szene durch eine Spielfigur repräsentiert, welche aus zwei Armen besteht, die seitlich an einer Kamera angebracht sind und eine „First-Person-Perspektive” simulieren. Dies sollte den Benutzer besser in die Szene integrieren und eine physische Präsenz im Spiel ermöglichen.

6.5.2 Die GUI

Der Spieler-Klasse *„DirigentBP”* wurde das Widget *„VolumeBar”* hinzugefügt (siehe Abbildung 8), welches eine Lautstärke-Anzeige in der oberen linken Bildschirmecke hinzufügt.

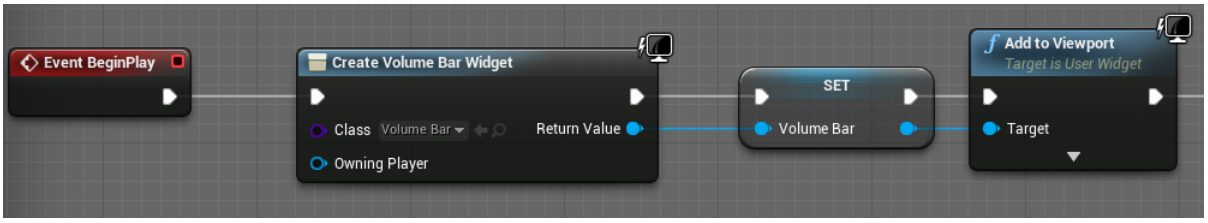

Abbildung 8: Der Lautstärke-Balken wird als „Widget” dem Sichtbereich des Spielers hinzugefügt

Diese Anzeige dient dazu, die aktuelle Lautstärke der Sound-Datei dem Spieler farblich darzustellen. Hierfür wird eine vertikale Box von unten nach oben gefüllt. Unten befindet sich die leiseste Lautstärke (*ppp*) und oben die Lauteste (*fff*), wobei die Lautstärke in musikalischer Notation an der rechten Seite des Balkens angezeigt wird (siehe Abbildung 9).
Solange der Spieler kein Instrument anvisiert (näheres hierzu im Abschnitt 6.12) wird der Balken für die Lautstärke und seine zugehörige Beschriftung nicht angezeigt und als „Hidden”(engl. für „versteckt”) definiert. Sobald ein Instrument anvisiert wird, holt sich die Klasse die Audio-Komponente des Instrumentes (siehe Abschnitt 6.6.2) und setzt diese als Klassen-Variable. Zusätzlich wird der Lautstärke-Balken und die zugehörige Beschriftung sichtbar gemacht (Status „Visible”) und dem Spieler angezeigt.

Abbildung 9: Der Lautstärke-Balken als „GUI” im Spiel

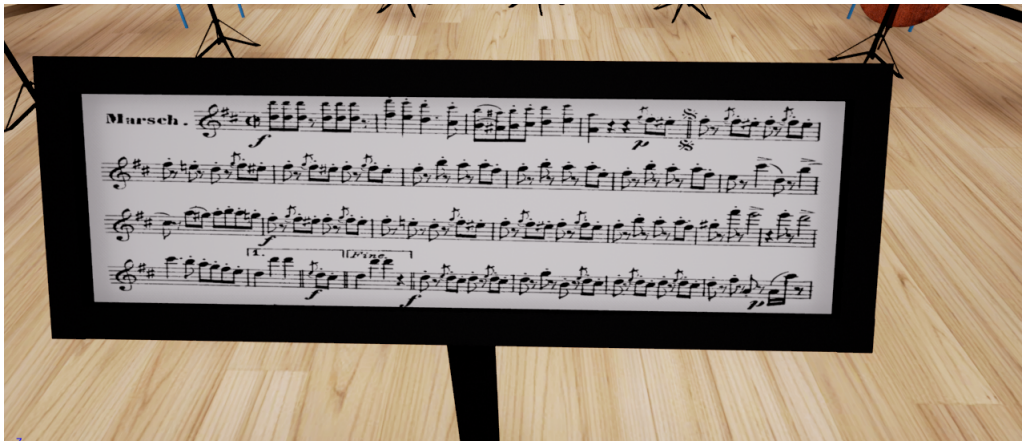

Abbildung 10: Das Dirigenten-Pult

Zusätzlich hat der Spieler als Dirigent ein Notenpult vor sich stehen, welches als eigene Klasse umgesetzt wurde (*„DisplayBP"*). Dieses Pult wurde zu einem Bildschirm umfunktioniert, so dass auf dem Pult die originalen Noten des Liedes angezeigt werden konnten (siehe Abbildung 10). Dafür wurde eine Textur auf eine flache Geometrie gelegt. Das Material der Geometrie wurde als dynamisches Material angelegt mithilfe der „CreateDynamicMaterialInstance"-Funktion der *Unreal Engine* (siehe Abbildung 11) und dadurch wurde es ermöglicht, dass die Textur zur Laufzeit verändert werden kann.

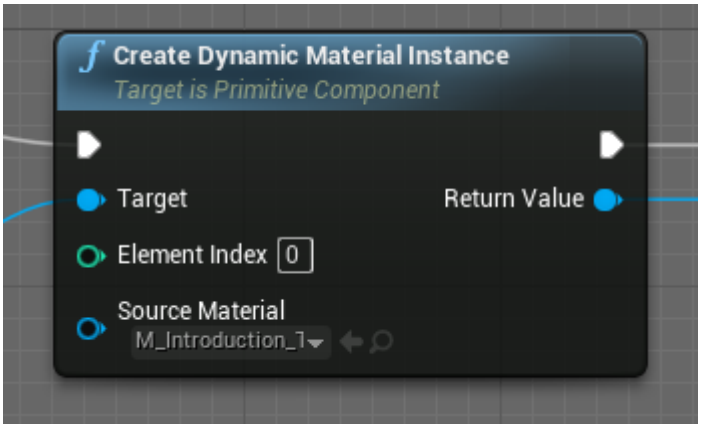

Abbildung 11: Die *„Create Dynamic Material Instance"*-Funktion

Mithilfe einer Zeit-Funktion („Timer"), die zu bestimmten, vom Programmierer festgelegten, Intervallen die Textur auswechselt, werden die Seiten des Stückes automatisch umgeblättert. Damit der „Timer" die richtigen Texturen auswählt, zählt die Klasse eine Integer-Variable *Page* hoch und entscheidet auf Grundlage der aktuellen Zahl der Variable, welche Textur geladen werden soll. Die zu vergehende Zeit (Integer-Variable *Seconds*), bis zum nächsten Wechsel der Textur, wird in Sekunden angegeben und nach jedem Seitenwechsel neu gesetzt.

Ein zweiter „Timer" verringert die Integer-Variable *Seconds* jede Sekunde um eins und sobald die Zeit abgelaufen ist, also *Seconds* den Wert null angenommen hat, wird ein „Change-Image" Event ausgelöst und die neu anzuzeigende Seite wird ausgewählt.

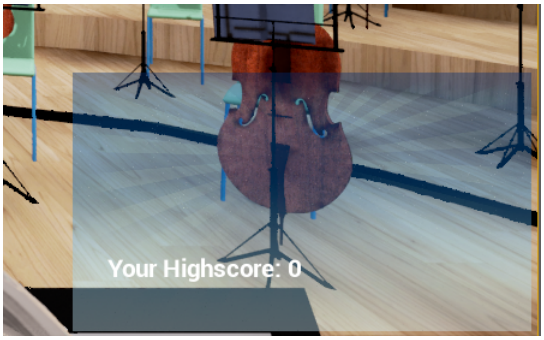

Abbildung 12: Der Highscore wird am Ende des Spiels angezeigt

Am Ende des Spiels wird dem Spieler sein persönlicher Highscore angezeigt (siehe Abbildung 12). Hierfür fügt die Klasse *„DisplayBP"* der Spielerklasse das *„HighscoreWindow"*-Widget hinzu, sobald die Musik beendet ist. Die Punktzahl des Spieler wird von der Klasse *„Highscore_RadetzkyMarch"* berechnet (siehe Abschnitt 6.7) und das *„HighscoreWindow"*-Widget holt sich die benötigte Punktzahl aus dieser Klasse und gibt sie als Text aus.

6.6 Der Sound

Im folgenden Abschnitt wird erklärt, wie die Sound-Dateien für das Orchester erstellt wurden und welches Programm dafür benutzt wurde. Danach werden die im Spiel verwendeten Sound-Klassen erläutert und ihre Funktionen vorgestellt.

6.6.1 Die Sound-Dateien

Im Spiel werden für jedes Stück Sound-Dateien eingebunden, die das entsprechende Musikstück abspielen. Anstatt vorgefertigte *MIDI-Dateien* zu benutzen wurden, mithilfe des Programms *„Music-Maker"* der Firma „Magix"([19]), eigene .wav-Dateien erstellt.
Magix-Music-Maker ermöglicht das Erstellen von eigener Musik. Dafür stellt das Programm eine Vielfalt an Musikinstrumenten (Violinen, Klavier, Schlagzeug u.v.a.) zur Verfügung, die digital aufgenommen wurden. Diese können vom Benutzer zusammengesetzt und abgespielt werden (siehe Abbildung 13). Es muss nur angegeben werden, zu welchem Zeitpunkt welcher Ton von welchem Instrument gespielt

[19] Software: Magix Music Maker

werden soll. Hierfür dient eine virtuelle Klaviatur, um den richtigen Ton auszuwählen, welcher durch einen Balken repräsentiert wird (siehe Abbildung 14). Durch Veränderung der Länge des Balkens kann der Notenwert eingestellt werden. Darüber hinaus kann der Benutzer die Lautstärke und Dynamik verändern und weitere Klangeigenschaften individuell anpassen.

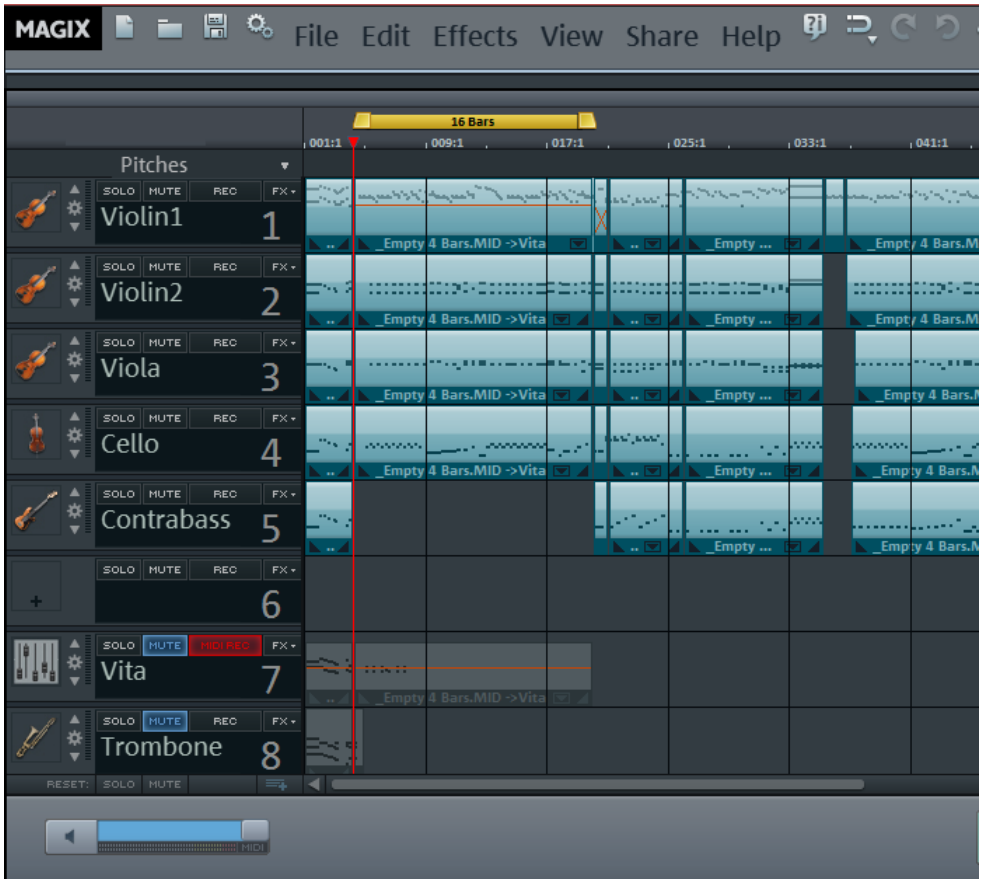

Abbildung 13: Benzteroberfläche von Magix Music Maker

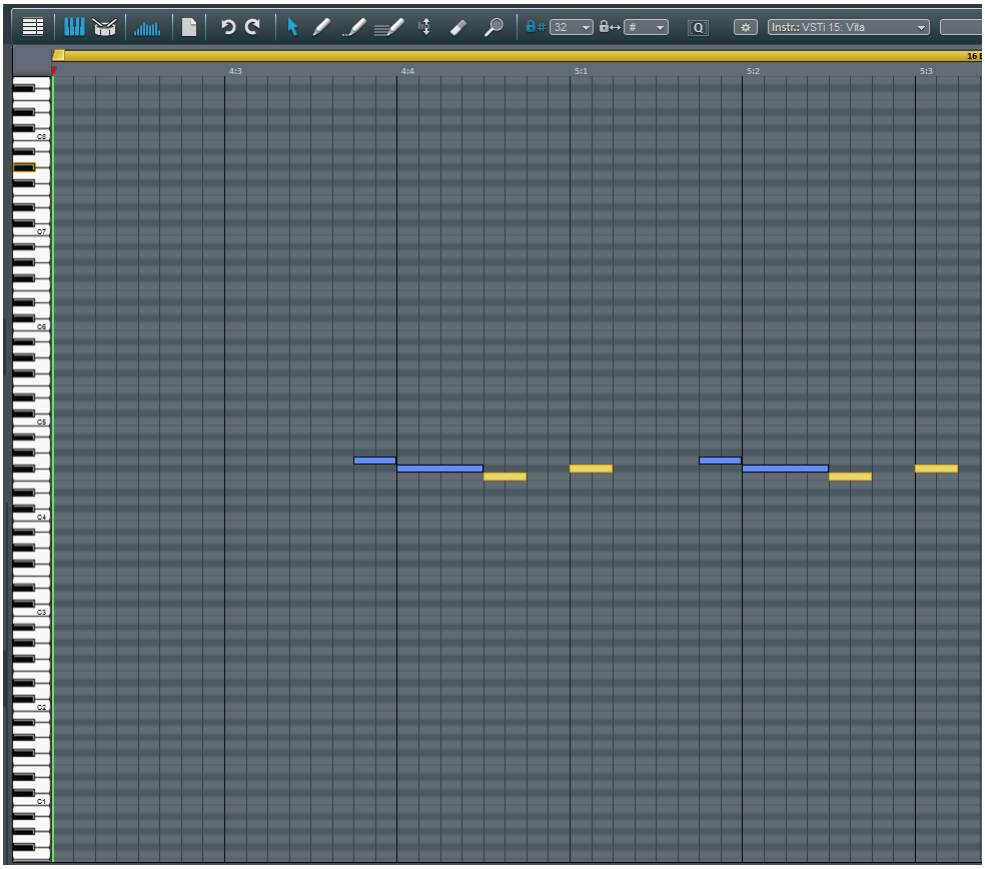

Abbildung 14: Noteneingabe von Magix Music Maker

Aus der originalen Orchester-Partitur wurde das Stück „Radetzky-March" von Johann Strauß erstellt ([20]). Insgesamt wurden sechs Instrumente (Violine 1, Violine 2, Viola, Cello, Kontrabass, Schlagzeug) in das Programm übersetzt. Für die erste Version des Spiels wurde das Musikstück als einzelne *.wav-Datei* exportiert, da der Spieler zunächst nur das gesamte Orchester leiten sollte und keine einzelnen Instrumente. Aus diesem Grund wurde darauf geachtet, dass die Dynamik der Instrumente untereinander mit der Dynamik der Original-Partitur übereinstimmte. Für den Import in die *Unreal Engine* musste die .wav-Datei eine *16-bit-PCM* Datei sein, weil nur dieses Format unterstützt wird.
Die zweite Version des Spiels erlaubte es dem Spieler auch die Lautstärke einzelner Instrumente zu steuern (siehe Abschnitt 6.12). Hierfür wurde für jedes Instrument eine einzelne 16-bit .wav-Datei exportiert und in die *Unreal Engine* importiert.

[20] Quelle der Orchesterpartitur: Radetzky Marsch

6.6.2 Die Sound-Klassen

Für jedes Instrument, das der Spieler im späteren Spiel leiten sollte, wurde eine eigene Blueprint-Klasse erstellt (beispielsweise *„Violin01_BP"*). Als Grundlage diente ein „Text-Render-Actor" ([21]), welcher es ermöglicht einen Text als 3D-Objekt in einer Szene zu platzieren. Der im Spiel angezeigte Text entsprach dem jeweiligen Instrument, damit im späteren Spiel der Spieler diesen Text als Orientierung im Level nutzen würde können.
Neben dem Text-Objekt, der *„SpriteComponent"*, erhielt jede Klasse eine Audio-Komponente, die den Verweis auf die, dem Instrument zugehörige, *.wav-Datei* enthielt. Eine *Bounding Box* („Box Collision"-Komponente) wurde für die späteren Gesten eingefügt, damit eine Kollisionserkennung möglich war (siehe auch Abschnitt 6.12). Um die Kinect-Steuerung zu ermöglichen, brauchte die Klasse eine *Kinect-Komponente*, das sogenannte „KinectInterface". Diese Komponente ermöglichte den Zugriff auf die, von der Kinect bereitgestellten, Funktionen und Informationen.
Zuletzt erhielt jede Klasse ein *„InstrumentComponent_BP"*-Blueprint. In dieser Klasse wurde die Logik für die Übersetzung der Gesten des Spielers implementiert, die es ermöglicht, die Lautstärke der Audio-Komponente zu verändern (siehe Abschnitt 6.11).

Im „Event-Graph" der Klasse, wo die visuelle Skriptsprache programmiert wird, wurden drei Ereignisse („Events") benutzt. Sobald das Spiel startet wird das, von der *Unreal Engine* vordefinierte, Ereignis „EventBeginPlay" aufgerufen. Die Audio-Komponente der Klasse wird in einer Klassen-Variable *Music* gespeichert und die Musik wird zunächst gestoppt.
Das *Level-Blueprint* schickt an alle Instrument-Klassen das Ereignis „MusicStarted", sobald die Musik vom Spieler gestartet wurde (siehe Abschnitt 6.10). Dieses Ereignis bewirkt in der jeweiligen Klasse, dass die *Boolean*-Variable *isStarted* auf „true" gesetzt wird und die Musik zu starten beginnt.
Die Spieler-Klasse schickt das Ereignis „PlayerGestureActive" (siehe Abschnitt 6.12) an die Instrument-Klassen. Wenn die Musik der Klasse bereits gestartet wurde, dann wird die Funktion „VolumeGestureControl" aufgerufen. Hierfür übergibt jede Klasse der Funktion die *Audio-Komponente* und das *Kinect-Interface*. Diese Funktion befindet sich in der *„InstrumentComponent_BP"*-Komponente und wird dort aufgerufen. In Abschnitt 6.11 wird diese Funktion erklärt.

Am Ende des Spiels sollte der Spieler Applaus von dem virtuellem Publikum erhalten. Hierfür musste eine Instrument-Klasse eine zusätzliche Funktion bereitstellen. In der Violinen-Klasse *„Violin01_BP"* wurde die Funktion „isMusicFinished" implementiert. Sobald das Spiel gestartet und das Ereignis „EventBeginPlay" ausgelöst wurde, wird zusätzlich eine Zeitfunktion („Timer") aktiviert. Dieser „Timer" ruft die Funktion jede Zehntelsekunde auf und speichert seinen „Handle" in der

[21] Unreal Engine Wiki [Gam15b]: Text Render Actor

Variable *isMusicFinished_Handle*. Mithilfe des „Handles" lässt sich die Zeitfunktion eindeutig identifizieren.
In der Funktion „isMusicFinished" wird überprüft, ob die Musik gestartet wurde. Sollte dies der Fall sein wird zusätzlich überprüft, ob die Audio-Komponente noch abgespielt wird. Dies ist mit der integrierten Funktion „isPlaying" der *Unreal Engine* möglich. Sollte die Musik nicht mehr abgespielt werden, wird an der Position des Spielers ein Applaus-Sound abgespielt mithilfe der internen Funktion „Play Sound at Location" und der „Timer" wird mithilfe der gespeicherten Variabel *isMusicFinished_Handle* deaktiviert.

6.7 Der Highscore

Damit aus der „Virtual Reality"-Anwendung ein Computerspiel werden würde, wurde sich dazu entschieden einen Highscore einzubauen, welcher zur Motivation dienen sollte, um ein Musikstück mehrmals und in unterschiedlichen Schwierigkeitsstufen zu spielen.
Der Spieler erhält Punkte, indem die Lautstärke so eingestellt wird, wie sie auf dem originalen Notenblatt eingetragen ist. Am Ende wird der Highscore in einer separaten *GUI* (siehe Abschnitt 6.5.2, Abbildung 12) eingeblendet und ein Applaus-Sound wird als Zeichen, dass das Musikstück zu Ende ist, abgespielt.

In der *Unreal Engine* wurde der Highscore als Blueprint-Klasse umgesetzt (*„Highscore_RadetzkyMarch"*). Die Klasse besitzt drei Integer-Variablen: Jeweils eine Variable, die die Punktzahl (*Highscore*), die Anzahl der Lautstärke-Überprüfungen (*ChecksDone*) und eine bestimmte Sekundenzeit festhält (*Seconds*).
Wenn das Spiel gestartet wird, wird das Ereignis „EventBeginPlay" aufgerufen und die Audio-Komponente des aktuellen Stückes wird als Klassen-Variable gespeichert. Der Startwert für die Sekunden, die angeben, wann die nächste Lautstärke-Überprüfung stattfinden soll, wird ebenfalls zu diesem Zeitpunkt gesetzt. Im Fall des Radetzky-Marsches waren es zehn Sekunden.
Solange die Musik noch nicht abgespielt wird, wird keine Funktion in der *„Highscore_RadetzkyMarch"*-Klasse ausgeführt. Das *Level-Blueprint* schickt das Ereignis „EventMusicStarted", woraufhin eine Zeitfunktion („Timer") aktiviert wird, die die Funktion „CheckVolumeFunction" jede Sekunde aufruft. Diese Funktion reduziert die Integer-Variable *Seconds* jede Sekunde um eins. Sobald der Wert null erreicht, wird der Wert von *Seconds* erneut auf Zehn gesetzt und das Ereignis „CheckVolume" wird aufgerufen. Dieses Ereignis wird, im Falle des Radetzky-Marsch, alle zehn Sekunden aufgerufen.

„CheckVolume" überprüft, wie viele Lautstärke-Kontrollen bisher ausgeführt wurden. Für den Radetzky-Marsch wurden insgesamt 18 Kontrollen einprogrammiert. Sobald die richtige Kontrolle mithilfe von if-Funktionen, welche in der *Unreal Engine* „Branch" heißen, gefunden wurde, wird die Variable *ChecksDone* um eins er-

höht und eines von drei Ereignissen wird aufgerufen: „CheckVolume_P", „CheckVolume_F" oder „CheckVolume_FF". In der Orchesterpartitur des Radetzky-Marsches gibt es die entsprechenden Lautstärke-Angaben, die bei der entsprechenden Lautstärke-Kontrolle überprüft werden.
Aus der Audio-Komponente der *„Highscore_RadetzkyMarch"*-Klasse wird der aktuelle Wert der Variable *Volume Multiplier* abgefragt, welche die Lautstärke-Information enthält und einen Wert zwischen 0.0 und 1.0 annehmen kann. Die Lautstärke wurde für das Spiel und speziell für den Radetzky-Marsch in drei Bereiche gegliedert: Die Lautstärke *p* entspricht einem Wert zwischen 0.45 und 0.55, die Lautstärke *f* entspricht einem Wert zwischen 0.6 und 0.9 und die Lautstärke *ff* entspricht einem Wert ab 0.8.
Wenn der Lautstärke-Wert der Audio-Komponente zum Zeitpunkt der Kontrolle dem jeweiligen Bereich entspricht, erhält der Spieler zehn Punkte, die auf die Variable *Highscore* addiert werden.

Der Highscore als Spielelement funktioniert nur, wenn der Spieler ein ganzes Orchester mit einer Audio-Datei steuert. Sollte der Spieler die zweite Version des Spiels spielen, in der alle Instrumente einzeln gesteuert werden, steht die Funktion des Highscores nicht zur Verfügung.
Insgesamt war es möglich 190 Punkte bei dem Stück „Radetzky-Marsch" zu erspielen. Die erreichte Punktzahl wird dem Spieler in einer separat implementierten *GUI* angezeigt (siehe Abschnitt 6.5.2).

6.8 Die Spieler-Klasse

Das programmierte Spiel wurde auf Grundlage eines vorgefertigten Projektes der *Unreal Engine* entwickelt. Dieses Projekt steht als Vorlage bei der Erstellung eines neuen Projektes zur Verfügung und nennt sich „First Person Template" [22].
Eine Vorlage der *Unreal Engine* stellt vordefinierte „Blueprints" zur Verfügung, damit ein Entwickler keine grundlegenden Funktionen zu programmieren braucht. Dazu gehören zum Beispiel ein Blueprint für das Level und den Spieler. Das Spieler-Blueprint stellt eine Spielfigur, Funktionen zum Steuern der Spielfigur und im Falle des „First Person Template" die Funktion, eine Waffe abzufeuern, bereit. Das zugehörige Blueprint heißt *„FirstPersonCharacter"* und wurde für das selbst programmierte Spiel in *„DirigentenBP"* umbenannt.

Keine der vordefinierten Funktionen der Vorlage wurden für das programmierte Spiel verwendet und wurden aus der Klasse entfernt. Insgesamt drei neue Funktionen wurden einprogrammiert: Die Funktion „Trace", welche die Blickrichtung des Spielers verfolgt und es ermöglicht, Objekte auszuwählen (siehe Abschnitt 6.12), die Funktion „DetermineSound", welche überprüft, welches Instrument vom Spieler ausgewählt wurde (siehe Abschnitt 6.12) und die Funktion „GetHandDif-

[22] Unreal Engine Wiki [Gam15b]: First Person Template

ference_Func", mit deren Hilfe die Maussteuerung ersetzt wurde (siehe Abschnitt 6.9). Alle vordefinierten Variablen des *„FirstPersonCharacter"*-Blueprint wurden gelöscht und jede Funktion benutzt eigene Variablen, die angelegt wurden.
Nur die Geometrie der Spieler-Figur wurde unverändert übernommen, weil bei dem programmierten Spiel eine *Ego-Perspektive* genutzt werden sollte, bei der die Hände der Spielfigur sichtbar sein würden. Diese Perspektive war in der Vorlage bereits enthalten und musste nicht selbstständig programmiert werden.

6.9 Die Maussteuerung als Handbewegung

Der Spieler sollte die Möglichkeit erhalten sich in der 3D-Szene umsehen zu können ohne eine Maus dafür benutzen zu müssen. Eine Maus hätte der *virtuellen Realität* ihre Glaubwürdigkeit genommen, da der Spieler damit in direkten Kontakt mit der realen Umgebung gekommen wäre. Auch in Hinblick auf die verschiedenen Gesten, die es nötig machten, sich in der Szene umsehen zu können, war ein Ersatz für die Maus notwendig.

Im Blueprint des Spielers *„DirigentBP"* wird für jedes Einzelbild („Frame"), mithilfe eines „EventTick"-Ereignisses, die Funktion „GetHandDifferenceFunc" aufgerufen. Zuerst wird aus der *Kinect-Komponente* der *„DirigentenBP"*-Klasse (siehe Abschnitt 6.8) der erkannte Körper des Spielers ausgelesen („Get Centered Body", siehe Abbildung 15). Es wird überprüft, ob der Zeigefinger der rechten Hand hochgehalten wird. Dies geschieht, indem die Funktion „Get Hand State as Execution" (siehe Abbildung 16) die Informationen über die rechte Hand ausliest und überprüft, ob die Hand sich in einer sogenannte „*Lasso-Geste*" befindet. Diese Handhaltung wird von dem *Kinect-Plug-In* (siehe Abschnitt 6.3) unterstützt und bezeichnet eine Handhaltung, bei der die ausgestreckten Finger bei geschlossener Handfläche zu sehen sind. Auch das gleichzeitige Vorzeigen mehrerer Finger, beispielsweise des Zeige- und Ringfingers, würde als *Lasso-Geste* gelten. Diese Funktion ermöglicht darüber hinaus die Abfrage, ob die Hand des Spielers offen oder zur Faust geschlossen ist. Sowohl die linke als auch die rechte Hand können separat abgefragt werden.
Sobald die *Lasso-Geste* erkannt wurde, wird die Position der rechten Hand in Relation zur Hüfte ausgelesen („Get Joint Relative Position", siehe Abbildung 17) und als Vektor in der Variable *HandPosition* gespeichert. Die „Get Joint Relative Position"-Funktion ermöglicht den Zugriff auf alle möglichen, von der Kinect erfassten, Körperpunkte des Spielers und berechnet den Differenzvektor zwischen zwei gewählten Punkten. Hierbei werden der Startpunkt und Endpunkt des Vektors festgelegt. Unter anderem ist es möglich die Handposition der rechten Hand in Relation zum Kopf oder zum Knie des Spielers auslesen zu lassen.

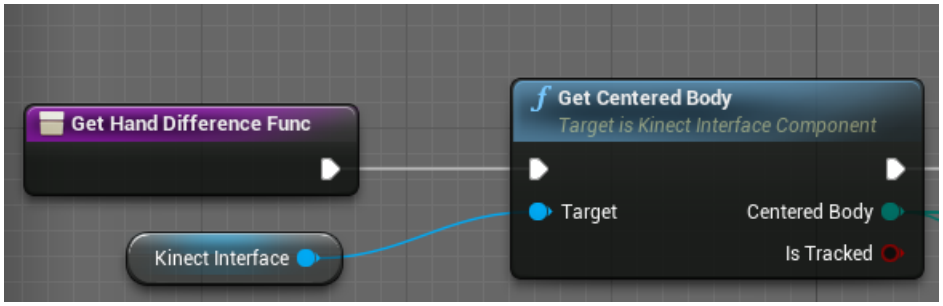

Abbildung 15: Die *GetCenteredBody*-Funktion

Im nächsten Schritt sollte die Differenz zwischen der Handposition im vorherigem Frame und dem aktuellem Frame berechnet werden, um diese Differenz als Maus-Eingabe interpretieren zu können.
Hierfür gibt es die Vektor-Variable *LastFrameHandPosition*, die die Handposition vom letzten Frame speichert und die Boolean-Variable *Movement*, die überprüft, ob eine Handbewegung ohne Lasso-Geste stattgefunden hat.
Wenn der Spieler die Lasso-Geste nicht benutzt und nur die offene oder geschlossene Hand von der Kamera erfasst wird, wird *Movement* auf „true" gesetzt. Sollte der ausgestreckte rechte Zeigefinger erkannt werden, wird zuerst die Variable *LastFrameHandPosition* auf den Wert der Variable *HandPosition* gesetzt. Dies geschieht, damit die Steuerung der Sicht keine ungewollten Sprünge verursacht.

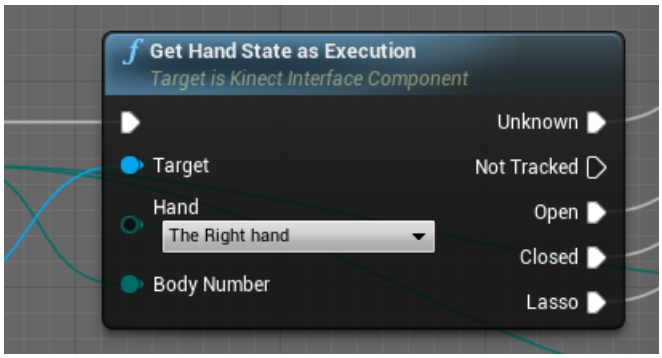

Abbildung 16: Die *GetHandStateAsExecution*-Funktion

Abbildung 17: Die *GetJointRelativePosition*-Funktion

Wird die *Lasso-Geste* über einen Frame lang erkannt, wird die Differenz zwischen der letzten Handposition und der aktuellen Handposition berechnet. Die Vektoren enthalten Informationen zu allen drei Achsen X, Y und Z. Nur die Differenz der Z- und Y-Achsen werden berechnet, da die Steuerung der Sicht nur auf zwei Bildschirmachsen unterstützt wird. Nach der Berechnung der Differenzen wird die Variable *LastFrameHandPosition* erneut mit der Variable *HandPosition* überschrieben.
Damit die Sicht des Spielers nicht durch sehr kleine Handbewegungen anfängt zu zittern und damit die Sichtverschiebung ungleichmäßig wirkt, wurde die Bewegung der Hand zusätzlich geglättet. Diese Glättung wird durchgeführt, wenn der Differenzbetrag der Z- und Y-Achsen jeweils größer als 0.06 beträgt. Sollte dies nicht der Fall sein, wird die Sichtveränderung mithilfe des Ereignisses „Update-Look" angewandt.
Falls der Differenzbetrag unter dem Wert 0.06 liegt, wird überprüft, ob der Differenzwert negativ oder positiv ist. Dementsprechend wird die „Float"-Variable [23] *HandDPI* zu der Differenz addiert oder von der Differenz subtrahiert. Dies sollte bewirken, dass die Handbewegung gleichmäßiger wirkt und leichtes Zittern der Hände und Ungenauigkeiten der Kinect-Erkennung ausgeglichen werden würden. Nach der Berechnung wird ebenfalls das Ereignis „UpdateLook" aufgerufen.

Das Ereignis „Update-Look" ersetzte in der *„DirigentBP"*-Klasse die Maussteuerung. Die von der Maus bereitgestellten Informationen über die Y- und Z-Achsenverschiebungen wurden durch die berechneten Variablen *DifferenceY* und *DifferenceZ* ersetzt. Die Funktion der *Unreal Engine* „AddControllerYawInput" verschiebt die Sicht des Spieler in Y-Richtung und die Funktion „AddControllerPitchInput" verschiebt die Sicht des Spielers in Z-Richtung.
So war es möglich die Maussteuerung durch die Handbewegung des Spielers zu ersetzen.

[23] engl. für „Fließkommazahl"

6.10 Die erste Geste: Orchester starten

Als erste Dirigenten-Geste wurde eine Geste implementiert, die die Musik starten lässt. Dies sollte dazu dienen dem Spieler von Anfang an das Gefühl zu vermitteln, die Kontrolle über das Orchester zu haben und dieses anleiten zu können. Bei der Geste wurde sich ein Vorbild an echten Dirigenten genommen, welche zu Beginn eines Stückes mit einer simplen Handbewegung die Aufmerksamkeit der Musiker erhalten, bevor sie den Start mit einer häufig aufwärts gerichteten Handbewegung signalisieren.

Der Spieler zeigt als erstes den ausgestreckten Zeigefinger der linken Hand vor und erhält damit automatisch die Aufmerksamkeit des virtuellen Orchesters. Nun wird die nächste Geste freigeschaltet, die es dem Spieler ermöglicht die Musik zu starten. Hierfür muss der Spieler beide Hände offen über den Kopf heben, woraufhin die Musik gestartet wird und das virtuelle Orchester zu spielen beginnt. Ab diesem Zeitpunkt werden alle Gesten freigeschaltet und der Spieler kann das Orchester leiten (siehe Abschnitt 6.11).

Realisiert wurde die Geste innerhalb des *Level-Blueprints*. Sobald das Spiel gestartet wird, wird eine Zeitfunktion („Timer") aktiviert, die jede zehntel Sekunde die Funktion „RecognizeStartGesture" aufruft. Die „Timer"-Funktion wird als Variable „StartGestureFunction" abgespeichert, damit die Zeitfunktion beendet werden kann, sobald die Startgeste ausgeführt wurde.
Die Funktion „RecognizeStartGesture" holt sich aus der Klasse *„DirigentBP"* (siehe Abschnitt 6.8) die *Kinect-Komponente*. Da die Funktion in der Level-Klasse steht, hat sie Zugriff auf alle im Level platzierten Objekte. Mithilfe der *Kinect-Komponente* liest die Funktion die Informationen über den Körper des Spielers aus. Hierfür wird die Funktion „GetCenteredBody" verwendet. Die *Boolean*-Variable *StartGestureReady* entscheidet darüber, ob der Spieler die Aufmerksamkeit des Orchesters benötigt oder diese bereits erhalten hat. Wenn der Spieler die Aufmerksamkeit nicht hat, dann überprüft die Funktion mit der „Get Hand State as Execution"-Funktion, ob der Spieler eine *Lasso-Geste* mit der linken Hand ausführt. Sollte dies der Fall sein wird *StartGestureReady* auf „true" gesetzt.

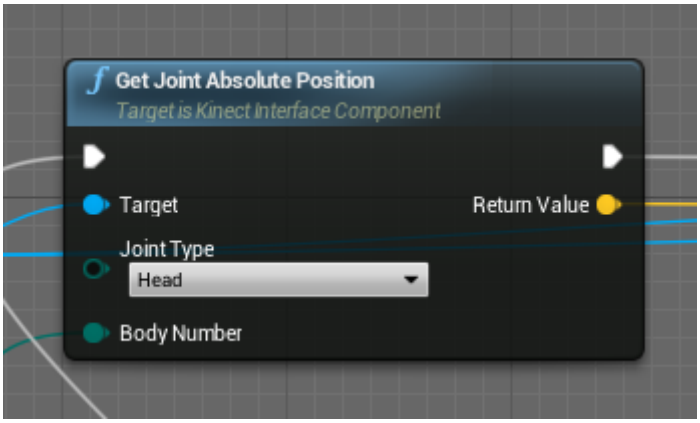

Abbildung 18: Die *Get Joint Absolute Position*-Funktion

Sobald *StartGestureReady* auf „true" gesetzt wurde, liest die Funktion die Position des Kopfes des Spielers aus („Get Joint Absolute Position") und speichert den Vektor in der Variable *HeadPosition* (siehe Abbildung 18). Mit der „Get Joint Absolute Position"- Funktion des *Kinect-Plugins* lassen sich die Positionen von allen erkannten Körperpunkten auslesen, welche als Vektoren zurückgegeben werden.
Zusätzlich zur Position des Kopfes werden auch die Positionen der rechten und linken Hand ausgelesen und in entsprechenden Variablen gespeichert (*RightHandPosition*, *LeftHandPosition*). Es wird überprüft, ob die beiden Hände sich über dem Kopf oder auf Höhe des Kopfes befinden. Hierfür werden die z-Koordinaten der Handpositionen von der z-Koordinate der Kopfposition jeweils subtrahiert und es wird überprüft, ob die Differenz der z-Koordinaten, die die Höhe im *Kinect-Koordinatensystem* darstellen, größer oder gleich null entspricht.

Abbildung 19: Die *Clear Timer by Handle*-Funktion

Bevor die Musik gestartet wird sollte der Spieler seine Hände ein paar Sekunden über dem Kopf halten, um die Reaktionszeit des Orchesters zu simulieren. Dafür wurde eine Variable *Timer* erstellt, die auf einen festen Zeitwert gesetzt wird. In der vorliegenden Version des Spiels wurde die Variable auf den Wert 120 gesetzt, was zwei Sekunden entspricht, weil der Wert jeden „Frame" um eins verringert wird.
Sobald *Timer* den Wert null erreicht hat, wird an jede Instrument-Klasse, die sich im Level befindet, das Ereignis „MusicStarted" geschickt (siehe Abschnitt 6.6.2). Der GUI-Klasse *„DisplayBP"* und der Highscore-Klasse wird dasselbe Ereignis

mitgeteilt, damit diese Klassen ihre Funktionen starten können(siehe Abschnitt 6.7 und 6.5.2).
Zum Schluss wird die Zeitfunktion, die *Timer* jeden „Frame" um eins verringerte, mit „Clear Timer by Handle" gestoppt (siehe Abbildung 19).

6.11 Die zweite Geste: Lautstärke regeln

Nachdem der Spieler die Musik gestartet hat (siehe Abschnitt 6.10), hat der Spieler die Möglichkeit die Lautstärke des Orchesters mithilfe einer weiteren Geste zu steuern.
Durch heben und senken der Hände, während diese zur Faust geschlossen sind, wird die Lautstärke der Musik entsprechend angehoben oder gesenkt. Ein Dirigent signalisiert einem Orchester durch das Anheben oder Absenken seiner Hände, ob die Musiker leiser oder lauter spielen sollen. An dieser Bewegung wurde sich ein Vorbild genommen. Damit der Spieler nicht aus Versehen die Lautstärke der Musik verändert, wurde sich dazu entschieden die Hände zu Fäusten schließen zu lassen, während die Lautstärke verändert werden soll. Dies sollte sicherstellen, dass keine ungewollten Veränderungen an der Lautstärke vorgenommen werden können.
In der ersten Version des Spiels gab es nur eine Musik-Datei, bei der die Lautstärke verändert werden konnte. In der zweiten Version konnte die Lautstärke jedes einzelnen Instrumentes beeinflusst werden. Die Mechanik der Steuerung der Lautstärke ist bei beiden Versionen gleich und es können somit unterschiedlich viele Musik-Dateien genutzt werden. Besonderheiten bei der Steuerung mehrerer Musik-Dateien wird im Abschnitt 6.12 erklärt.

Für die Steuerung der Lautstärke wurde die Blueprint-Klasse *„InstrumentComponent_BP"* erstellt. Diese Klasse enthält nur eine Funktion, die die Lautstärke einer übergebenen Audio-Datei verändert, indem die Handbewegung des Spielers erfasst wird.
Wenn die Funktion aufgerufen wird, wird die übergebene Audio-Datei von der Klasse als Variable *Music* gespeichert. Zusätzlich erwartet die Funktion eine *Kinect-Komponente*, da die Klasse keine eigenen Komponenten enthält. Diese *Kinect-Komponente* wird als Variabe *KinectInterface* gespeichert.
Wie auch bei den Gesten aus Abschnitt 6.9 und 6.10 werden die, von der *Kinect*, erfassten Informationen ausgelesen. Es wird als erstes kontrolliert, ob die rechte und linke Hand geschlossen sind oder offen gehalten werden. Sollten sich die Hände in einer geöffneten Haltung befinden, wird die Boolean-Variable *Movement* auf „true" gesetzt. Mithilfe der Funktion „GetDistanceBewteenJoints" wird die Entfernung der rechten Hand des Spielers zur Hüfte berechnet und als *Float*-Wert zurückgegeben (siehe Abbildung 20). Diese Funktion, die von dem *Kinect-Plug-In* (siehe Abschnitt 6.3) bereitgestellt wird, ermöglicht die Berechnung der Entfernung zwischen zwei beliebigen Körperpunkten, die von der Kinect erkannt werden. Hierfür werden ein Start- und Endpunkt festgelegt.

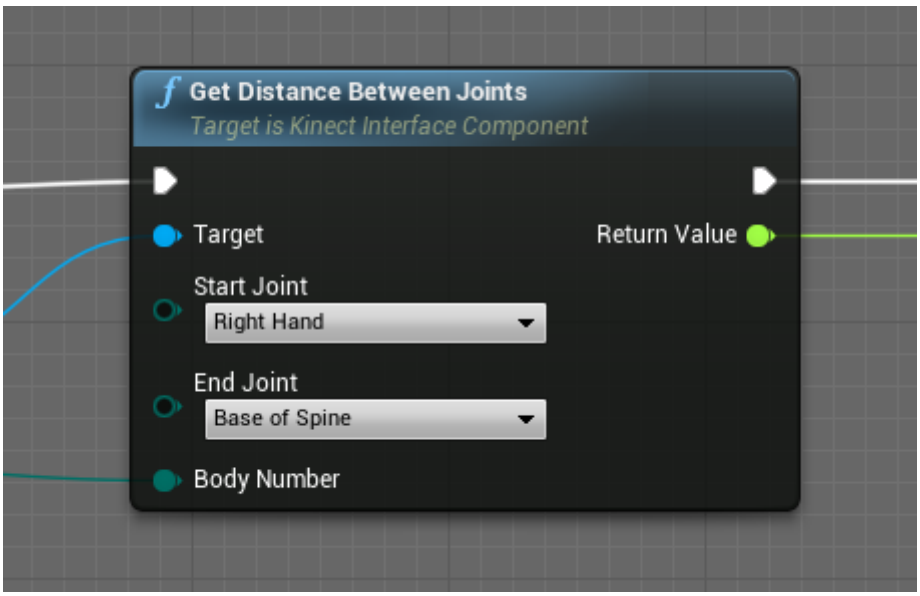

Abbildung 20: Die *Get Distance Between Joints*-Funktion

Die Entfernung der rechten Hand zur Hüfte wird als *CurrentDifference*-Variable gespeichert. Sollte der Spieler die Hand geöffnet haben und nicht zur Faust geschlossen, wird die Variable *LastFrameDifference* auf den Wert von *CurrentDifference* und *Movement* auf „false" gesetzt. Wie bei der Maussteuerung (siehe Abschnitt 6.9) sollten sprunghafte Veränderungen bei der Einstellung der Lautstärke vermieden werden. Aus diesem Grund wird die Bewegung der Hände zwischen zwei „Frames" berechnet und auf die Lautstärke angewandt.
Sobald der Spieler beide Hände geschlossen hält und die Variable *Movement* auf „false" steht, startet die Berechnung. Zunächst wird überprüft, ob sich die Entfernung der Hand zur Hüfte verringert oder erhöht hat, um zu entscheiden, ob die Lautstärke erhöht oder verringert werden soll.
Wenn der Spieler die Hände hebt wird im nächsten Schritt kontrolliert, ob die Variable *VolumeMultiplier* der Sound-Datei, welche die Lautstärke darstellt, noch weiter erhöht werden kann oder ihren maximalen Wert erreicht hat. Falls die maximale Lautstärke erreicht wurde, wird keine weitere Berechnung durchgeführt. Andernfalls wird die Differenz der Entfernung von der Hand zur Hüfte vom vorhergegangenem und aktuellem „Frame" berechnet. Um diese Differenz wird die Variable *VolumeMultiplier* mithilfe der Funktion „SetVolumeMultiplier" erhöht (siehe Abbildung 21).

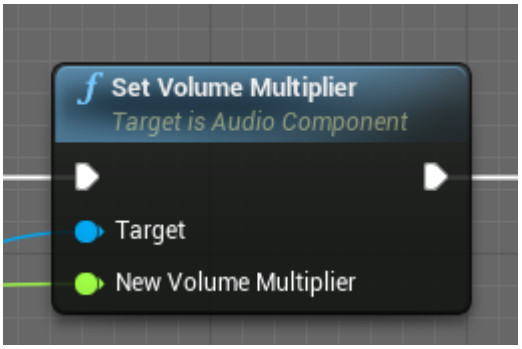

Abbildung 21: Die *Set Volume Multiplier*-Funktion

Sollte die Differenz zu gering sein, wird ein zusätzlicher Wert von 0.02 addiert. Für die Überprüfung wird eine „InRange(float)"-Funktion genutzt, welche überprüft, ob ein eingegebener „Float"-Wert in einem festgelegten Intervall liegt (siehe Abbildung 22). Dies wurde einprogrammiert, damit die Lautstärke gleichmäßig angehoben werden konnte und es möglich wäre mit einer Handbewegung die maximale Lautstärke zu erreichen.

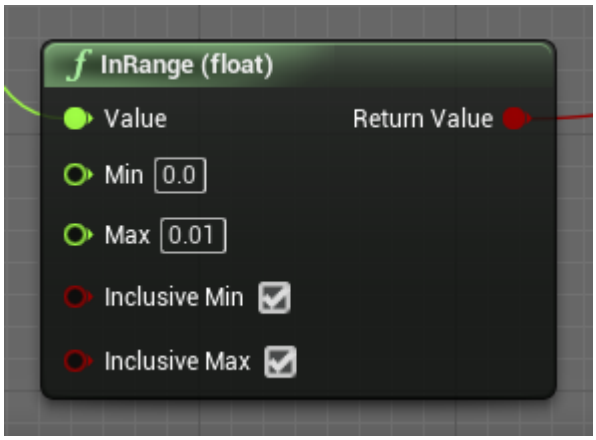

Abbildung 22: Die *In Range (float)*-Funktion

Falls der Spieler die Hände senken sollte, werden die gleichen Berechnungen ausgeführt. Es wird jedoch überprüft, ob die Lautstärke der Audio-Datei weiter gesenkt werden kann oder bereits auf null steht. Anstatt *VolumeMultiplier* zu erhöhen, wird die Differenz der Entfernung von der Hand zur Hüfte vom vorherigen und aktuellem „Frame" von *VolumeMultiplier* subtrahiert. Auch bei dieser Berechnung wird zusätzlich kontrolliert, ob die Differenz zu gering ist, um die Differenz gegebenenfalls um 0.02 zu erhöhen.
Zum Schluss wird die *LastFrameDifference*-Variable wieder auf den Wert der *CurrentDifference*-Variable gesetzt.

6.12 Instrumente dirigieren

Mithilfe der zweiten Geste, welche im Abschnitt 6.11 besprochen wurde, kann der Spieler die Lautstärke einer Audio-Datei verändern. Im Rahmen des programmierten Spiels bedeutete dies, dass der Spieler bereits die Rolle eines Dirigenten einnehmen konnte, doch hatte der Spieler in der ersten Version des Spiels keine Möglichkeit einzelne Instrumente zu dirigieren. Es fehlte eine Funktion, die es ermöglichen würde einzelne Instrumente auszuwählen, um die zweite Geste auf dieses Instrument anwenden zu können.
Eine solche Funktion wurde für die zweite Version des Spieles implementiert. Der Spieler konnte sich in der Spielwelt mit der rechten Hand umsehen (siehe Abschnitt 6.9). Diese Geste wurde genutzt, um im Level platzierte Instrumente auswählen zu können. Vom Spieler aus wird ein Strahl in die Spielwelt geschickt. Sobald dieser Strahl auf ein Instrument trifft, wird das Instrument farblich gekennzeichnet und aktiv geschaltet. Der Lautstärke-Balken wird eingeblendet (siehe Abschnitt 6.5.2) und der Spieler kann mithilfe der zweiten Geste die Lautstärke des ausgewählten Instrumentes verändern. Wenn der Spieler kein Instrument ausgewählt hat, kann keine Lautstärke verändert werden. Um dies dem Spieler auch graphisch zu signalisieren, wird der Lautstärke-Balken ausgeblendet.

In der Klasse *„DirigentBP"* wird, sobald das Spiel gestartet wird, die Funktion „Trace" mithilfe einer Zeitfunktion („Timer") aktiviert. Der „Timer" ruft die Funktion jede zehntel Sekunde auf. In der Funktion wird mithilfe einer weiteren Funktion, „LineTraceByChannel"(siehe Abbildung 23), ein Strahl in die Spielwelt geschickt. Hierfür werden ein Start- und Endpunkt angegeben. Der Startpunkt entspricht der Position der Kamera in Welt-Koordinaten, die mit der Funktions „GetWorldLocation" ausgelesen werden kann. Für den Endpunkt wird zunächst die Variable *„ForwardVector"* der Kamera ausgelesen. Dies ist der Vektor, der von der Kamera aus nach vorne zeigt. Dieser Vektor wird um einen festen Betrag verlängert. In dem programmierten Spiel beträgt dieser Betrag 10.000, damit der Strahl in der Spielwelt lang genug war, um die Instrumente zu erreichen. Zuletzt wird auf die Startposition die Variable „ForwardVector" addiert, um den Endpunkt zu ermitteln.

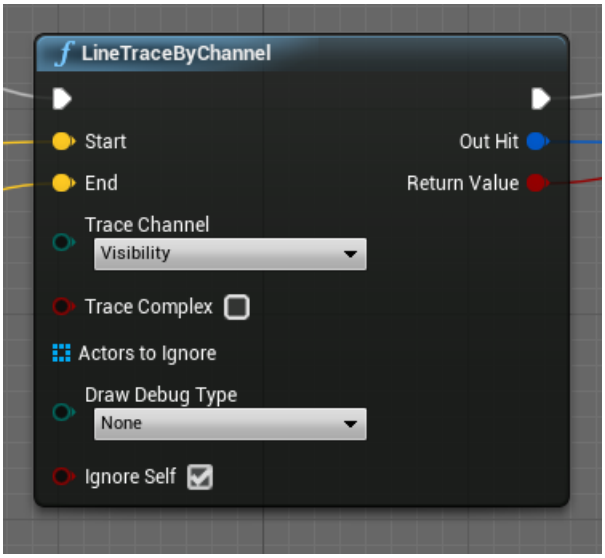

Abbildung 23: Die *Line Trace by Channel*-Funktion

Die „LineTraceByChannel"-Funktion überprüft automatisch, ob der Strahl ein Objekt getroffen hat. Sollte dies der Fall sein, wird das getroffene Objekt mit der Funktion „BreakHitResult" (siehe Abbildung 24) in verschiedene Informationen aufgeteilt. Unter anderem hat der Entwickler Zugriff auf die *Normale*, die *Position*, das *Material* und die *Klasse* des Objektes. Für die Erkennung, dass ein Objekt getroffen wurde, benötigen die Objekte eine *Kollision-Komponente*, wie zum Beispiel eine „Bounding-Box". Die Instrument-Klassen besitzen eine solche Komponente (siehe Abschnitt 6.6.2).

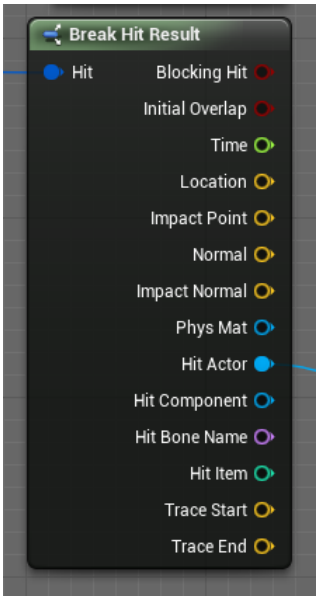

Abbildung 24: Die Aufteilung der Informationen des geschnittenen Objektes

Um sicherzustellen, dass es sich bei dem getroffenem Objekt um ein Instrument handelt, wird die „HitActor”-Variable zu einem „TextRenderActor”-Objekt umgewandelt. Wie im Abschnitt 6.6.2 beschrieben, dient ein „TextRenderActor”-Objekt als Klasse für die Instrumente. Handelt es sich nicht um ein solches Objekt ignoriert die Funktion dieses und es werden keine weiteren Schritte ausgeführt.
Wenn ein Instrument getroffen wurde, soll der Text des „TextRender”-Objektes farblich hervorgehoben werden, indem eine bläulich leuchtende Umrandung hinzugefügt wird (siehe Abbildung 25). Hierfür wird zunächst überprüft, ob das Instrument im letzten „Frame” schon ausgewählt wurde. Dafür speichert sich die *„DirigentBP”*-Klasse das Instrument, welches der Spieler zuletzt ausgewählt hatte, in der Variable *CurrentTextTarget*. Sollte das aktuelle Instrument noch nicht ausgewählt worden sein, wird die Variable *CurrentTextTarget* mit diesem Instrument-Objekt überschrieben. Die farbliche Umrandung des alten Objektes wird gelöscht und das neue Instrument-Objekt erhält eine farbliche Umrandung. Handelt es sich um das gleiche Objekt, wird keine Änderung vorgenommen.

Abbildung 25: Der Instrument-Schriftzug mit und ohne Umrandung

Die farbliche Umrandung wurde mithilfe einer „RenderCustomDepth”-Funktion (siehe Abbildung 26) der *Unreal Engine* realisiert. Diese erlaubt es, dass innerhalb eines vorher im Level platzierten „Post Processing Volumes” ein *Post-Prozess-Effekt* hinzugefügt werden kann. Dieser Effekt sollte als bläuliche Umrandung umgesetzt werden, welche als Material dem „GlobalPostProcessVolume”-Objekt hinzugefügt wurde. Wenn die Umrandung gezeichnet werden soll, muss diese mit der Funktion „SetRenderCustomDepth” aktiviert werden und kann mit dieser auch wieder deaktiviert werden.

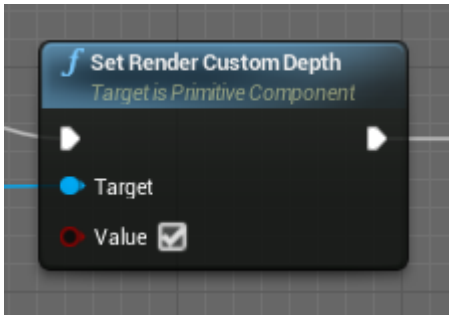

Abbildung 26: Die *Set Render Custom Depth*-Funktion

Sobald die farbliche Umrandung aktiviert wurde, wird die nächste Funktion „DetermineSound" aufgerufen und die Boolean-Variable *TargetIsSound* wird auf „true" gesetzt. Diese Variable wird von der GUI benutzt, um den Lautstärke-Balken anzuzeigen oder auszublenden (siehe Abschnitt 6.5.2).
Wenn der Spieler kein Instrument auswählt, wird die farbliche Umrandung des zuletzt ausgewählten Objektes deaktiviert und die Variable *TargetIsSound* auf „false" gesetzt.

Abbildung 27: Die *Component Has Tag*-Funktion

Die Funktion „DetermineSound" wurde in der *„DirigentBP"*-Klasse implementiert. Diese Funktion überprüft, ob das aktuelle „TextRender"-Objekt einen speziellen *„Tag"* hat. Ein *„Tag"* ist eine Text-Variable, die vom Programmierer hinzugefügt werden kann, um ein Objekt näher zu beschreiben und zu definieren. Jedem „TextRender"-Objekt wurde ein *Tag* hinzugefügt, der dem Instrument entspricht, welches die Klasse repräsentiert.
Je nachdem welchen *„Tag"* das vom Spieler ausgewählte „TextRender"-Objekt besitzt (*Violin1*, *Violin2*, *Violina*, *Cello*, *Contrabass* oder *Percussion*)(siehe Abbildung 27), wird in der entsprechenden Klasse die Funktion „PlayerGestureActive" aufgerufen, die die Geste zur Steuerung der Lautstärke aktiviert (siehe Abschnitt 6.11).

7 Evaluation

Dieses Kapitel widmet sich der Evaluation, die im Rahmen der Bachelorarbeit durchgeführt wurde. Der Benutzertest diente dazu, die erste Version des Spiels von Testpersonen spielen zu lassen und ihre Bewertung festzuhalten. Nach einer Erklärung des Fragebogens (Abschnitt 7.1.1) und der allgemeinen Testbedingungen (Abschnitt 7.1.2), unter denen getestet wurde, werden die Ergebnisse zusammengefasst (Abschnitt 7.2.1) und analysiert (Abschnitt 7.2.2).

7.1 Durchführung

In diesem Abschnitt wird der verwendete Fragebogen erläutert und wie die Evaluation durchgeführt worden ist.

7.1.1 Der Fragebogen

Der Fragebogen für die Evaluation der ersten Version des Spiels wurde in drei Teile gegliedert. Im ersten Teil wurden allgemeine Informationen und Vorkenntnisse der Teilnehmer abgefragt. Danach folgte der praktische Teil, in welchem das Spiel gespielt werden durfte, nach dem vom Entwickler eine Einleitung in die grundlegende Steuerung gegeben wurde. Die Bewertung des Spiels wurde im dritten Teil des Fragebogens von den Teilnehmern ausgefüllt.
Folgende Leitfrage sollte mithilfe des Fragebogens beantwortet werden: Welche Ansprüche sollten „Virtual Reality"-Anwendungen erfüllen, um ein immersives Erlebnis zu bieten?

Das Ziel des ersten Teils des Fragebogens war es herauszufinden, welche Erwartungen die Teilnehmer der Evaluation an eine „Virtual Reality"-Anwendung hatten und wie viel Erfahrung mit solchen Anwendungen vor dem Test bestand. In Hinblick auf die Leitfrage, sollte die festgehaltene Erwartungshaltung und Vorerfahrung dazu dienen, die Bewertung des Spiels und Verbesserungsvorschläge nachvollziehen zu können.
Die Teilnehmer wurden ebenfalls nach ihrer musikalischen Vorerfahrung befragt. Es wurde gefragt, ob ein Instrument gespielt werden würde und ob Erfahrungen mit der Arbeit eines Dirigent bestehen würden.

Die Bewertung des Spiels beschäftigte sich mit den einzelnen Komponenten, die implementiert wurden. Im Vordergrund stand die Wichtigkeit der Komponenten für die Teilnehmer beim Spielen. Mithilfe von offenen Fragen über erfüllte und nicht-erfüllte Erwartungen und die Möglichkeit Verbesserungsvorschläge zu machen, diente der letzte Teil des Fragebogens dazu herauszufinden, welche Komponenten für eine zweite Version des Spiels überarbeitet werden müssten und welche

Komponenten fehlen würden.

7.1.2 Der Evaluationstest

Der Fragebogen für die Evaluation wurde von den Teilnehmern online ausgefüllt. Hierfür wurde die Webseite *www.umfrageonline.com* verwendet. Auf dieser Webseite konnte ein Studentenkonto eingerichtet werden, mit dem es möglich war kostenlos alle Funktionen des Anbieters zu nutzen. Es können mehrere Umfragen vom Benutzer erstellt und aktiv geschaltet werden. Bei der Gestaltung des Fragebogens stehen mehrere Vorlagen zur Verfügung. Unter anderem können *Multiple-Choice Fragen*, *Bewertungsskalen*, *offene Fragen* und *Übergangstexte* und *Bilder* eingefügt werden. Auch die dynamische Veränderung der Reihenfolge der Fragen ist möglich, falls Fragen unter bestimmten Bedingungen übersprungen werden sollen. Nach Beendigung der Umfrage werden die eingegebenen Daten automatisch ausgewertet und vom System werden Diagramme erstellt.
Aufgrund dieser Funktionen wurde sich für diesen Anbieter entschieden, um den Fragebogen zu realisieren.

Im Zeitraum zwischen dem 30. November und dem 27. Dezember 2015 wurde die Evaluation durchgeführt. Von den 13 Teilnehmern haben bis auf vier Teilnehmer alle den Test im Labor der *Computergrafik* (Raum B013) in der Universität Koblenz-Landau, am Campus Koblenz, absolviert. Im Labor wurde einer der frei zugänglichen Computer genutzt. Es wurde darauf geachtet, dass der zu benutzende Computer genügend Leistung hatte, um ein flüssiges Spielen mit mindestens 30 Bildern pro Sekunde zu ermöglichen.
Zum Zeitpunkt der Tests waren auch andere Studenten der Universität im Raum anwesend. Aus diesem Grund haben die Testpersonen Kopfhörer bekommen, um nicht von äußeren Geräuschen gestört zu werden. Jeder Testperson wurde der Fragebogen auf einem separaten Laptop geöffnet, wo der Fragebogen ausgefüllt werden konnte.
Sobald der erste Teil des Fragebogens ausgefüllt worden war, wurde den Testpersonen vom Entwickler persönlich die Steuerung und das Ziel des Spiels erklärt. Es wurde betont, dass es bei der Anwendung nicht darum gehen würde, einen hohen Highscore zu erzielen, sondern dass es um den Spaß gehen würde. Das Hauptmenü wurde den Teilnehmern gezeigt und das Spiel vom Entwickler gestartet.
Während des Spielens wurden die Testpersonen vom Entwickler beobachtet, um die Reaktionen festzuhalten und in die Auswertung der Evaluation einbauen zu können (siehe Abschnitt 7.2). Nachdem das Spiel beendet wurde durfte der dritte Teil des Fragebogens ausgefüllt werden. Zu jeder Zeit waren Fragen an den Entwickler möglich, sollte es Unklarheiten gegeben haben.

Die vier Testpersonen, die nicht im Labor der *Computergrafik* getestet wurden, wurden vom Entwickler bei sich zu Hause getestet. Es wurde immer zwei Test-

personen gleichzeitig, in einem Raum, der Fragebogen vorgelegt. Nach dem ersten Teil des Fragebogens wurde beiden Testpersonen die Steuerung und das Ziel des Spiels erklärt.

Bei der Spielversion, die den Testpersonen vorgelegt wurde, hatten die Spieler die Möglichkeit das Orchester als Einheit zu dirigieren. Die einzelnen Instrumente konnten noch nicht gesteuert werden (vgl. Kapitel 6, Abschnitt 6.12).
Die Geometrie des Levels und die Besetzung des Orchesters waren bereits vorhanden (siehe Abschnitt 6.5), aber die Beschriftungen der Instrumente fehlte (vgl. Abschnitt 6.6.2). Es war möglich sich mit der rechten Hand umzusehen (siehe Abschnitt 6.9) und die Regelung der Lautstärke mit der im Abschnitt 6.11 beschriebenen Geste wurde unterstützt. Am Ende des Spiels wurde den Teilnehmern der Highscore angezeigt (siehe Abschnitt 6.7).

7.2 Ergebnisse

Der folgende Abschnitt widmet sich den Ergebnissen der durchgeführten Evaluation des, im Rahmen der Bachelor-Arbeit, programmierten Spiels. Zunächst werden die festgehaltenen Antworten der befragten Personen zusammengefasst. Danach werden die Antworten analysiert, um die Leitfrage der Evaluation zu beantworten (siehe Abschnitt 7.1.1).
Die verwendeten Diagramme wurden mithilfe von „Microsoft Excel" erstellt.

7.2.1 Beschreibung

Insgesamt wurde 13 Personen das programmierte Spiel, im Rahmen einer Evaluation, präsentiert. Die Personen waren im Alter zwischen 20 und 30 Jahren. Eine Mehrheit von *46%* der Teilnehmer war zum Zeitpunkt der Evaluation 23 Jahre alt (siehe Abbildung 28).
Unter den Teilnehmern befanden sich vier weibliche Personen. Den Hauptteil haben männliche Personen gestellt mit *69%* (siehe Abbildung 29). Die Mehrheit studierte das Fach „Computervisualistik" (*61%*) und *31%* studierten das Fach „Informatik". Dabei befanden sich fünf Personen höchstens im 5. Fachsemester und zwei Teilnehmer waren im 10. oder höheren Fachsemester. Eine Testperson hatte kein Studium, sondern eine Ausbildung abgeschlossen (siehe Abbildung 30).

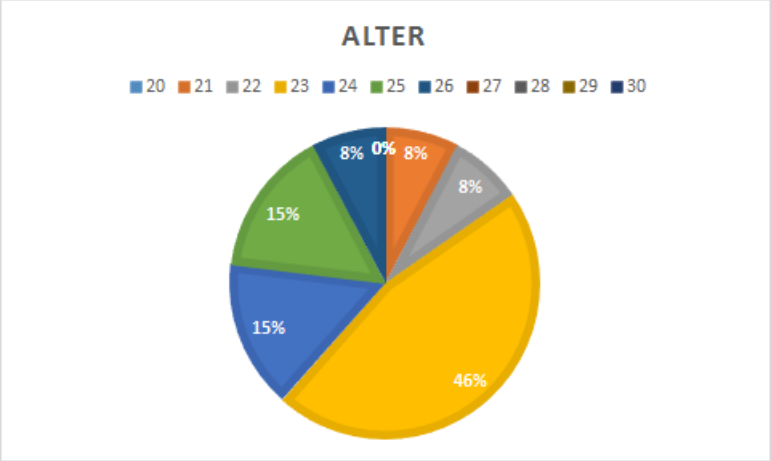

Abbildung 28: Die Altersverteilung der Teilnehmer

Alle Teilnehmer hatten vor der Evaluation bereits von dem Begriff „Virtual Reality" gehört (siehe Abbildung 32). Circa *53%* hatten bereits eine solche Anwendung genutzt (siehe Abbildung 31). Von den Teilnehmern, die bereits eine „Virtual Reality"-Anwendung verwendet hatten, wurde die „Oculus Rift" am häufigsten als Anwendung genannt. Eine Person hatte „Handtracker"-Anwendungen verwendet und eine andere Person hatte Abschlussarbeiten, wie die Vorliegende, getestet.

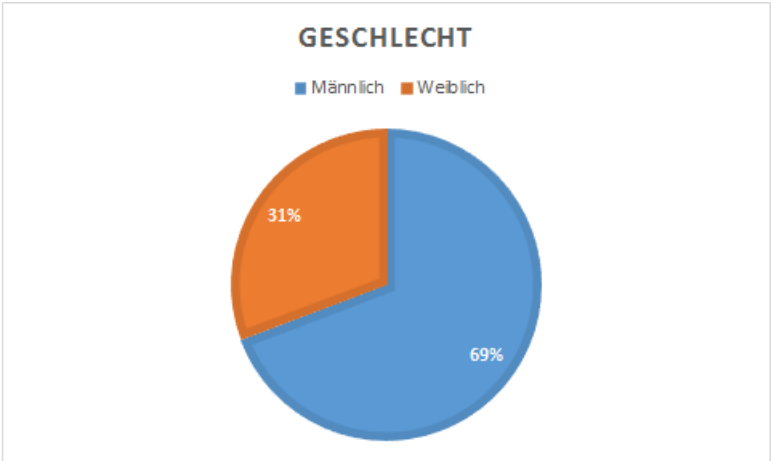

Abbildung 29: Das Geschlecht der teilnehmenden Personen

Die Teilnehmer, die bereits eine „Virtual Reality"-Anwendung ausprobiert hatten wurden gefragt, welche Aspekte ihnen an der jeweiligen Anwendung gut und welche ihnen nicht gut gefallen hätten. Am häufigsten genannt wurde die *Immersion* der Anwendung, also das Eintauchen in die virtuelle Realität. Auch der *3D-Effekt*, welcher beispielsweise bei der „Oculus Rift" vorhanden ist, habe den Teilnehmern gefallen.

Es wurden auch negative Aspekte genannt. Die auftretende „Motion-Sickness" [24] wurde bemängelt, die beispielsweise bei Anwendungen wie der „Oculus Rift" auftreten kann, da der Körper des Spielers sich nicht so bewegt, wie die virtuelle Spielfigur es tut. Schlechte Bildschirmauflösungen und Latenz wurden ebenfalls genannt. Ebenso wurde die Alltagstauglichkeit der Anwendungen negativ bewertet. Als Grund dafür wurde die fehlende Haptik und die eingeschränkte Bewegung im realen Raum angemerkt.

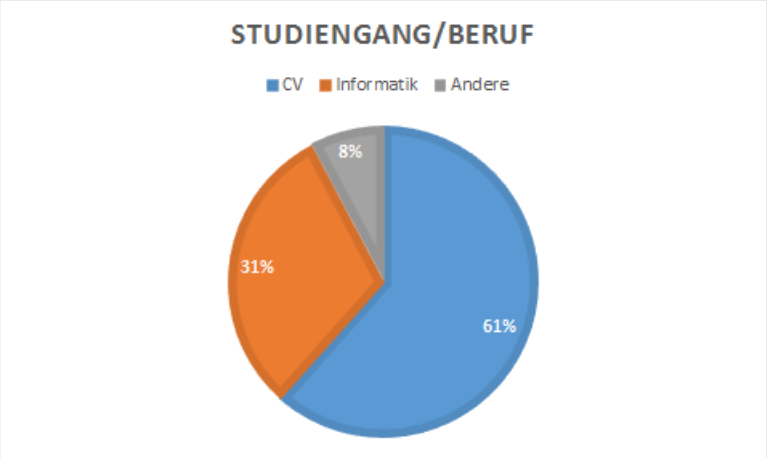

Abbildung 30: Die Ausbildung der Teilnehmer

Der Fragebogen widmete sich nun der musikalischen Vorerfahrung der Testpersonen. Insgesamt spielten vier Personen ein Instrument (*30.8%*). Sowohl *Gitarre*, *Klavier*, *Schlagzeug* und *Gesang* wurden als Musikinstrumente angegeben. Die Durchschnittszeit, die das Instrument bereits gespielt wurde, betrug circa 10 Jahre (siehe Abbildung 33).
Mit der Arbeit eines Dirigenten waren *53.8%* der Teilnehmer vertraut (siehe Abbildung 34). Unter „Vertraut mit der Arbeit eines Dirigenten" wurde verstanden, dass die Teilnehmer ein klassisches Konzert gesehen hatten, ob im Fernsehen oder bei einem Live-Konzert spielte keine Rolle.

[24] engl. für " Bewegungs-Übelkeit"

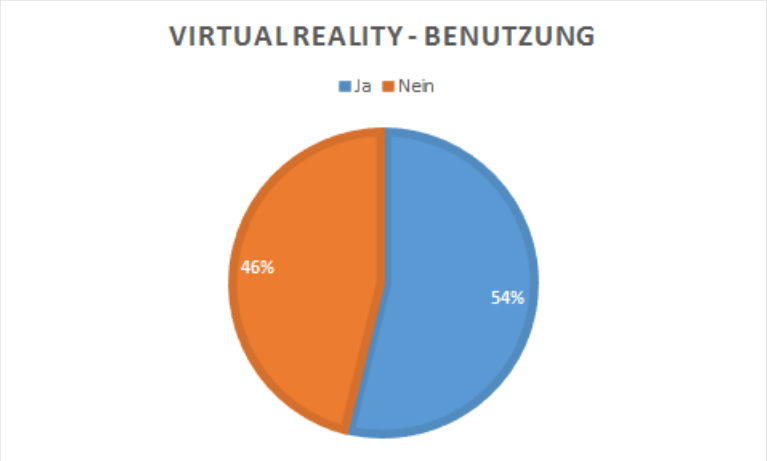

Abbildung 31: Benutzung von „Virtual Reality"-Anwendungen

Im nächsten Abschnitt des Fragebogens wurden die Teilnehmer nach ihren Erwartungen gegenüber „Virtual Reality"-Anwendungen befragt. Zuerst wurden die Erwartungen an Spiele, die „Virtual Reality"-Technik verwenden, festgehalten und danach Erwartungen, die speziell an das programmierte Spiel der Bachelorarbeit gestellt wurden.
Häufig wurde von den Teilnehmern die *Immersion* des Spiels genannt. Diese sollte möglichst überzeugend sein und keine negativen Nebeneffekte erzeugen, wie zum Beispiel *„Motion-Sickness"*, schlechte *Latenz* oder unzureichende *Übersetzung der Bewegungen* des Spielers in das Spiel. Die Integration in das Spiel wurde ebenfalls häufig genannt. Die Bewegungen des eigenen Körpers sollten im Spiel repräsentiert werden und dabei sollten keine Spielelemente die virtuelle Realität stören. Speziell wurde die graphische Benutzeroberfläche (*GUI*) genannt, welche unauffällig im Spiel integriert sein sollte. Vor allem war es den Teilnehmern wichtig, dass die reale Welt von dem Spiel vollständig ausgeblendet werden würde.

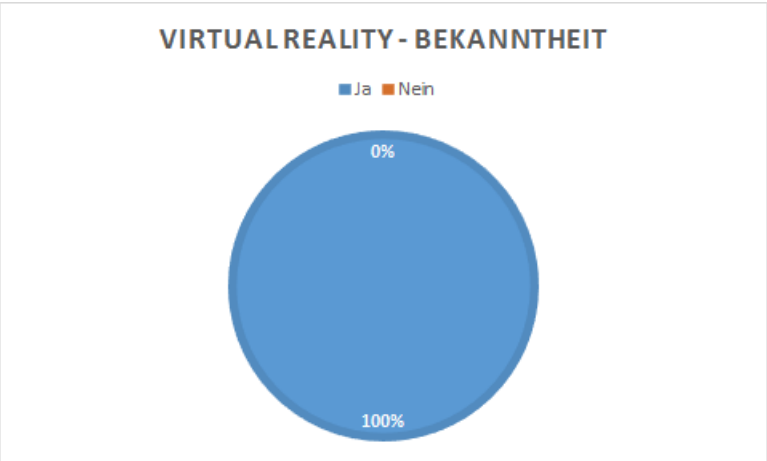

Abbildung 32: Das Bekanntheit von „Virtual Reality"

Speziell bei der musikalischen Anwendung, welche im Rahmen der Bachelor-Arbeit programmiert wurde, erwarteten die Teilnehmer, dass die Steuerung zum Dirigieren des Orchesters präzise funktionieren würde, einzelne Instrumente ausgewählt werden könnten, neben der Lautstärke auch die Geschwindigkeit der Musik beeinflusst werden könnte und das falsche Gesten eine Rückmeldung der Musik im Spiel zur Folge haben würden. Einige Teilnehmer erwarteten, dass die Arbeit eines Dirigenten näher erläutert werden und das Spiel einen musikalischen Lerneffekt mitbringen würde. Auch eine ansprechende graphische Repräsentation wurde von den Teilnehmern genannt. Beispielsweise wurde ein animiertes Orchester erwartet und ein überzeugender Konzertsaal.
Mit am häufigsten wurde von den Teilnehmern genannt, dass das zu testende Spiel Spaß machen sollte.

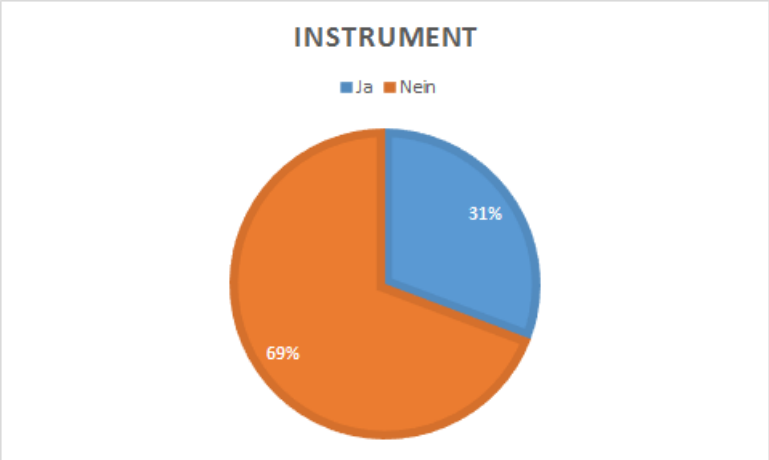

Abbildung 33: Anzahl der Teilnehmer, die ein Musikinstrument spielen

Nachdem die Teilnehmer das Spiel spielen durften, wurde festgehalten, wie wichtig die einzelnen, im Spiel vorhandenen, Komponenten den Testpersonen beim Spielen waren. Die Wichtigkeit wurde von *Sehr Wichtig*, repräsentiert durch den Wert 1, bis *Gar nicht wichtig*, repräsentiert durch den Wert 5, in fünf Schritte eingeteilt (siehe Abbildung 35).
Am wichtigsten war den Teilnehmern die Steuerung der Hände. Circa *84%* der Befragten hatten diese Komponente mit *„Sehr wichtig"* markiert und den restlichen *15%* war die Steuerung *„Eher wichtig"*. Noch vier weitere Komponenten schienen den Teilnehmern wichtig zu sein, wenn der Durchschnittswert betrachtet wird. Neben der Soundqualität wurden der Lautstärke-Balken und das virtuelle Notenblatt (siehe Kapitel 6, Abschnitt 6.5.2) als wichtigste Komponente markiert. Bei allen drei Komponenten haben circa *61%* sich für *„Sehr wichtig"* entschieden. Auch die Repräsentation des Spielers im Spiel durch Hände (siehe Kapitel 6, Abschnitt 6.8) wurde von ca. *53%* als *„Eher wichtig"* eingestuft.
Die Architektur des virtuellen Konzertsaales und das nicht vorhandene Publikum wurden von den Teilnehmern als unwichtig eingestuft mit einer Durchschnittsbewertung von 3.15 Punkten, was zwischen den Stufen *„Unentschieden"* und *„Weniger Wichtig"* lag. Unentschlossen schienen die Teilnehmer bei der Bewertung der Grafikqualität, des Highscores, des Applauses und der Repräsentation der Musiker. Bei diesen Komponenten lag der Durchschnittswert bei circa 2.5 und damit zwischen *„Eher wichtig"* und *„Unentschieden"*.

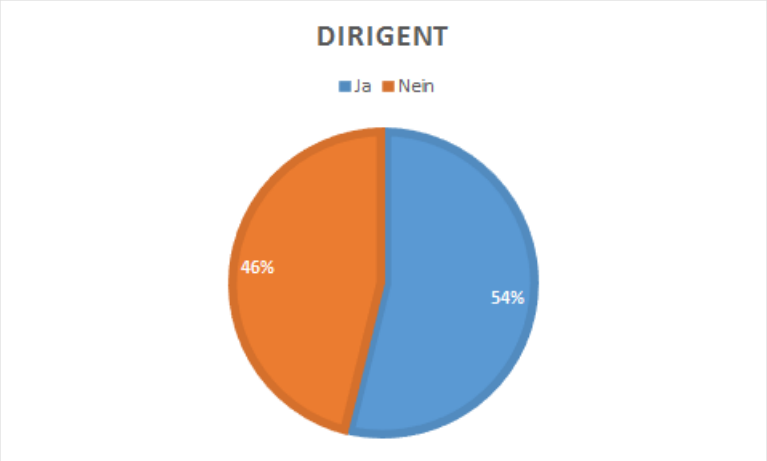

Abbildung 34: Anzahl der Teilnehmer, die mit der Arbeit eines Dirigenten vertraut sind

Die Teilnehmer hatten die Möglichkeit zu beschreiben, was ihnen am Spiel gefallen habe.
Das Umsehen innerhalb des Spiels und die Lautstärke-Regelung mithilfe der Hände wurden am häufigsten genannt. Hierbei wurde die „gute" *Latenz* der Übersetzung der Geste ins Spiel hervorgehoben und das „einfache, intuitive" erlernen und anwenden der Gesten. Sowohl der Lautstärke-Balken als auch das virtuelle Notenblatt wurden positiv erwähnt. Die Herausforderung, die Noten zu verfolgen und

den richtigen Lautstärke-Pegel einzustellen, um Punkte dafür zu bekommen, wurde ebenfalls angegeben. Auch die graphische Gestaltung des Spiels wurde an dieser Stelle genannt.

Als Antwort bei der Frage, was den Teilnehmern nicht am Spiel gefallen habe, wurde am häufigsten genannt, dass die Hände der virtuellen Spielfigur nicht der Bewegung des Spielers gefolgt sind. Diese Funktion war zu diesem Zeitpunkt noch nicht implementiert worden. Teilweise wurden die Hände auch als störend empfunden, da durch sie das Notenblatt verdeckt worden wäre.
Negativ empfanden die Teilnehmer auch Ungenauigkeiten bei der Steuerung mit den Händen. Ungenauigkeiten der *Kinect-Erfassung* führten zu ruckartigen Bewegungen, die von den Teilnehmern angemerkt wurden. Das Notenblatt war aber einer bestimmten Entfernung nicht mehr zu lesen und fehlende Notenkenntnisse haben dazu geführt, dass es einigen Teilnehmern schwer fiel dem Musikstück zu folgen.
Das statische Orchester, welches nur durch 3D-Objekte repräsentiert war, wurde von den Spielern negativ erwähnt. Einigen Teilnehmern hat es auch nicht gefallen, dass nur das ganze Orchester gesteuert werden konnte und nicht einzelne Instrumente.

	Sehr wichtig (1)		Eher wichtig (2)		Unentschieden (3)		Weniger wichtig (4)		Gar nicht wichtig (5)			
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Ø	±
Architektur des Konzertsaals	-	-	3x	23,08	6x	46,15	3x	23,08	1x	7,69	3,15	0,90
GUI - Lautstärkebalken	8x	61,54	3x	23,08	1x	7,69	-	-	1x	7,69	1,69	1,18
GUI - Notenblatt	8x	61,54	4x	30,77	1x	7,69	-	-	-	-	1,46	0,66
Grafikqualität	3x	23,08	2x	15,38	7x	53,85	1x	7,69	-	-	2,46	0,97
Highscore	2x	15,38	5x	38,46	3x	23,08	2x	15,38	1x	7,69	2,62	1,19
Publikum	1x	7,69	4x	30,77	1x	7,69	6x	46,15	1x	7,69	3,15	1,21
Reaktion des Publikums (Applaus)	1x	7,69	6x	46,15	2x	15,38	2x	15,38	2x	15,38	2,85	1,28
Repräsentation der Musiker	4x	30,77	2x	15,38	4x	30,77	2x	15,38	1x	7,69	2,54	1,33
Repräsentation des Spielers (Hä...	5x	38,46	7x	53,85	-	-	1x	7,69	-	-	1,77	0,83
Soundqualität	8x	61,54	3x	23,08	1x	7,69	1x	7,69	-	-	1,62	0,96
Steuern mit den Händen	11x	84,62	2x	15,38	-	-	-	-	-	-	1,15	0,38

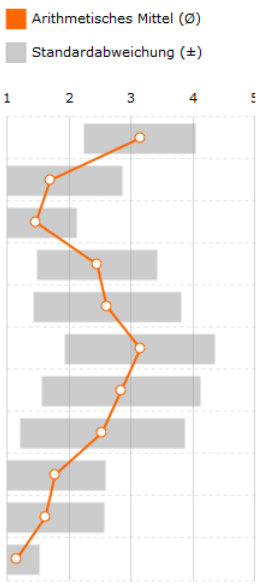

Abbildung 35: Die Bewertung der Komponenten des Spiels von den Teilnehmern

Circa *76%* der Personen, die an der Evaluation teilgenommen haben, sagten aus, dass ihre Erwartungen an das Spiel erfüllt worden sind (siehe Abbildung 36). Als Begründung wurde unter anderem die Gestensteuerung genannt, welche als „leicht" und „gut steuerbar" bezeichnet wurde. Auch der Spaß-Faktor und die Musik wurden als Grund für die erfüllten Erwartungen genannt.
Die circa *23%* der Teilnehmer, deren Erwartungen nicht erfüllt wurden, nannten

den Umfang der Steuerung als einen der Gründe. Es wäre erwartet worden, einzelne Instrumente und neben der Lautstärke auch Einsätze und Geschwindigkeit steuern zu können. Auch das nicht animierte Orchester und das generelle visuelle Feedback wurden als Antwort gegeben.

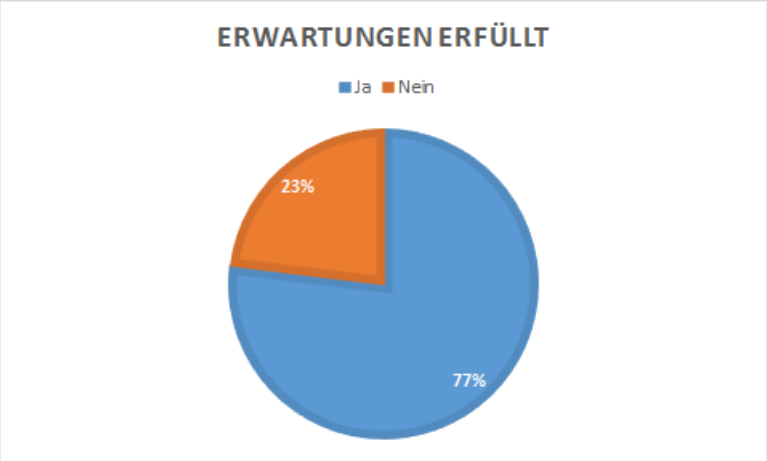

Abbildung 36: Die Anzahl der Teilnehmer, deren Erwartungen erfüllt wurden

Die letzte Frage des Fragebogens bot den Teilnehmern die Möglichkeit Verbesserungsvorschläge an den Entwickler zu geben. Es wurde gewünscht, dass die Hände der Spielfigur sich mit den Bewegungen des Spielers mit bewegen und dass die Atmosphäre des Spiels verbessert wird durch animierte Musiker und ein animiertes Publikum.
Manchen Teilnehmern fehlte die Übersicht im virtuellen Notenblatt und aus diesem Grund wurde als Verbesserungsvorschlag angemerkt, dass dem Spieler angezeigt werden sollte, wo die Musik sich im Notenblatt befindet. Auch der Vorschlag, das Spiel in Richtung einer *Karaoke-Anwendung* wie „Singstar" [25] weiterzuentwickeln, wo der Spieler versucht eine angezeigte Tonhöhe zu treffen, wurde unterbreitet.
Zudem sollten die Noten größer dargestellt werden und der Highscore sollte während des Spielens hochgezählt und nicht nur am Ende angezeigt werden.

[25] Singstar: Webseite

7.2.2 Analyse

Im Abschnitt 7.1.1 wurde die Leitfrage: „Welche Ansprüche müssen „Virtual Reality"-Anwendungen erfüllen, um ein immersives Erlebnis zu bieten?" aufgestellt. Die im Abschnitt 7.2.1 zusammengefassten Ergebnisse der Evaluation sollen nun genutzt werden, um die Leitfrage zu beantworten.

Die Mehrheit der Teilnehmer, welche an der Evaluation des programmierten Spiels teilgenommen haben, thematisierten im dritten Teil des Fragebogens (siehe Abschnitt 7.1.1) die Überführung der eigenen *Bewegungen* in das Spiel. Teilweise wurde die Steuerung der Lautstärke als „gleichmäßig" und „gut" in das Spiel integriert empfunden. Doch wurde auch angemerkt, dass vor allem bei dem *Umsehen* in der Szene Ungenauigkeiten aufgetreten wären und ungewollte Bewegungen dazu geführt hätten, dass sich der Sichtbereich der Spielfigur bewegt oder die Lautstärke der Musik sich ungewollt verändert habe.
Als Verbesserungsvorschlag wurde häufig der Wunsch geäußert, dass die Hände der Spielfigur den Bewegungen des Spielers folgen sollten und *nicht statisch* in einer Pose verbleiben. Ansonsten sollten die Hände weggelassen werden. Das Umsehen mit dem Kopf anstatt mit den Händen wurde ebenfalls genannt, da das Umsehen nicht möglich war, während die Spieler die Lautstärke der Musik verändert haben.
Diese Aussagen deckten sich auch mit den, im ersten Teil des Fragebogens, geäußerten Erwartungen an „Virtual Reality"-Anwendungen. Die Erkennung von Körperbewegungen und deren Einsatz in der Anwendung wurde häufig als Erwartung genannt. Auch eine *geringe Latenz* und eine *präzise Eingabeerkennung* wurden als Erwartungen geäußert.
Auf Grundlage dieser Anmerkungen der Teilnehmer zeigt sich, dass eine "Virtual Reality"-Anwendung den Spieler mit seinen Bewegungen im Spiel repräsentieren sollte, damit ein immersives Erlebnis möglich werden kann. Die Bewegungen des Spielers sollten genau und ohne spürbare Verzögerung in das Spiel übersetzt werden. Auch ungewollte Bewegungen durch Ungenauigkeiten der Hardware, die die Bewegungen des Spielers aufnimmt, oder in der Programmierung, sollten vermieden werden, da der Spieler dadurch den Bezug zur Spielwelt verlieren könnte. Natürliche Bewegungen, wie das Umsehen in einer Umgebung durch Drehen des Kopfes, sollten in einer „Virtual Reality"-Anwendung unterstützt werden. Fehlende Bewegungsmöglichkeiten könnten dem Benutzer auffallen und die Immersion beeinträchtigen.

Keine Testperson hat im ersten Teil des Fragebogens eine Erwartung an die *Grafik* der „Virtual Reality"-Anwendung gestellt. Im zweiten Teil des Fragebogens fiel die Grafikqualität des programmierten Spiels nur wenig auf und wurde weder als wichtig noch unwichtig empfunden. Die Qualität der szenischen Geometrie wurde positiv bewertet. Den meisten Teilnehmern fiel die *Inszenierung der Szene* dafür negativ auf. Es fehlte den Teilnehmern an *animierten Musikern* und auch ein *ani-*

miertes Publikum wäre den Antworten zur Folge wünschenswert gewesen. Diese Verbesserungsvorschläge und Aspekte, die den Testpersonen negativ aufgefallen waren, deckten sich mit den angegebenen Erwartungen. Es wurde ein realistischer Konzertsaal erwartet, in dem Musiker visuell dargestellt werden würden und der es ermöglichen würde, die echte Realität zu vergessen, um in die virtuelle Realität „eintauchen" zu können.
Als weiterer negativer Aspekt wurde die graphische Benutzeroberfläche (*GUI*) im Spiel genannt. Nach den Erwartungen der Teilnehmer sollte diese möglichst unauffällig im Spiel integriert oder gar nicht vorhanden sein. Der *Lautstärke-Balken* im programmierten Spiel wurde als störend und die *Notenanzeige* als unzureichend empfunden.
Diese Aussagen der Teilnehmer zeigen, dass die Grafikqualität des Spiels nebensächlich ist und mehr Wert auf die Inszenierung gelegt wird. Das Spiel sollte eine überzeugende Umgebung darstellen, welche zum Thema des Spiels oder des Levels passt und den Erwartungen der Spieler entspricht. Vor allem die Dynamik und Lebendigkeit einer Szene spielen eine wichtige Rolle bei der Immersion. Dabei sollten alle wichtigen Elemente im Spiel gut sichtbar und lesbar platziert sein. Wenn eine *GUI* programmiert wird, sollte diese dem Spieler unauffällig präsentiert werden, damit die Verbindung in die virtuelle Realität möglichst gut funktioniert.

Im Gegensatz zur Grafikqualität wurde von der *Soundqualität* erwartet, dass sie den visuellen Eindruck unterstützen und in guter Qualität vorhanden sein würde. Dementsprechend wurde der Sound als eine der wichtigsten Komponenten im programmierten Spiel bewertet.
Eine „Virtual Reality"-Anwendung sollte aus diesem Grund einen guten *Raumklang* simulieren, um das visuelle Erlebnis durch eine realistisch-wirkende Soundkulisse zu erweitern. Dies kann für das „Eintauchen" in die virtuelle Welt förderlich sein.

Wenn der Spieler die Möglichkeit hat mit der Spielwelt zu interagieren, sollte diese *Interaktion* intuitiv vom Spieler erlernt werden können. Die Erfahrungen und Erwartungen des Benutzers einer „Virtual Reality"-Anwendung sollten beachtet werden bei der Implementation der Interaktionsmöglichkeiten und auch die Bedingungen der realen Umgebung, in der das Spiel gespielt wird.
Es sollte vielfältige Möglichkeiten für Interaktionen in der Anwendung geben, wobei die Interaktionen nicht zu kompliziert ausfallen, aber dem Spieler genügend Freiheiten lassen sollten.
Dies lässt sich aus den Antworten der Teilnehmer schließen. Erwartet wurde eine intuitiv erlernbare Steuerung und die Möglichkeit, das Orchester den eigenen Vorstellungen entsprechend zu leiten. Je nach Erfahrungsstand der Teilnehmer wurden mehr oder weniger Interaktionsmöglichkeiten erwartet. Nicht-Musiker haben angegeben, dass die Steuerung der Lautstärke erwartet wird, während Teilnehmer, die ein Musikinstrument spielen, auch die Geschwindigkeit und Einsätze der Instrumente leiten wollten.

Dementsprechend wurde die Steuerung mit den Händen von den Teilnehmern als wichtigste Komponente im Spiel eingeschätzt. Die fehlende Steuerung der einzelnen Instrumente wurde negativ angemerkt und als Verbesserungsvorschlag wurde öfter angegeben, dass die Steuerung der Lautstärke der einzelnen Instrumente gewünscht wird. Auch die Steuerung der Geschwindigkeit und Einsätze der Instrumente wurden von den Teilnehmern als fehlende Interaktionsmöglichkeit angegeben. Manche Teilnehmer haben angegeben, dass ihnen die Rückmeldung zur Lautstärkeveränderung zu gering ausfiel.

Nicht nur die Interaktion mit einer Anwendung sollte sich an den Erfahrungsgrad des Benutzers richten, sondern auch das *Anwendungsdesign*.
Bei dem programmierten Spiel stand der Spaß an der Anwendung im Vordergrund und nicht die Vermittlung von musikalischem Wissen und dessen Anwendung. Es gab Teilnehmer die angaben, dass sie von dem programmierten Spiel erwarteten, etwas über die Arbeit eines Dirigenten zu erfahren, musikalisches Wissen vermittelt zu bekommen, wie zum Beispiel das Notenlesen und das ein musikalischer Tiefgang vorhanden sein würde. Andere Teilnehmer gaben an, dass sie vor allem erwarteten, dass die Anwendung Spaß machen würde und es nicht nötig sein würde, musikalisches Wissen bereits zu haben, um das Spiel spielen zu können.
Dementsprechend gab es bei der Bewertung des Spiels Teilnehmer, die es zu schwer fanden, den Noten auf dem virtuellen Notenblatt zu folgen und die richtige Lautstärke des Orchesters einzustellen. Es gab aber auch Teilnehmer, die die Herausforderung, die Noten auf dem Notenblatt verfolgen zu müssen, positiv bewerteten. Für die eindeutige Mehrheit gehörte das virtuelle Notenblatt zu einer der wichtigsten Komponenten im Spiel.
Eine „Virtual Reality"-Anwendung sollte den *Grad an Realismus* der Zielgruppe anpassen. Wenn eine Anwendung für keine spezielle Zielgruppe entwickelt werden soll und der Spaß im Vordergrund steht, sollten spezielle Komponenten vereinfacht oder es sollten Hilfestellungen für diese Komponenten erstellt werden. Bei dem programmierten Spiel wurde beispielsweise vorgeschlagen, dass dem Spieler eingeblendet werden sollte, wo auf dem Notenblatt die Musik sich gerade befindet.
Die Herausforderung des Spiels sollte für die Zielgruppe angemessen sein.

Ein „Virtual Reality"-Spiel, dessen Fokus auf dem Spaß liegt, sollte eine Einweisung in die notwendige Steuerung enthalten und das Ziel des Spiels deutlich hervorheben. Die spielerische Herausforderung sollte durch *visuelle Hilfsmittel* unterstützt werden, damit der Spieler die Übersicht nicht verliert.
Die Teilnehmer stellten heraus, dass die Einführung vom Entwickler im Spiel integriert hätte sein sollen. Auch Hilfestellungen für unerfahrene Spieler wurden gewünscht. Für die Herausforderung, die Lautstärke richtig einzustellen, wurde vorgeschlagen, dass ein sichtbarer Balken die Lautstärke vorgeben und die Lautstärke dementsprechend angepasst werden sollte. Die Punktzahl des Highscores hätten einige Testpersonen lieber angezeigt bekommen während des Spielens, anstatt nur am Ende des Spiels.

7.2.3 Fazit

Die Evaluation des programmierten Spiels, im Rahmen dieser Bachelor-Arbeit, hat gezeigt, welchen Erwartungen eine „Virtual Reality"-Anwendung gerecht werden muss, damit die Grundlage für ein immersives Erlebnis gelegt werden kann.
Die *Repräsentation* des Spielers im Spiel ist wichtig, um das Körpergefühl in die virtuelle Realität übertragen zu können. Spielfiguren sollten den Bewegungen des Spielers folgen und die Übertragung der Bewegung sollte ohne Latenzprobleme erfolgen. *Interaktionsmöglichkeiten* sollten nicht zu kompliziert ausfallen und der vom Benutzer erwarteten Interaktion entsprechen. Dabei sollten möglichst viele Interaktionen unterstützt werden, um dem Benutzer das Gefühl zu geben viele Freiheiten zu haben und damit die Spielwelt gestalten zu können.
Bei den Anwendungen spielt die *Grafikqualität* nur eine untergeordnete Rolle. Wichtig ist eine *realistische Inszenierung* der Welt, die den Erwartungen des Spielers entspricht und die Realität so genau wie möglich widerspiegelt. Hierbei sollte die graphische Inszenierung durch eine gute *Soundqualität* unterstützt werden, die einen realistischen Raumklang erzeugen sollte.
Je nach Zielgruppe und Art der Anwendung sollten *Hilfestellungen* gegeben und die Herausforderung in der Schwierigkeit angepasst werden. Graphische Oberflächen, die als *GUI* dienen, sollten dem Spieler nach Möglichkeit nicht auffallen, um die Immersion nicht zu stören. Dafür sollten die interaktiven Objekte in der Welt dem Spieler genügend *Rückmeldung* geben, damit keine Unklarheiten bei der Interaktion entstehen.

Wenn diese, durch die Evaluation herausgearbeiteten, Aspekte bei der Entwicklung einer „Virtual Reality"-Anwendung beachtet werden, kann eine Grundlage für eine immersive Anwendung gelegt werden, die den Spieler in eine virtuelle Realität versetzt.

8 Fazit und Aussicht

Im folgenden Kapitel wird die Entwicklung der „Virtual Reality"-Anwendung abschließend zusammengefasst. Erreichte Ziele werden aufgezeigt und bei nicht erreichten Zielen wird erläutert, woran diese Ziele gescheitert sind. Anschließend wird ein Ausblick auf die weiteren Entwicklungsmöglichkeiten der Anwendung gegeben.

8.1 Ausgangslage

Als Ausgangslage diente die Idee eine „Virtual Reality"-Anwendung zu entwickeln, welche es einem Benutzer ermöglichen sollte in die Rolle eines Dirigenten versetzt zu werden und ein Orchester steuern zu können. Die Herausforderung bestand darin, dass die Interaktion mit der Anwendung über Gesten erfolgen sollte. Dafür sollte sich in entsprechende Hardware, wie die *Microsoft Kinect*, eingearbeitet werden, um die Gestensteuerung anschließend mit der *Unreal Engine* zu verbinden.

8.2 Fazit

Die Implementation der „Virtual Reality"-Anwendung erfolgte mithilfe der *Unreal Engine* und es wurde ausschließlich die integrierte visuelle Programmiersprache, das „Blueprints Visual Scripting"-System, genutzt. Für die Gestenerkennung des Benutzers wurde sich für die *Microsoft Kinect* als Hardware entschieden (siehe Kapitel 6, Abschnitt 6.1), welche mithilfe des „Kinect4Unreal"-Plug-In mit der *Unreal Engine* verbunden wurde (siehe Kapitel 6, Abschnitt 6.3).
Insgesamt wurden drei Gesten entwickelt, welche in der abgegebenen Version des Prototypen zur Verfügung stehen. Damit wurden zwei Gesten mehr entwickelt, als die ursprüngliche Aufgabenstellung vorsah. Der Benutzer der Anwendung erhält durch die Gesten die Möglichkeit sich in der Spielwelt umzusehen, einzelne Instrumente eines Orchesters auszuwählen und die Lautstärke der Musik anzupassen. Durch das Sammeln von Punkten und einen zu erreichenden Highscore sollte die Anwendung um eine Spiel-Komponente erweitert werden, welche die Herausforderung und den Wiederspielwert positiv beeinflussen sollte. Auch ein Hauptmenü wurde programmiert, um die Anwendung in Richtung eines Videospiels zu entwickeln.
Laut Aufgabenstellung galt die graphische Repräsentation als kein Schwerpunkt der vorliegenden Arbeit (siehe Kapitel 3). Dementsprechend wurde auf ein realitätsnahes Level-Design während der Entwicklung Wert gelegt, jedoch nicht auf eine fotorealistische Grafik. Auch Animationen wurden für die Anwendung nicht erstellt. Aus zeitlichen Gründen wurde darauf verzichtet die *Oculus Rift* als zusätzliche Hardware einzubinden.

Die umgesetzten Ideen stammten aus dem, vor der Programmierung entstandenem, Konzept (siehe Kapitel 5). Von den im Konzept überlegten Komponenten des Spiels wurden bis auf vereinzelte Komponenten alle umgesetzt. So sollte zum Beispiel eine Geste dazu dienen, die Geste für die Lautstärke zu aktivieren und es sollte eine Songauswahl im Hauptmenü zur Verfügung stehen. Diese Komponenten wurden während der Entwicklung verworfen, da sie nicht gebraucht wurden.

Im Rahmen einer Evaluation wurde die erste Version des Prototyps durch mehrere Teilnehmer bewertet. Das Ziel der Evaluation war es herauszufinden, welche Anforderungen an „Virtual Reality"-Anwendungen von den Benutzern gestellt werden und in wie weit der Prototyp diese Anforderungen bereits erfüllen würde. Als Ergebnis wurde festgehalten, dass einige der Anforderungen, wie die Übersetzung der Gesten in die Anwendung, erfüllt waren und positiv bewertet wurden. Es wurden auch Verbesserungsvorschläge und unzureichende Komponenten herausgearbeitet. Ein musikalischer Lerneffekt, welcher als wünschenswert in der Aufgabenstellung der Bachelorarbeit genannt wurde, konnte bei den Testpersonen nicht erreicht werden.
Die zu beantwortende Leitfrage konnte im Rahmen der Evaluation beantwortet werden und die erfassten Anforderungen könnten einem Entwickler ähnlicher Anwendungen dazu dienen, um die Grundlage für eine „Virtual Reality"-Anwendung zu legen und Fehler von vornherein zu vermeiden.

Ein Einblick in die Arbeit des Entwicklungsprozesses von Videospielen konnte gewährt werden, durch die Entwicklung der Anwendung mithilfe der *Unreal Engine*. Darüber hinaus konnte in Erfahrung gebracht werden, welche Anforderungen speziell „Virtual Reality"-Spiele erfüllen müssen und durch die Verwendung der *Kinect* als Eingabegerät konnte eine Möglichkeit zur Umsetzung eines solchen Spiels erfahren werden. Dadurch wurde der gewünschte Lernerfolg im Rahmen dieser Arbeit erreicht.

8.3 Ausblick

Wie im vorherigen Abschnitt beschrieben (siehe 8.2), wurden einige Komponenten des Konzeptes der „Virtual Reality"-Anwendung nicht umgesetzt und im Rahmen der Evaluierung wurden unzureichende und fehlende Komponenten festgestellt. Der Umfang der Anwendung könnte erweitert werden, in dem mehr Musikstücke in das Spiel übertragen und dem Spieler zur Verfügung gestellt werden. Dies würde auch eine Songliste im Hauptmenü rechtfertigen, aus der der Spieler ein Stück aussuchen könnte. Ein Schwierigkeitsgrad fehlt bisher in dem Spiel und könnte nachträglich implementiert werden. Dieser sollte sich an den musikalischen Erfahrungswert des Spielers richten und damit mehrere Zielgruppen für die Anwendung erschließen.
Die spielerische Komponente könnte um einen Karaoke-Modus erweitert werden,

indem der Spieler gezielt versucht die Lautstärke der Instrumente einzustellen, wobei Hilfestellungen anzeigen sollten, welche Lautstärke eingestellt werden muss. Für erfahrenere Benutzer oder Benutzer, die speziell die musikalische Herausforderung in der Anwendung suchen, könnte ein separater Spielmodus programmiert werden, indem keine Hilfestellungen gegeben werden und der Spieler ausschließlich durch das Lesen der Noten die richtige Lautstärke einstellen muss. Hierfür würde auch jedes Instrument sein eigenes Notenblatt benötigen. Auch ein freier Modus, indem kein Highscore verfügbar wäre und der Spieler seine Interpretation des Musikstückes verwirklichen könnte, wäre vorstellbar.

Mehr Gesten zur Steuerung des Orchesters könnten nachträglich in die Anwendung integriert werden. Es wäre denkbar neben der Lautstärke auch die Geschwindigkeit der Musik einzustellen und die Einsätze der Instrumente zu verwalten. Dies würde auch erfordern, dass ein falsches Dirigieren des Spielers eine entsprechende Rückmeldung der Musik zur Folge hätte.
Das Umsehen im Spiel könnte durch die Anbindung der *Oculus Rift* besser integriert werden und würde keine Geste des Spielers mehr nötig machen, da die Erfassung des Kopfes von der *Oculus Rift* unterstützt wird. Dies könnte die Immersion des Spiels erhöhen und den Spieler besser in die Welt integrieren.
Die Inszenierung des Orchesters sollte um animierte Musiker erweitert werden, um einen höheren Grad an Realismus zu bieten. Auch ein animiertes Publikum könnte in die Szene integriert werden und die Hände der virtuellen Spielfigur sollten sich mit den Bewegungen des Spielers mit bewegen, um glaubhafter zu wirken.
Durch Ungenauigkeiten der *Microsoft Kinect* kam es im Prototypen noch zu ungewollten, zittrigen Bewegungen. Es könnte versucht werden diese Ungenauigkeiten zu minimieren und die Bewegungen im Spiel zu verbessern.

Literatur

[BFK+09] Johannes Behr, Torsten Froehlich, Christian Knoepfle, Wolfram Kresse, Bernd Lutz, Dirk Reiners, and Frank Schoeffel. In *The Digital Cathedral of Siena - Innovative Concepts for Interactive and Immersive Presentation of Cultural Heritage Sites*, 2009. http://www.archimuse.com/publishing/ichim01_vol1/behr.pdf.

[Fro02] Torsten Froehlich. In *Dynamisches Objektverhalten in Virtuellen Umgebungen*. TU Darmstadt, 2002. http://tuprints.ulb.tu-darmstadt.de/262/.

[Gam15a] Epic Games. *UMG UI Designer Quick Start Guide*, 2004-2015. https://docs.unrealengine.com/latest/INT/Engine/UMG/QuickStart/index.html.

[Gam15b] Epic Games. *Unreal Engine 4 Documentation*, 2004-2015. https://docs.unrealengine.com/latest/INT/index.html.

[Gei13] Christian Geiger. In *Conductor's Philosophy Mood Video*. Youtube, 2013. https://www.youtube.com/watch?v=4R2Op8uxcTs.

[Ger04] Moritz Gerl. In *Virtuelle Spruehwand*. Universitaet Koblenz, 2004. http://userpages.uni-koblenz.de/~cg/Studienarbeiten/virtuelle_spruehwand_gerl.pdf.

[Lut09] Bernd Lutz. 2009. http://www.3d-computergraphik.de/page9/page7/page7.html.

[PFSG14] Cornelius Poepel, Jochen Feitsch, Marco Strobel, and Christian Geiger. In *Design and Evaluation of a Gesture Controlled Singing Voice Installation*. nime.org, 2014. http://www.nime.org/proceedings/2014/nime2014_439.pdf.

[ton11] tonbilder. In *Synse - Visual Music App*. Youtube, 2011. https://www.youtube.com/watch?v=B3OK5Hw8Tus.

[WC09] Matt Wain and Mark Cunningham. In *A Legacy Unfolds - The British Music Experience*. tpi-magazine, 2009. http://www.tpimagazine.com/production-profiles/eventprofiles/245922/a_legacy_unfolds_the_british_music_experience.html.

Anhang

Der virtuelle Dirigent - Evaluation

Begrüßung

Vielen Dank für Ihre Teilnahme an der folgenden Evaluation. Im Rahmen meiner Bachelor-Arbeit habe ich ein "Virtual-Reality"-Spiel entwickelt, in dem Sie als Benutzer in die Rolle eines Dirigenten schlüpfen und ein Orchester dirigieren können.
Die Evaluation beginnt mit einem Theorie-Teil, danach dürfen Sie das Spiel spielen, um es anschließend zu bewerten.

Viel Spaß beim Spielen wünscht Ihnen
Marcel Bock

Allgemeine Informationen

Allgemeine Informationen

Falls Sie Interesse hätten auch am zweiten Teil der Evaluation teilzunehmen, geben Sie bitte Ihre E-Mail-Adresse an. (Der zweite Test erfolgt unter Vorbehalt)

Alter	
Geschlecht	
Studiengang	
Fachsemester	
E-Mail (optional)	

Vorkenntnisse

Haben Sie von dem Begriff "Virtual Reality" schon vorher einmal gehört? *

- ○ ja
- ○ nein

Vorkenntnisse

Haben Sie vorher schon einmal eine "Virtual Reality"-Anwendung benutzt? *

Textfeld ignorieren bitte

- ○ Ja
- ○ Nein
- ○

Vorkenntnisse

Welche Anwendung haben Sie schon ausprobiert? *

Vorkenntnisse - Erfahrung

Was hat Ihnen an der "Virtual-Reality"-Anwendung gefallen? *

Was hat Ihnen an der "Virtual-Reality"-Anwendung nicht gefallen oder gefehlt? *

Virtual Reality - Eine kurze Erklärung

Sogenannte "Virtual Reality"-Anwendungen wollen Sie in eine virtuelle Welt hinein versetzen und Ihnen ein möglichst realistisches Erlebnis bieten. Dies kann zum Beispiel eine virtuelle Achterbahnfahrt sein, ein virtueller Tauchgang oder eine virtuelle Reise in antike Städte, bei der Sie etwas über Kunst und Kultur erfahren, ohne Ihr Haus verlassen zu müssen.

Musikalische Erfahrung

Spielen Sie ein Instrument? *

- ○ ja
- ○ nein

Musikalische Erfahrung

Welches Instrument spielen Sie und wie lange spielen Sie es schon? *

Orchester Erfahrung

Haben Sie Erfahrung mit der Arbeit eines Dirigenten? *

Unter Erfahrung zählt auch z.B. der Besuch eines Konzertes mit ORchester oder eine Fernsehübertragung des selbigen. (Textfeld ignorieren bitte)

- ○ Ja
- ○ Nein
- ○

Erwartungen

Welche Erwartung(-en) haben Sie allgemein an ein "Virtual Reality"-Spiel? *

Im Rahmen der Bachelor-Arbeit wird ein "Virtual-Reality"-Spiel entwickelt, dass Sie als Spieler in die Rolle eines Dirigenten versetzt. Welche Erwartungen haben Sie ganz konkret an das Spiel? *

Worauf freuen Sie sich bzw. was stellen Sie sich unter dem Spiel vor?

Sie dürfen nun Spielen

Hiermit ist der erste Teil der Evaluation zu Ende. Sie dürfen nun das Spiel spielen und können dann mit dem zweiten Teil fortfahren. Viel Spaß beim Spielen!

Spiel-Bewertung

Wie wichtig waren Ihnen die folgenden Aspekte des Spiels? *

	Sehr wichtig	Eher wichtig	Unentschieden	Weniger wichtig	Gar nicht wichtig
Architektur des Konzertsaals	○	○	○	○	○
GUI - Lautstärkebalken	○	○	○	○	○
GUI - Notenblatt	○	○	○	○	○
Grafikqualität	○	○	○	○	○
Highscore	○	○	○	○	○
Publikum	○	○	○	○	○
Reaktion des Publikums (Applaus)	○	○	○	○	○
Repräsentation der Musiker	○	○	○	○	○
Repräsentation des Spielers (Hände)	○	○	○	○	○
Soundqualität	○	○	○	○	○
Steuern mit den Händen	○	○	○	○	○

Spiel - Positive und negative Erfahrungen

Was hat Ihnen an dem Spiel gefallen? *

Was hat Ihnen nicht am Spiel gefallen? *

Spiel - Erwartungen

Wurden Ihre Erwartungen an das Spiel erfüllt? *

- ○ ja
- ○ nein

Spiel - Erwartungen

Warum wurden Ihre Erwartungen erfüllt und/oder warum nicht? *

Spiel - Verbesserungsvorschläge

Welche Aspekte des Spiels müssten Ihrer Meinung nach verbessert werden, damit die "Virtual Reality" - Erfahrung überzeugend(-er) wird? *

» Umleitung auf Schlussseite von Umfrage Online