HOCHSCHULE MITTWEIDA
University of Applied Sciences

BACHELORARBEIT

Herr
Vincent Ritter

Virtual Reality (VR)-Tools in der raumakustischen Planungspraxis

Mittweida, Juli 2023

Fakultät **Medien**

BACHELORARBEIT

Virtual Reality (VR)-Tools in der raumakustischen Planungspraxis

Autor:
Vincent Ritter

Studiengang:
Media and Acoustical Engineering

Seminargruppe:
MG19wA-B

Erstprüfer:
Prof. Dr.-Ing. Michael Hösel

Zweitprüfer:
Dr.-Ing. Eckard Mommertz

Einreichung:
Mittweida, 09.07.2023

Verteidigung/Bewertung:
Mittweida, 21.07.2023

Faculty of **Media Sciences**

BACHELOR THESIS

Application of virtual reality (VR)-tools for the design of room acoustics

Author:
Vincent Ritter

Course of Study:
Media and Acoustical Engineering

Seminar Group:
MG19wA-B

First Examiner:
Prof. Dr.-Ing. Michael Hösel

Second Examiner:
Dr.-Ing. Eckard Mommertz

Submission:
Mittweida, 09.07.2023

Defense/Evaluation:
Mittweida, 21.07.2023

Bibliografische Beschreibung:

Ritter, Vincent:
Virtual Reality (VR)-Tools in der raumakustischen Planungspraxis. – 2023. – 45 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Fakultät Medien, Bachelorarbeit, 21.07.2023.

Abstract:

Virtual reality (VR)-headsets and 3D Sound are common techniques for gaming and consumer audio that aim to enhance immersive experiences and try to increase realism. Game engines like Unity or Unreal Engine allow the creative and technical design of VR environments and simple soundscapes. In contrast, acoustic planning aims to get the most accurate precalculations for the acoustics of non-existing rooms. Therefore, the planning of acoustics of cultural spaces like concert halls or theaters are determined using different approaches. 3D spatial software modelling tools (e.g. Rhino, Sketchup) are used for a quick and creative creation of room models to examine through simple acoustic simulations, import room models and calculate sound propagation from source to receiver position leading to an auralization.
The user interface and the offline processing complicate the desired straightforward experience of the acoustic space. This bachelor thesis therefore deals with the possibilities and limitations of existing VR techniques in context of room acoustics planning. In order to obtain the most realistic auditory impressions possible, measured room impulse responses or those calculated with a proven room acoustics simulation program are to be integrated into VR within the scope of this work. For this purpose, the listener is located at fixed positions in the virtual room.

Referat:

VR-Brillen und 3D-Sound sind mittlerweile verbreitete Techniken für Gaming und Consumer-Audio, die das Erleben intensivieren und realistischer machen sollen. Der kreativ-technische Zugriff ist unter anderen mit Game-Engines wie Unity oder der Unreal Engine möglich, die Support für VR-Bild- und einfache Audio-Renderings liefern. Im Gegensatz dazu versucht eine akustische Planung ein möglichst genaues Bild der späteren klanglichen Realität zu erhalten. Die Akustikplanung von Kulturräumen wie etwa Konzertsälen oder Theatern erfolgt deshalb üblicherweise auf anderen Wegen. Zeichen-Software für 3D-Raummodellierung (Bsp. Rhino, Sketchup) wird für die schnelle und kreative Erstellung der Räume gebraucht und einfache akustische Untersuchungen mit Plug-Ins durchgeführt. Die Ansicht erfolgt am Bildschirm. Spezialisierte Akustik-Simulationsprogramme importieren die Raummodelle und berechnen die Schallübertragung von Sendeposition zu Empangsposition bis hin zur Auralisation.
Die Programm-Schnittstellen und das Offline-Processing erschweren das eigentlich gewünschte, unkomplizierte Erleben des Raumes. Die Bachelorarbeit zeigt die Möglichkeiten und Grenzen von bestehender VR-Techniken im Kontext der Raumakustikplanung. Um möglichst realistische Höreindrücke zu erhalten, sollen im Rahmen dieser Arbeit gemessene bzw. mit einem bewährten raumakustischen Simulationsprogramm berechnete Raumimpulsantworten in die VR integriert werden. Der Hörer befindet sich dafür auf fest definierten Positionen des virtuellen Raumes.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3D ..... Dreidimensional

ACN ..... Ambisonics Channel Numbering

Ak ..... Audiokinetic

AR ..... Augmented Reality

AUX ..... Auxilary

BSO ..... *Müller-BBM* Building Solutions

CAD ..... Computer-Aided Design

DAW ..... Digitale Audioworkstation

ESM ..... Early Scattering Methode

FOA ..... First Order Ambisonics

FuMa ..... Furse-Malham

GmbH ..... Gesellschaft mit beschränkter Haftung

HOA ..... Higher Order Ambisonics

HRIR ..... Head-Related Impulse Response

HRTF ..... Head-Related Transfer Function

ILD ..... Interaural Level Difference

ISM ..... Image Source Methode

ITD ..... Interaural Time Difference

LZI ..... Linear-Zeitinvariant

POS ..... Position

RRM ..... Ray-Radiosity Methode

RT ..... Ray-Tracing

RWTH ..... Rheinisch-Westfälische Technische Hochschule *Aachen*

SDK ..... Software Development Kit

UI ..... User Interface

URL ..... Uniform Resource Locator

USB ..... Universal Serial Bus

VR ..... Virtual Reality

VST ..... Virtual Studio Technology

WLAN . . . . . .Wireless Local Area Network

Danksagung

An dieser Stelle möchte ich mich bei der Firma Müller-BBM BSO GmbH und besonders bei meinem Betreuer Dr.-Ing. Eckard Mommertz für die fachliche Unterstützung bedanken. Während der Entwicklung des im Folgenden beschriebenen Modellaufbaus wurde ich durch Herrn Dipl.-Ing Elias Hoffbauer weiter unterstützt und bin ihm für seinen umfangreichen Einsatz im Verlauf der Arbeit zu Dank verpflichtet.

1 Einleitung

Die rasante Entwicklung von Virtual Reality (VR)-Technologien der letzten Jahre schafft neue Möglichkeiten für immersive Erlebnisse, sowie eine praktische Handhabung der zugehörigen Hardware für Konsumenten. Das von Apple im Juni 2023 präsentierte VR-Headset *Vision Pro* zeigt wieder einmal Fortschritte der Integration von VR-Inhalten für die Anwendung in klassischen Arbeitsszenarien sowie auch für Unterhaltungsmedien [1]. Inhalte werden nicht mehr in der völlig einhüllenden VR dargestellt, die Einbindung der Augmented Reality (AR) setzt neue Maßstäbe für die Darstellung von Anwendungen durch eine Mischung mit dem visuellen Umfeld für den täglichen Arbeitsalltag. Die Einbindung audiovisueller Inhalte fordern den Hör- und Sehsinn des Menschen und emulieren virtuelle Räumlichkeiten, wie bereits durch VR-Spiele bekannt. Die erlebte Immersion lässt sich jedoch ebenso für weniger unterhaltende, sondern auch für planerische Anwendungen einsetzen.

Die Raumakustik wird in mehrere Fachbereiche unterteilt und beschäftigt sich mit der Planung akustischer Gegebenheiten von Räumen. Jeder Fachbereich berechnet anhand unterschiedlicher mathematischer Grundlagen und Modelle Kenngrößen des zu planenden Raumes. Die statistische Raumakustik befasst sich beispielsweise mit der Prognose des entstehenden Nachhalls und der zugrundeliegenden Schallenergieverteilung auf Grundlage statistischer Methoden. Die detaillierte Betrachtung der Schallübertragung und den auftretenden Reflexionen fällt in das Einsatzgebiet der geometrischen Raumakustik [2]. Simulationsprogramme integrieren die nötige Funktionalität für eine Berechnung von den notwendigen Methoden und Verfahren. Eine Simulation basiert meist auf einem Raummodell, anhand dessen akustische Untersuchungen in Form von Methoden basierend auf der Strahlenverfolgung durchgeführt werden. Nach der Untersuchung der Schallverteilung im Raum können Parameter wie Nachhallzeit und Energiekurven ausgegeben werden. Die Auswertung dieser Simulationsdaten erfolgt meist jedoch über die Betrachtung von Graphen und Zahlenwerten. Ein hörbarer akustischer Eindruck der Simulation kann meist zwar durch eine integrierte Auralisierungs-Funktion durchgeführt werden, diese bietet aber wenig Flexibilität und keinen visuellen Bezug zur Räumlichkeit.

Diese Arbeit beschäftigt sich deshalb mit der Implementierung von akustischen Simulations- und Messdaten in eine VR Umgebung. Dem Nutzenden soll durch ein VR-Headset die Möglichkeit gegeben werden, einen Raum auditiv und visuell wahrzunehmen. Das Ziel ist es deshalb, einen Workflow für die Integration von akustischen Simulations- und Messdaten in einer VR-Entwicklungsumgebung aufzuzeigen. Damit soll die Fragestellung beantwortet werden, wie eine Möglichkeit der Verknüpfung von VR und Akustiksimulation aufzubauen ist. Durch die Nutzung von entsprechender Software und Hardware kann eine Annäherung an den realen Raum in der VR gestaltet werden. Der Fokus des zu erstellenden Modellaufbaus liegt auf der Verknüpfung von Software- und Hardwarekomponenten sowie einer einfachen Nutzung für den Anwender. Der dafür ergründete Modellaufbau wird in dem Methodikteil der Arbeit basierend auf empirischen Erkenntnissen und Leitquellen beschrieben.

Letztendlich soll die Arbeit dazu beitragen, die Implementierung der Mess- und Simulations-

daten nachvollziehen zu können. Dies trägt dazu bei das Verständnis für die Bedeutung einer akustischen Simulation in VR-Erlebnissen zu erweitern. So soll eine andere Perspektive für das Erleben der Raumakustik aufgezeigt werden.

2 Theoretischer Teil

2.1 Simulation eines Schallfelds

Die Raumakustik befasst sich in den letzten 100 Jahren überwiegend mit der akustischen Einschätzung von Räumlichkeiten anhand berechneter Parameter, beruhend auf empirisch ermittelten Verfahren. Durch die Simulation akustischer Gegebenheiten in Räumen werden Parameter wie Nachhallzeit und Energiekurven bestimmt. Die Simulationsergebnisse werden derzeit mit farbigen Graphen dargestellt. Die Lautstärkeskala wird beispielsweise ähnlich wie die Temperatur mit einem Verlauf von blau nach rot gekennzeichnet. Eine Auswertung der simulierten Daten benötigt einen Akustik-Ingenieur, der anhand seiner Erfahrungswerte die Parameter entsprechend interpretiert und auch Grenzen der Simulation kennt [3].
Die Möglichkeit einer Schallfeldsynthese ist aufgrund der benötigten Rechenleistung lange keine Option gewesen. Für diese wird eine Impulsantwort des Schallfelds benötigt. Die für eine Simulation nötigen physikalischen Hintergründe zur Berechnung einer Impulsantwort sind ebenfalls gegeben. Programme wie Odeon, CATT oder Ease können die Schallübertragung von einem Sender zu einem Empfänger in Form einer Impulsantwort berechnen. Dabei kann die Richtcharakteristik von Quellen und von Empfängern berücksichtigt werden [4, 5].

2.1.1 Impulsantworten

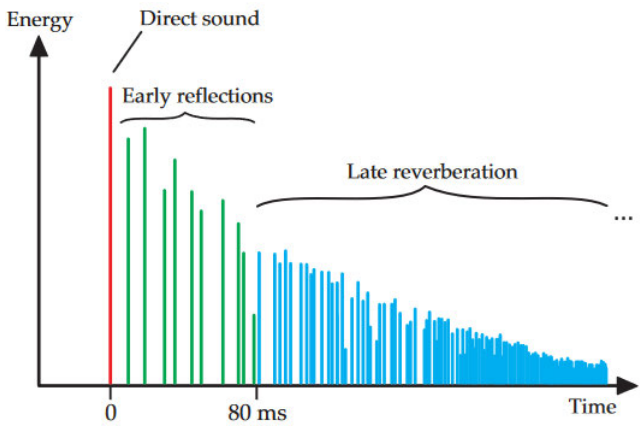


Abbildung 2.1: Schematische Energie-Impulsantwort (aus [3]).

Betrachtet man die Energieverteilung wie in Abbildung 2.1 dargestellt, sind verschiedene charakteristische Zeitbereiche erkennbar. Der erste Energieanteil, der den Empfänger erreicht, wird als Direktschall bezeichnet. Dieser Anteil des Impulses beschreibt den direkten Weg zwischen einer Schallquelle sowie einem Empfänger und tritt in den ersten 5ms der Schallenergieverteilung auf [6]. Die bis zu einer Verzögerung von $80\,ms$ eintreffenden Reflexionen werden frühe Reflexionen (engl. early reflections) genannt und tragen maßgeblich zu dem subjektiven Höreindruck bei. Anschließend treffen in der Zeitspanne von $80\,ms$ bis zum Ende des Betrachtungsfensters die späten Reflexionen (engl. late reflections) am Empfänger ein. Dieser Abschnitt beinhaltet weitere Impulsanteile mit mehrfachen Reflexionen an den Raumgrenzen, die die benannte Verzögerung am Empfänger aufweisen. Die zugehörige Reflexionsordnung einer Schallwelle gibt die Häufigkeit des Kontakts mit den Raumgrenzen an [3, 7, S.192 f.].

2.1.2 Theoretische Darstellungsmöglichkeiten raumakustischer Eigenschaften

Für die theoretische Beschreibung der akustischen Beschaffenheit einer Räumlichkeit bietet sich die Kombination der geometrischen sowie der wellentheoretischen Akustik an. Die wellenphysikalische Betrachtung von Raummoden und die Strahlenverfolgung der geometrischen Akustik sind in der Lage, den hörbaren Frequenzbereich mithilfe entsprechender Methoden und Betrachtungen anzunähern. Voraussetzung für die Ermittlung der akustischen Eigenschaften ist ein Raum mit endlichen Raumflächen sowie zugehörige bekannte Abmessungen mit gegebenen Absorptionskoeffizienten [3].

2.1.2.1 Geometrische Methoden der Akustik

Die geometrische Betrachtung der Schallausbreitung im Zeitbereich setzt eine omnidirektionale Schallquelle und einen Schallempfänger in dem zu simulierenden Raum voraus. Die Schallquelle sendet einen Delta-Impuls in Form einer Kugelwelle aus. Der Empfänger nimmt den Direktschall des Impulses, sowie die Reflexionen im Raum auf [3]. Dieses Verfahren ist aus der Systemtheorie bekannt, und wird in Abschnitt 2.1.4.1 weiter für die Nutzung in Räumlichkeiten erläutert. Der ausgesendete Impuls in Form von Schallenergie reflektiert an den Grenzflächen des Raumes, wodurch Reflexionen gewisser Ordnung entstehen. An Oberflächen des Raumes wird die reflektierte Schallwelle mit einem Ausfallswinkel zurückgeworfen, der einer Spiegelung des Einfallswinkels an der Oberflächennormalen entspricht [8, S.148 f.]. Zusätzlich tritt eine Schallstreuung an rauen und ungleichmäßigen Raumoberflächen auf [2, S.11]. Die Verteilung kann durch die Lambert'sche Reflexion angenähert werden. Diese beschreibt die eintretende Energieverteilung der eintreffenden Schallenergie in Abhängigkeit des Einfallswinkels und das dadurch entstehende diffuse Schallfeld [3, 9, S.441]. Analysiert man die erhaltenen Energieanteile am Empfänger im Zeitbereich, ergibt sich ein Histogramm der im Raum reflektierten Schallenergieanteile, auch als Impulsantwort $h(t)$ bekannt [3, 10, S.103]. Ein Zusammenhang in den Frequenzbereich $\overline{H}(f)$ ist über die Fouriertransformation gegeben [10, S.106 f.].

$$F\{h(t)\} = \overline{H}(f) \tag{2.1}$$

Aus dem resultierenden Energie-Zeitdiagramm $h_2(t)$ kann die quadrierte Impulsantwort errechnet werden, die wiederum durch die Berechnung der Schroeder-Rückwärts Integration die Abklingkurve des Raumes ergibt [4, 11]. Die Grafik 2.2 verdeutlicht nochmals die Entstehung der Schallenergieverteilung im Zeitbereich.

Für die Anwendung der beschriebenen Methodik in einem Simulationsumfeld gibt es mehrere Modelle der Berechnung zu der eintretenden Schallverteilung im Raum. Das Image-Source Verfahren wird unter Abschnitt 2.1.2.2 weiter ausgeführt und bietet sich für die Berechnung der Reflexionen in den ersten 80 Millisekunden an. Die Ermittlung der späten Reflexionen kann durch optimierte Ray-Tracing Verfahren erfolgen. Da die Berechnung einer simulierten Impulsantwort je nach Programm variiert, beschreibt Abschnitt 2.1.3 einen möglichen Simulationsablauf. Image Source- sowie Ray-Tracing Verfahren zählen zu den geometrischen Methoden der Simulation akustischer Gegebenheiten. Diese Form der Betrachtung raumakustischer Eigenschaften schließt die Betrachtung des modalen Raumverhaltens aus [3].

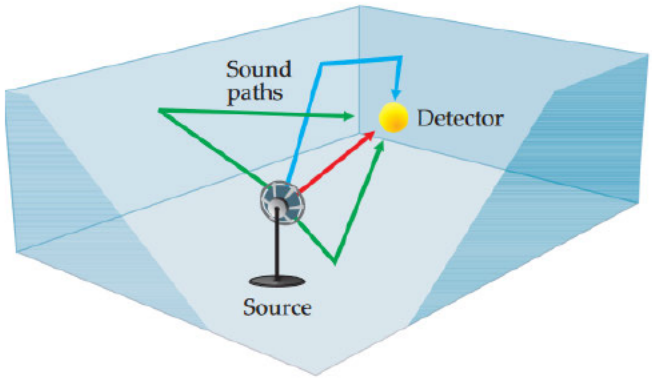


Abbildung 2.2: Raummodell mit Quelle und Empfänger.

2.1.2.2 Image Source Method nach Allen und Berkley

Das Spiegelquellenverfahren (engl. Image-Source-Method (ISM)) berechnet anhand von Spiegelquellen die Reflexionen einer Schallquelle auf den umliegendem Oberflächen eines zugehörigen Raumes [2, S.21-25, 12]. Die Grundidee des Verfahrens beruht auf den Eigenschaften der ebenen Spiegelung von Schallstrahlreflexionen an einer Oberfläche. Eine kugelförmig strahlende Schallquelle S befindet sich in einem rechteckigen Raum mit glatten Wandoberflächen. Betrachtet man die ausgesendeten Schallstrahlen von S, bilden diese orthogonal zur individuellen Ausbreitungsrichtung eine Wellenfront in Form einer Kugel um die Schallquelle. Trifft diese Wellenfront auf eine der Raumoberflächen, lässt sich die Reflexion aller Schallstrahlen durch eine ebene Spiegelung darstellen. Darunter versteht man die Spiegelung des Einfallswinkels an einer Lotsenkrechten zur Oberfläche, woraus der Ausfallswinkel resultiert. Der Schnittpunkt aller weitergeführten Schallstrahlen der Reflexion bildet die Spiegelquelle der ursprünglichen Schallquelle [2, S.21-25]. Vereinfacht kann die Spiegelquellenbildung anhand von einzelner Schallstrahlen wie in der nachfolgenden Grafik nachvollzogen werden.

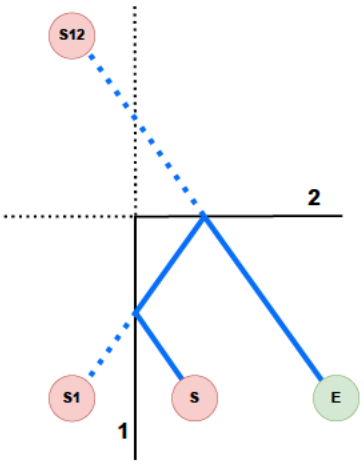


Abbildung 2.3: Betrachtung eines Schallstrahls mit Berücksichtigung der Spiegelquellen S1 und S12.

Für jede Oberfläche lässt sich so durch die Spiegelquellenmethode eine Spiegelquelle zu einer zugehörigen Schallquelle an den Grenzflächen des Raumes bilden. Betrachtet man nun den Weg des Schalls zwischen Sender und Empfänger, kann für jede Oberflächenberührung eine eindeutige Spiegelquelle identifiziert werden. Bei Reflexionen höherer Ordnung wird die Spiegelschallquelle

S1 der Oberfläche **1** an der berührten Grenzfläche **2** gespiegelt und der zurückgelegte Weg dadurch aufaddiert, siehe Grafik 2.3. Summiert man alle Spiegelquellen aller Schallwege, erhält man die Gleichung

$$h(t) = \sum_{u=0}^{1} \sum_{v=-\infty}^{\infty} A(u,v) * \delta(t - \tau(u,v)) \tag{2.2}$$

mit dem Amplitudenvorfaktor A, dem Impuls δ sowie der Zeitverschiebung τ zur Darstellung der Impulsantwort. Für die Indizierung der Ordnung bzw. der Oberfläche werden die Vektoren u, v genutzt. Bei jeder Reflexion verliert die Schallwelle an Energie, weshalb nur ein Bruchteil der ursprünglichen Schalleistung den Empfänger erreicht. Dies ist auf die schallstreuenden- und absorbierenden Eigenschaften von Oberflächen zurückzuführen. Deshalb wird β eingeführt, der Schallreflektionskoeffizient mit dem Zusammenhang $\alpha = 1 - \beta^2$ zu α, dem Absorptionskoeffizienten [12]. Da ein Material unterschiedlich stark eintreffende Schallwellen reflektiert, gibt α die Schallenergieabnahme in Form eines Faktors mit dem Maximum 1 an. Der Reflexionskoeffizient β beschreibt hingegen den Schallenergieanteil, der an einer Materialoberfläche reflektiert wird. Der Koeffizient kann ebenfalls Werte von 0 bis 1 annehmen und drückt durch sein Maximum die vollständige Reflexion der eintreffenden Schallenergie aus [13]. Die Abnahme der Schallenergie wird durch die Spiegelquellenmethode in Form des Amplitudenvorfaktors A ausgedrückt. Der Delta-Impuls wird mit Gewichtung von A multipliziert und um τ zeitlich verschoben. Die entstehende Impulsantwort entspricht einem Histogramm aller gewichteten Delta-Impulse im zeitlichen Kontext [14, 12]. Die Methodik der geometrischen Akustik findet Einsatz bei deterministischen Berechnungen der Schallwegführung sowie für die Betrachtung von Energiekurven (engl. Energy-Decay-Curves) [12].

2.1.2.3 Wellentheoretische Untersuchung eines Raumes

Die wellentheoretische Akustik beschäftigt sich mit der Modellierung von Schallwellen über das Aufstellen mathematischer Wellengleichungen. Durch eine Umformung erhält man die Helmholtzgleichung 2.3 zur Untersuchung des Verhaltens von Moden in einem rechteckigen Raum. Durch diese Gleichung lassen sich die Frequenzen der Moden eines Raumes berechnen. Der Raum entspricht der Form eines Quaders mit der Länge L, der Breite B und der Höhe H. Die Umformung der Helmholtzgleichung ergibt

$$f_{lmn} = \frac{c}{2}\sqrt{\left(\frac{l}{L}\right)^2 + \left(\frac{m}{B}\right)^2 + \left(\frac{n}{H}\right)^2} \tag{2.3}$$

in der für c die Schallgeschwindigkeit einzusetzen ist. Die Moden werden nach ihrer jeweiligen Ordnung durch l, m und n charakterisiert und entsprechend nach Schema eingesetzt. Durch die Berechnung von f_{lmn} erhält man die Frequenz einer Mode der jeweiligen Ordnung. Auf diese Weise lässt sich eine Räumlichkeit im tieffrequenten Bereich untersuchen, und das modale Verhalten eines Raumes berechnen [3, 15]. Die Grenzfrequenz des Wirkungsbereichs für die wellentheoretische Betrachtung wird durch die Schroederfrequenz festgelegt, siehe Abschnitt 2.1.2.4 [3].

2.1.2.4 Die Schroederfrequenz

Die Dichte an Moden nimmt bei steigender Frequenz um f^2 zu. Dies resultiert in einer größer werdenden Modendichte in höheren Frequenzbereichen. Dadurch ergeben sich bei Betrachtung des Frequenzspektrums zwei zu trennende Bereiche. Der tiefer liegende Bereich unterhalb der Schroederfrequenz lässt eine klare Charakterisierung der Modenbildung zu. Der Frequenzbereich oberhalb der Schroederfrequenz besitzt eine zu hohe Modendichte, sodass einzelne Moden das Schallfeld nicht mehr dominieren und kann akustisch durch statistische Methoden beschrieben werden [3, 16, S.148 f.]. Die Schroederfrequenz f_S gibt die Grenzfrequenz zwischen den beiden Bereichen an und wird durch das Raumvolumen V und die Nachhallzeit T des Raumes gebildet.

$$f_s = 2000\sqrt{\frac{T}{V}} \tag{2.4}$$

Unter dieser Grenzfrequenz ist das modale Verhalten des Raumes nicht mehr vernachlässigbar und für eine Annäherung der akustischen Eigenschaften eines Raumes zu berücksichtigen [15, 16, S.148]. Durch den Zusammenhang von Raumvolumen und Nachhallzeit zu der Schroederfrequenz wird erkennbar, dass bei größer werdendem Raumvolumen die Schroederfrequenz kleiner wird. Für den Konzertsaal Blaibach, der in dem folgenden Versuchsaufbau eine größere Rolle spielen wird, ergibt sich für sein Raumvolumen von $1360\,m^3$ eine Schroederfrequenz von $64\,Hz$. Durch die tiefe f_s kann eine wellentheoretische Betrachtung für diesen Raum vorerst ausbleiben.

2.1.3 Akustische Simulation und Auralisation einer Impulsantwort

Die in Kapitel 2.1.2 erläuterten Methoden zur Gewinnung von akustischen Simulationsdaten sind bereits in Programme wie Odeon oder CATT integriert [4, 5]. Anhand von Odeon wird im Folgenden ein Simulationsvorgang beschrieben. Das Programm integriert Berechnungsmethoden der geometrischen Akustik. Odeon nutzt hierfür die Spiegelquellenmethode, die Early-Scattering Methode (ESM), das Ray-Tracing (RT) sowie die Ray-Radiosity Methode (RRM). Die verschiedenen Verfahren werden für die in Abschnitt 2.1.2.1 beschriebenen, zeitlich unterschiedlich gelegenen markanten Bereiche der zu berechnenden Schallenergieverteilung angewandt [17, S.133 ff. 4].

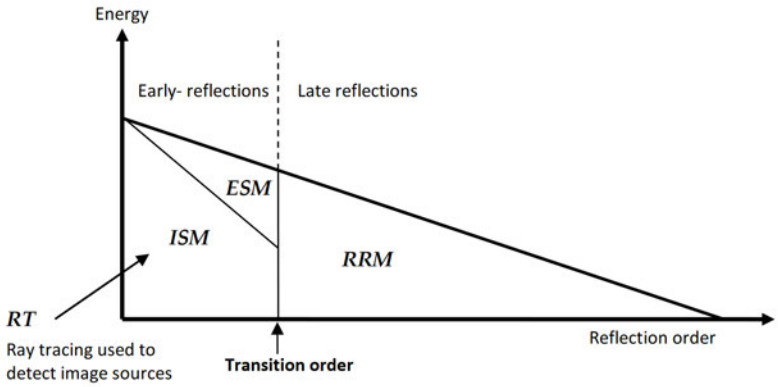


Abbildung 2.4: Aufbau des Berechnungsmodells von Odeon (aus [17, S.134]).

Für die akustische Simulation unterteilt Odeon die zu simulierende Impulsantwort, wie in Abbildung 2.4 ersichtlich, in die Bereiche der frühen Reflexionen sowie in die späten Reflexionen. Den Übergang dieser Bereiche bildet die Transition Order, zu deutsch Übergangsordnung. Diese gibt außerdem an bis zu welcher Ordnung Reflexionen in den Bereich der frühen Reflexionen fallen und entsprechend berechnet werden. Die Berechnung wird über die Spiegelquellenmethode aus Abschnitt 2.1.2.2 und durch die ESM realisiert. Für die Erstellung von Spiegelquellen wird ein RT-Verfahren genutzt. Dieses erstellt bei dem Erreichen einer Grenzfläche des Raumes durch einen Schallstrahl eine neue Spiegelquelle bis die eingestellte Übergangsordnung erreicht ist. Zusätzlich überprüft Odeon die Visibilität einer Spiegelquelle aus der Empfängerposition. Dadurch können Spiegelquellen, die nicht zur Generierung der frühen Reflexionen beitragen, ausgeschlossen werden. Die Simulation der Schallstreuung wird über die ESM mit in die Simulation einbezogen. Für jede Spiegelquelle einer Oberfläche wird eine sogenannte Early Secondary Source erstellt. Diese emittiert gestreute Schallstrahlen nach der Vorgabe durch Simulationsparameter. Diese Schallstrahlen werden bis zur Übergangsordnung verfolgt und erstellen auf berührten Oberflächen ebenfalls wieder Early Secondary Sources. Berücksichtigt werden durch die Berechnung dabei jedoch nur sichtbare Spiegelquellen der Empfängerposition [17, S.134-136].
Bei Erreichen der Übergangsordnung übernimmt die RRM die Berechnung des späten Nachhalls. Dieses Verfahren optimiert das RT-Verfahren und führt in zwei Stufen eine akustische Raumuntersuchung durch. Der Renderingprozess ermittelt empfängerunabhängig sogenannte Late Secondary Sources durch eine Strahlenverfolgung von späten Strahlen (engl. late rays). Diese werden nach Vorgaben der Simulationsparameter bis zu einer gewissen Ordnung berechnet. Die resultierenden späten Late Secondary Sources emittieren Schall je nach Simulationseinstellung beispielsweise mit Berücksichtigung der Lambert'schen Streuung. In einem zweiten Schritt werden die für einen Empfänger beitragenden Late Secondary Sources in einer Impulsantwort zusammengefasst [17, S.136-139].
Die Zusammenführung der simulierten zeitabhängigen Schallenergieanteile der frühen und späten Reflexionen ergibt am Ende eine Impulsantwort des Raumes. Odeon implementiert zusätzlich Variablen zur Steuerung der Simulation und berücksichtigt weitere Faktoren wie die Lambert'sche Streuung der Schallenergie in unterschiedlichen Formen sowie andere physikalische Eigenschaften der Räumlichkeit. Die Methodik der Schallstreuung wird beispielsweise weiter durch den Streuungs- und Absorptionskoeffizienten von Materialien beeinflusst. Dadurch wird eine materialbedingte Streuung sowie die Absorption von Schallenergie durch die Simulation berücksichtigt. Algorithmen berechnen zusätzlich die winkelabhängige Schallabsorption von Materialien.
Odeon bietet außerdem die Möglichkeit anhand der berechneten Impulsantwort eine Auralisation durchzuführen. Hierfür werden Aufnahmen aus einem schalltoten Raum vorerst mit der errechneten Impulsantwort gefaltet. Anschließend kann eine Binauralisierung des Signals durchgeführt werden, was eine Faltung mit einer HRTF, siehe Abschnitt 2.3 für das linke und rechte Ohr notwendig macht [17, S.87 f.]. Der Prozess der Faltung wird im anschließenden Abschnitt weiter erläutert.

2.1.4 Die Faltung von Impulsantworten

Der Faltungsprozess ist ein aus der Systemtheorie bekannter Vorgang und wird üblicherweise anhand eines linearen zeitinvarianten (LZI)-Systems beschrieben. Im akustischen Kontext wird eine Räumlichkeit als ein lineares und zeitinvariant wirkendes System betrachtet. Analog zu der

Betrachtung der Systemtheorie, wird durch das LZI-System in Form eines Raumes ein Prozess in Form einer Übertragungsfunktion ausgeführt, der das Ausgangssignal $Y(s)$ in Beziehung zu dem Eingangssignal $X(s)$ setzt [16, S.540 ff. 18, S.162 f.]. Durch $H(s)$ wird diese Systemfunktion bezeichnet. Daraus ergibt sich der Zusammenhang

$$Y(s) = H(s)X(s) \tag{2.5}$$

für die mathematische Darstellung eines LZI-Systems. $H(s)$ wird für den akustischen Kontext in den Zeitbereich transformiert und als Raumimpulsantwort bezeichnet. Durch ein Faltungsintegral kann der anzuwendende Prozess eines LZI-Systems auf ein Signal berechnet werden. Wie in Gleichung 2.5 durch eine Multiplikation das Ausgangssignal $Y(s)$ berechnet wird, erfolgt dies im Zeitbereich durch eine Faltung. Diese wird in Form eines Faltungsintegrals aufgebaut [18, S.163].

$$y(t) = \int_{-\infty}^{\infty} h(t-\tau)x(\tau)d\tau \tag{2.6}$$

Das Eingangssignal $x(\tau)$ wird mit der Impulsantwort $h(t-\tau)$ multipliziert. Dabei wird $h(t)$ um τ zeitlich verschoben. Das Integral integriert über alle Werte von $+\infty$ bis $-\infty$ für τ. Das Ergebnis entspricht dem Ausgangssignal $y(t)$. Die Einführung von zeit- und amplitudendiskreter Signale erfordert jedoch für die Faltung diskreter Signale die Form einer Summe [19, S.165-166].

$$x[n] * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \tag{2.7}$$

Die diskreten Funktionswerte werden unter der Summe multipliziert. Ähnlich wie unter dem Faltungsintegral wird $y[n]$ jeweils um k verschoben. Einer zeit- und amplitudendiskreten digitalen Audiodatei liegt die Unterteilung durch Samplerate und Bittiefe zugrunde. Für jedes Sample existiert ein Amplitudenwert, der je nach Bittiefe dargestellt wird. Dieser Wert kann für die Faltung in die Summe eingesetzt werden [19, S.165-166]. Anwendungsfelder einer Faltung im Audiobereich finden sich in der (Re-)Synthese von akustischen Schallfeldern, die in Form einer Impulsantwort festgehalten wurden. Durch die Faltung eines trockenen Signals mit einer Impulsantwort, kann so der Raumeindruck auf die ursprünglich hallfreie Aufnahme übertragen werden [20, S.49].

2.1.4.1 Anregung durch einen Delta-Impuls

Für die Ermittlung der Systemfunktion eines LZI-Systems im Zeitbereich wird dieses in der Simulationspraxis meist mit einer Delta-Impuls-Distribution δ angeregt. Der Impuls wird oft vereinfacht als

$$\delta(t) = \begin{cases} 0 & \text{für } t \neq 0, \\ \infty & \text{für } t = 0 \end{cases} \tag{2.8}$$

dargestellt. Wenn $x(t) = \delta(t)$ für ein LZI-System zutrifft, wird dieses für eine infinitesimal kurze Dauer angeregt, wodurch $y(t) = \delta(t) * h(t) = h(t)$ gilt [18, S.164].

2.2 Räumliches Hören

Das räumliche Hören beschreibt den Zusammenhang zwischen Schallereignissen und den richtungsabhängigen spektralen Veränderungen sowie Laufzeit- und Pegelunterschieden, die das menschliche Gehör zur Lokalisierung von Hörereignissen nutzt. Die Darstellung eines Schallfelds mit mehreren Hörereignissen erfolgt in einem Polarkoordinatensystem. Ein menschlicher Kopf befindet sich im Zentrum des Systems. Der Nullpunkt befindet sich genau auf der Hälfte der interauralen Achse, also genau im Mittelpunkt der beiden Ohrkanaleingänge. Die Oberkanten des Ohrkanals liegen auf Achsenhöhe [21, S.88, 22, S.5].

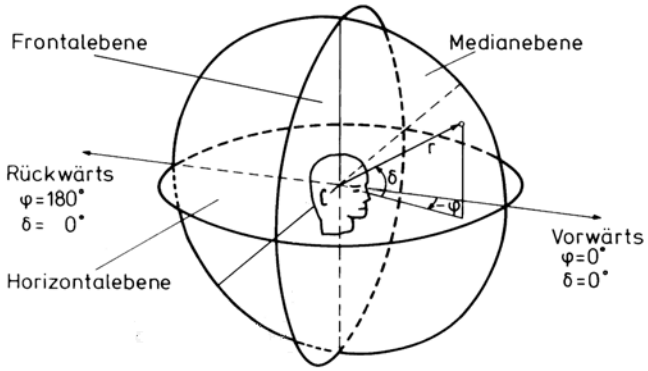


Abbildung 2.5: Kopfbezogenes Koordinatensystem (aus [21, S.88]).

Für die Beschreibung der Richtung von Hörereignissen ergeben sich drei Ebenen. Wie aus der Grafik 2.5 ersichtlich wird, spannen sich die Frontal-, Horizontal und Medianebene auf. Durch die Polarkoordinaten Azimut (ϕ), Elevation (δ) und der Entfernung r wird die Position einer Schallquelle beschrieben. Durch diese Darstellungsform werden die zwei essenziellen Informationen der Schallereignisrichtung, die laterale Auslenkung zur Darstellung auf der Horizontalebene sowie der zugehörige Elevationswinkel auf der Medianebene dargestellt. Eine auftretendes Hörereignis ist durch eine Darstellung auf den unterschiedlichen Ebenen klar lokalisierbar [21, S.88 f. 22, S.5, 23].
Bei Empfang eines Schallereignisses werden verschiedene Merkmale durch das Gehör ausgewertet, um den Ort des Hörereignisses zu lokalisieren. Zu unterscheiden sind die monotische, diotische und dichotische Beschallung. Bei einer monotischen Beschallungssituation empfängt nur eines der beiden Ohren ein Hörereignis. Durch die diotische Beschallung werden hingegen beide Ohren mit demselben Signal beschallt. Die dichotische Beschallung besagt, dass beide Ohren ein unterschiedliches Signal erhalten [24, S.88].
Das menschliche Gehör besitzt eine richtungs- und frequenzabhgängige Charakteristik. Je nach Position der Schallquelle im Polarkoordinatensystem wird der eintreffende Schall spektral gefiltert. Das bedeutet, dass zusätzlich zur Amplitude auch der Phasengang der eintreffenden Schallwelle beeinflusst wird. Die Lokalisierung des Gehörs funktioniert durch diese individuell auftretenden spektralen Änderungen. Die HRTF, siehe Abschnitt 2.3, charakterisiert die individuellen spektralen Änderungen beim Eintreffen eines Schallereignisses für beide Ohren. Der Prozess der Lokalisierung ist anhand der Aufteilung in die Horizontal- und Medianebene nachzuvollziehen [21, S.94 ff.].

2.2.1 Lokalisierung von Schallereignissen in der Medianebene

In einem Hörversuch zeigt Blauert 1970 die Funktionalität der Lokalisation des menschlichen Gehörs für Hörereignisse mit Einfallsrichtung aus der Medianebene.
Die aus dem Versuch resultierenden Blauert'schen Bänder bezeichnen Frequenzbänder, aus denen das Gehör die Richtung eines Hörereignisses auf der Medianebene interpretiert. Bei einer annähernd diotischen Beschallungssituation auf der Medianebene kann durch eine spektrale Anregung in den durch Blauert bestimmten Bändern ein Richtungsbezug durch das menschliche Gehör festgestellt werden. Dieser Effekt ist auf das Außenohr zurückzuführen. Dieses besitzt eine Filterwirkung und hebt Frequenzbänder nach Einfallsrichtung entsprechend an oder senkt sie ab. Die Filterung des einfallenden Schallfelds dient dem Interpretieren der Richtung auf der Medianebene, wie durch Grafik 2.6 zugeordnet. Das Außenohr des Menschen besitzt keine übergreifende einheitliche Form, sondern variiert stark im Aufbau sowie der resultierenden Filterwirkung [21, S.94 f.].

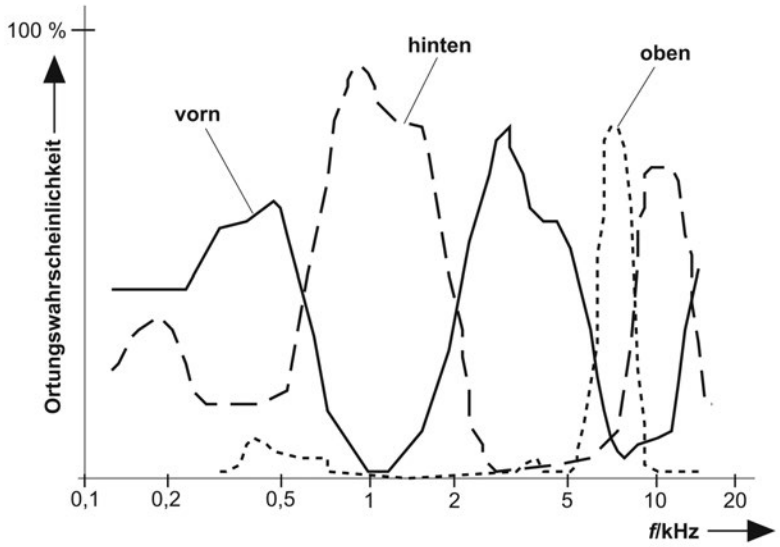
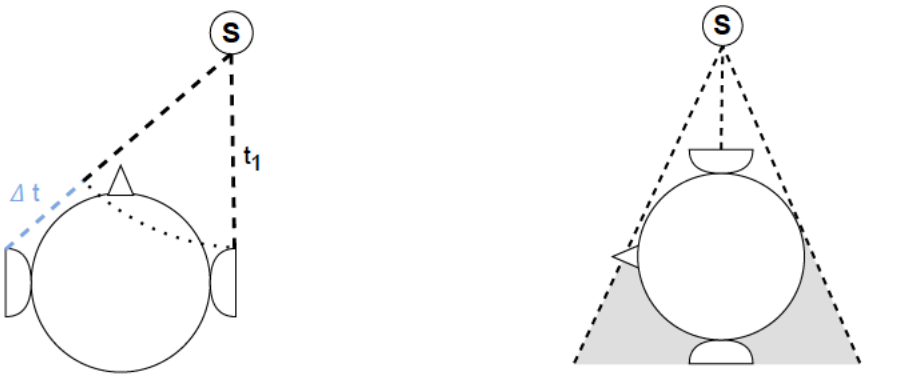


Abbildung 2.6: Aufbau der Blauert'schen Bänder nach [25, S.32].

2.2.2 Lokalisierung von Schallereignissen in der Horizontalebene

Das Eintreffen von Schall aus der Horizontalebene verursacht deutlichere Unterschiede beim Vergleich der beiden Ohrsignale. Für die Lokalisierung von Hörereignissen bei einer dichotischen Beschallung dienen dem Gehör die interaurale Pegeldifferenz (engl. Interaural Time Difference (ITD)) sowie die interaurale Zeitdifferenz (engl. Interaural Level Difference(ILD)).

Da in Grafik 2.7a die Schallquelle $\mathbf{S}$ unterschiedliche Abstände zu den beiden Ohren des Kopfes besitzt, erreicht der ausgesendete Schall nach der Dauer t_1 erst das rechte Ohr. Nach Ablauf von $t_1 + \Delta t$ trifft die Wellenfront ebenfalls am linken Ohr ein. Durch Δt wird die zusätzliche Laufzeit der Schallwelle, also die ITD bezeichnet. Eine ITD im Bereich von 10 bis 20 μs reicht dem menschlichen Gehör für eine Lokalisierung eines Hörereignisses aus. Bei Frequenzen über 1,5 kHz kann keine Zeitdifferenz mehr festgestellt werden [24, S.58]. Die ILD wird bei seitlich eintreffenden Schallereignissen ersichtlich. Da die Schallquelle $\mathbf{S}$ in Grafik 2.7b das rechte Ohr



(a) Laterale Auslenkung durch ITD. **(b)** Laterale Auslenkung durch ILD.

Abbildung 2.7: Lateralisation einer Schallquelle S.

direkt beschallt, und das linke Ohr aber in dem abgeschatteten Bereich des eigenen Kopfes liegt, entsteht eine Pegeldifferenz beim Vergleich beider Ohrsignale. Diese Unterschiede führen ebenfalls zu einer lateralen Auslenkung.

Die laterale Auslenkung in der Horizontalebene besitzt eine Lokalisationsunschärfe von $\pm 4°$, die Auslenkung auf der Medianebene $\pm 10°$ [21, S.95]. Die ILD und ITD werden vom menschlichen Gehör durch das spektrale Zerlegen des empfangenen Schallereignisses im Innenohr durch auditive Filter erkannt. Diese Bandpasssignale werden sowohl auf interaurale Phasenverschiebung sowie Gruppenlaufzeit analysiert. [21, S.92]

2.3 Head-Related Transfer Function (HRTF)

Für die Anwendung der genannten Methoden des Gehörs zur Lokalisation von Schall aus Kapitel 2.2 bedarf es der Möglichkeit die richtungsabhängigen Eigenschaften des Hörvorgangs zu charakterisieren. Die kopfbezogene Außenohrübertagungsfunktion (engl. Head-Related Transfer Function (HRTF)), beinhaltet spektrale Informationen eines zugehörigen Außenohrs. Eine HRTF beschreibt somit die Übertragungsfunktion eines freien Schallfelds aus einer Richtung bis zu einem Messpunkt des Hörkanals. Die bereits besprochenen spektralen Verzerrungen durch das Außenohr werden so in Form eines komplexen Spektrums festgehalten. Für die Berücksichtigung einer lateralen Auslenkung über die ITD und ILD in der Horizontalebene benötigt man ein HRTF-Paar beider Ohren für die binaurale Darstellung von linkem und rechtem Ohrsignal [21, 22, 26, 27].

2.3.1 Messung einer HRTF

Für die Messung einer amplituden- und phasengetreuen HRTF bietet sich die Pulstechnik, wie durch [26] beschrieben an. Die Messung erfolgt in einem reflexionsarmen Raum mit zwei Miniaturmikrofonen, die in beide Ohrkanäle des Probanden eingesetzt werden. Der Kopf des Hörers ist mit einer Vorrichtung fixiert. Zusätzlich wird ein sphärisch um den Kopf angeordneter Lautsprecher in einem gewissen Winkel auf das Außenohr gerichtet. Anschließend kann durch das Abspielen des in Abschnitt 2.1.4.1 aufgeführten Delta-Impulses über den Lautsprecher, eine kopfabhängige, sog. binaurale Impulsantwort (engl. Head-Related Impulse Response (HRIR)) über die Mikrofone gemessen werden.

Durch die Fouriertranformation der Impulsantwort erhält man die HRTF in spektraler Form. Dieses zeigt die im Spektrum auftretende Verzerrung des Außenohrs im Zusammenhang mit dem Einstrahlwinkel des im Versuchsaufbau eingebrachten Lautsprechers. Will man nun die Richtwirkung des Gehörs weiter durch Messungen bestimmen, bietet sich ein durch [23] und [28] beschriebener Messaufbau an. Mehrere Lautsprecher befinden sich sphärisch angeordnet mit definierten Abständen in der Horizontalebene links und rechts vom Hörer sowie mit einem Elevationswinkel auf der Medianebene. Über jeden Lautsprecher wird ein Deltaimpuls zur Ermittlung der richtungsspezifischen spektralen Veränderungen abgespielt und die Impulsantwort gemessen. Durch die Wiederholung des Verfahrens für alle Lautsprecherpositionen erhält man eine Anzahl von HRTF-Paaren, die für die Lautsprecherpositionen richtungsbezogene Hörereignisse repräsentieren können. Die durch [29] beschriebene Messung wurde für 710 Lautsprecherpositionen wiederholt. Da der Proband in dieser Zeit fixiert ist, bietet sich die Pulstechnik aufgrund der kürzeren Messdauer an.

2.3.2 Anwendungsbereiche einer HRTF

Für die Anwendung der gewonnenen Erkenntnisse einer HRTF gibt es mehrere Ansätze für das binaurale Abhören von Audiosignalen. Durch das Einbringen eines Kunstkopfes, eines gemessenen HRTF-Paars, sowie durch die Nutzung eines Faltungsprozesses ergeben sich mehrere Stufen der Binauralisierung von Audiosignalen [28].

a) Die Aufnahme eines Schallfelds durch einen Kunstkopf ist die einfachste Form der Anwendung einer HRTF. Ein Nachbau eines menschlichen Kopfs mit Mikrofonen in den Ohrkanälen wird in einem Schallfeld platziert und integriert eine gemittelte HRTF des Kunstkopfaufbaus direkt in die Aufnahme. Die beiden aufgenommen Signale für das linke und rechte Ohr werden auf einem Kopfhörer mit einem linken und rechten Kanal abgehört. Nachteil des Verfahrens ist jedoch, dass die HRTF sowie die Ausrichtung des Kopfes im Nachhinein nicht mehr beeinflusst werden können [30].
b) Durch das Prinzip der Faltung lässt sich eine HRTF als Filter auf ein Signal anwenden. Durch die nachträglich durchzuführende Signalverarbeitung kann die einzubringende HRTF beeinflusst und dadurch die Hörereignisrichtung geändert werden. Jedoch wird so simultan nur ein HRTF-Paar eingesetzt. Die Realisierung der Richtungsänderung von mehreren Hörereignissen des Schallfelds wird grundsätzlich nicht in Echtzeit umsetzbar [23, 30].
c) Eine Weiterführung des Faltungsprozesses aus b) integriert eine Anzahl an HRTF's, die wie in Abschnitt 2.3.1 gemessen wurden. Daraus resultiert ein sphärisches Modell an virtuellen Lautsprecherpositionen mit hinterlegten HRTF-Paaren. Über dieses Modell lässt sich eine Faltung von Audiosignalen durchführen, wodurch eine Richtungswirkung für mehrere Hörereignisse simultan durchgeführt werden kann. Ein Anwendungsbereich des Modells findet sich in Verbindung mit dem Ambisonicsformat, das im folgenden Abschnitt weiter erläutert wird [30, 31, S.9].

2.4 Ambisonics

Das Ambisonics Format dient zur Aufnahme, Speicherung und Resynthese von Schallfeldern. Die Idee eines Formats zur Schallfeldsynthese wurde durch Michael Gerzon und Peter Craven in den 1970er Jahren formuliert [32]. Grundlage bilden die En- und Dekodierung mehrerer Audiokanäle,

auf denen richtungsabhängige Informationen einer sphärischen Abtastung des Schallfelds transportiert werden. Überträgt man die Grundidee für eine Aufnahme auf ein Mikrofonarray, bildet dieses auf Basis der Charakteristik mehrerer koinzident angeordneter Mikrofone eine sphärische Charakteristik aus. Die abzutastende Sphäre kann in ihre harmonischen Kugelflächenfunktionen (engl. spherical harmonics) zerlegt werden, siehe Grafik 2.8. Das Ambisonics B-Format wird durch die Ordnung der beinhalteten harmonischen Kugelflächenfunktionen definiert. Die nötigen Kanäle einer Ordnung lassen sich durch $n_{Kanal} = (n+1)^2$ bestimmen. Desto höher die Ordnung eines Ambisonics B-Formats, desto genauer kann die Richtung eines Hörereignisses abgebildet werden. Das Ambisonics Format erster Ordnung (engl. First Order Ambisonics (FOA)) ist die niedrigste Form der Ambisonicsformate mit richtungsbezogener Hörereignisdarstellung. Die derzeitige technische Entwicklung erlaubt es mit höheren Ordnungen (engl. Higher Order Ambisonics (HOA)) zu arbeiten [31, S.7-12].

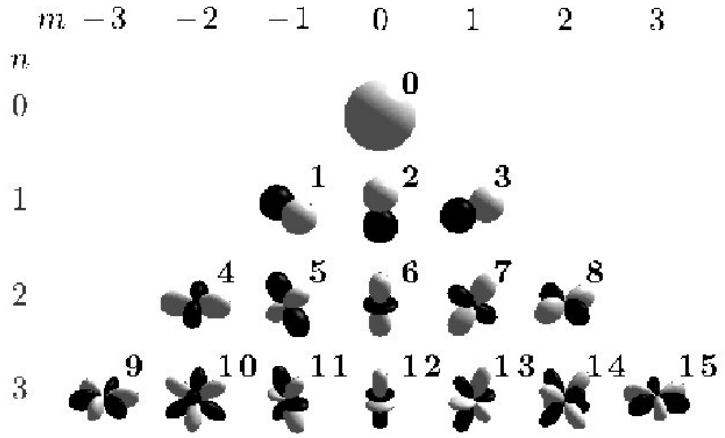


Abbildung 2.8: Harmonische Kugelflächenfunktionen nach Ordnung n und Rang m sortiert (aus [33]).

2.4.1 Formate und Kanalbenennung

Für die Nutzung des Ambisonics Formats bieten sich zwei Formatstandardisierungen an. Bei einer Nutzung von Formaten niedrigerer Ordnung ist der Einsatz des klassischen B-Formats nach Vorgabe der FuMa Standardisierung ohne Probleme umsetzbar. Die Kanäle eines FOA B-Formats werden nach den letzten vier Buchstaben des Alphabets W, X, Y und Z benannt. Der W-Kanal beinhaltet eine sphärische Aufnahme aller Richtungen, die X, Y und Z Signale stellen die jeweilige Achse des Polarkoordinatensystems in Form einer Achter-Mikrofoncharakteristik dar [34]. Mit höherer Ordnung werden Buchstaben beginnend vor den benutzten Buchstaben des Alphabets genutzt, was die Anwendung im Bereich höherer Ordnungen umständlich macht. Die Kanäle des B-Formats werden meist durch Aufnahme eines Mikrofonarrays im A-Format festgehalten, das erst in das B-Format dekodiert werden muss [35, 36].
Das AmbiX Format integriert hingegen andere Standardisierungen für die Anwendung in höheren Ordnungen. So gibt die Ambisonics Kanalnummer (engl. Ambisonics Channel Number (ACN)) den Kanälen zur Benennung je nach Rang m und Ordnung n der zuzuordnenden harmonischen Kugelflächenfunktion eine natürliche Zahl. Die Kanalzahl wird über

$$ACN = n^2 + n + m \tag{2.9}$$

berechnet [31, S.120, 35, 37]. Ein Vergleich der Benennungen ist Tabelle 2.1 zu entnehmen.

2.4.2 Normalisierung

Die ursprünglich durch Gerzon empfohlene Gewichtung des W-Kanals wurde durch Malham und Furse in [38] durch ihre Standardisierung FuMa übernommen und bis in die dritte Ordnung fortgeführt. Die Amplitude des W-Kanals wird dabei um den Faktor $\frac{1}{\sqrt{2}}$ abgeschwächt, die Kanäle der zweiten und dritten Ordnung sind ebenfalls mit einer Gewichtung zu multiplizieren. Dadurch wird das Übersteuern des W-Kanals verhindert sowie die ordnungsabhängigen Schallenergieanteile berücksichtigt.
Bei der Nutzung des AmbiX Formates bietet sich die Normalisierungsvorschrift SN3D an. Diese garantiert eine Schallenergieverteilung mit Gewichtung der jeweiligen Ordnung. Die Koeffizienten dafür lassen sich über

$$N_n^{|m|} = \sqrt{\frac{(2-\delta_m)}{4\pi}\frac{(n-|m|)!}{(n+|m|)!}},\ \delta_m = \begin{cases} 0 & \text{für } m \neq 0, \\ 1 & \text{für } m = 0 \end{cases} \tag{2.10}$$

berechnen. Das Kronecker delta δ_m ist je nach Rang m einzusetzen [37]. Das AmbiX Format bevorzugt die Nutzung von durch SN3D normalisierten Kanälen [31, S.120, 36, 37]. Die durch SN3D geltenden Kanalgewichtungen können in Tabelle 2.1 bis zur dritten Ordnung nachvollzogen werden.

Kanalzuordnung				**Normalisierung**	
Ordnung n	Rang m	ACN	FuMa	FuMa	SN3D
0	0	0	W	$\frac{1}{\sqrt{2}}$	0.282
1	-1	1	Y	1	0.282
	0	2	Z	1	0.282
	1	3	X	1	0.282
2	-2	4	V	$\frac{2}{\sqrt{3}}$	0.081
	-1	5	T	$\frac{2}{\sqrt{3}}$	0.163
	0	6	R	1	0.282
	1	7	S	$\frac{2}{\sqrt{3}}$	0.163
	2	8	U	$\frac{2}{\sqrt{3}}$	0.081
3	-3	9	Q	$\sqrt{\frac{8}{5}}$	0.015
	-2	10	O	$\frac{3}{\sqrt{5}}$	0.036
	-1	11	M	$\sqrt{\frac{45}{32}}$	0.115
	0	12	K	1	0.282
	1	13	L	$\sqrt{\frac{45}{32}}$	0.115
	2	14	N	$\frac{3}{\sqrt{5}}$	0.036
	3	15	P	$\sqrt{\frac{8}{5}}$	0.015

Tabelle 2.1: Zusammenhänge zu den unterschiedlichen Benennungskonventionen sowie Normalisierungsvorgaben, aus [36, 38].

2.4.3 Harmonische Kugelflächenfunktionen

Die harmonischen Kugelflächenfunktionen sind eine mathematische Verteilung zur Beschreibung einer Kugeloberfläche, siehe Grafik 2.8. Im Kontext zu Ambisonics beschreiben die harmonischen Kugelflächenfunktionen eine sphärische Abtastung eines Schallfelds. Die mathematischen Funktionen repräsentieren das Schallfeld in Form einer Druckverteilung. Eine Anzahl von harmonischen Kugelflächenfunktionen bilden eine Sphäre mit einer durch die Ordnung bestimmten Genauigkeit. Der mathematische Hintergrund der Funktionen kann für interessierte Leser durch [37, S.3] nachvollzogen werden.

2.4.4 Dekodierung von Ambisonicsformaten

Nach dem Enkodieren eines Audiostreams in Form eines B-Formats, stellt sich die Frage des Nutzens eines Signalaufbaus mit den bereits beschriebenen Eigenschaften. Der Prozess der Dekodierung ist einer der großen Vorteile des Formats und kann für verschiedene Anwendungsfälle durchgeführt werden. Das Abbilden einer Aufnahme im Ambisonics Format auf Lautsprechern und Kopfhörern soll im Folgenden betrachtet werden.
Für das Abspielen eines Ambisonics Formats auf einer Anzahl von Lautsprechern wird ein Einheitsvektor $\boldsymbol{\theta_1}$ mit der Ausrichtung eines jeweiligen Lautsprechers mit den Einheitsvektoren $\boldsymbol{\theta_{X,Y,Z}}$ der X, Y und Z Kanäle multipliziert. Daraus resultiert ein Vorfaktor, der den Beitrag des Lautsprechers zu der jeweiligen Richtung angibt. Die Lautsprechersignale S lassen sich für die Anzahl L der Lautsprecher als

$$\begin{bmatrix} S_1 \\ \vdots \\ S_L \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & \boldsymbol{\theta_1^T \theta_X} & \boldsymbol{\theta_1^T \theta_Y} & \boldsymbol{\theta_1^T \theta_Z} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \boldsymbol{\theta_L^T \theta_X} & \boldsymbol{\theta_L^T \theta_Y} & \boldsymbol{\theta_L^T \theta_Z} \end{bmatrix} \begin{bmatrix} W \\ X \\ Y \\ Z \end{bmatrix} \tag{2.11}$$

berechnen. Die Zuordnung der Kanäle lässt sich gut anhand des W-Kanals nachvollziehen. Dieser wird für alle Lautsprechersignale mit einer 1 multipliziert und wirkt so auf alle Lautsprecher des Aufbaus. Die Lautsprecheranordnung kann beispielhaft anhand Grafik 2.9 nachvollzogen werden. In dieser wird nochmals die Kanalzuordnung mittels Vorfaktoren ersichtlich. Die Additions- und Subtraktionszeichen beziehen sich auf einen positiven oder negativen Vorfaktor. Bei abweichenden Abhörsituationen mit unterschiedlicher Anzahl an Lautsprechern wird deren Ausrichtung in Form des zugehörigen Einheitsvektors durch die Dekodierungsmatrix mit berücksichtigt [31, S.8 ff. 36].

Die Dekodierung auf Kopfhörern funktioniert ähnlich zu der Wiedergabe auf multiplen Lautsprechern einer Abhörsituation. Die Kopfhörer bieten zwei Kanäle $L_{(Ohr)}$ und $R_{(Ohr)}$ für das Abspielen von Inhalten auf den beiden verbauten Lautsprechermembranen. Für die Erzeugung einer Darstellung von Hörereignissen mit Richtungsbezug wird sich einer Anzahl von HRTF-Funktionen bedient [39], die in Abschnitt 2.3 bereits beschrieben wurden. Für jegliche darzustellende Richtung wird eine HRTF bereitgelegt, die mit einer enkodierten Form des Ambisonics-Formats

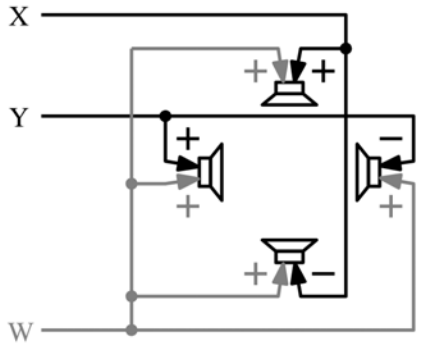


Abbildung 2.9: Lautsprechersetup für ein 2D Ambisonics-Format aus [31, S.8].

gefaltet wird. Dadurch erhält man für ein vereinfachtes 2D-Ambisonics-Format

$$\begin{bmatrix} L_{(Ohr)} \\ R_{(Ohr)} \end{bmatrix} = \begin{bmatrix} h_L^0(t)* & h_L^{90}(t)* & h_L^{180}(t)* & h_L^{-90}(t)* \\ h_R^0(t)* & h_R^{90}(t)* & h_R^{180}(t)* & h_R^{-90}(t)* \end{bmatrix} \begin{bmatrix} F \\ L \\ B \\ R \end{bmatrix} \tag{2.12}$$

für die Darstellung des Faltungsaufbaus mit den virtuellen Lautsprechern $Front(F), Left(L)$, $Back(B), Right(R)$. Durch die Addition und Subtraktion zwischen X und Y- Kanal wird das Ambisonicsformat wie durch Grafik 2.10 dargestellt, auf den virtuellen Lautsprechern abgebildet. Wie durch Grafik 2.10 repräsentiert, dienen die virtuellen Lautsprecherpositionen zur Resynthese des Schallfelds [31, S.8 ff.].
Für eine realistische Wiedergabe des Schallfelds über Kopfhörer muss sich das virtuelle kugelförmige Schallfeld der dreidimensionalen Rotation und Neigung des Kopfes entgegensetzen. Dies wird über eine Rotationsmatrix $\mathbf{R}$ realisiert, deren mathematische Herleitung [31, S.106] zu entnehmen ist. Die Rotationsmatrix führt eine zyz-Rotation durch, wodurch jede Drehung der Kugel realisiert werden kann. Für die resultierenden Kanäle $\tilde{X}, \tilde{Y}$ und $\tilde{Z}$ des B-Formats wird durch

$$\begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \end{bmatrix} = \mathbf{R}(\alpha, \beta, \gamma) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.13}$$

die Rotation der Ambisonics-Szene um die Winkel α, β, γ berücksichtigt [31, S.12-13].

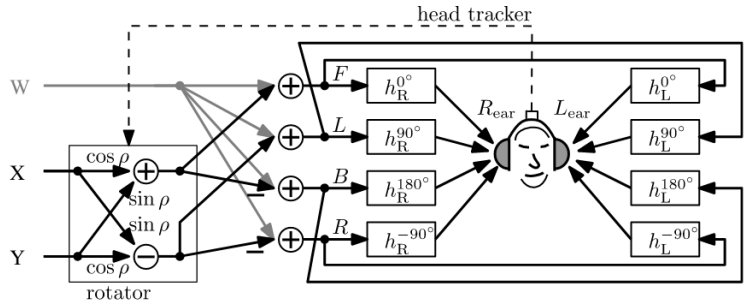


Abbildung 2.10: Binaurale Dekodierung eines 2D-Ambisonics-Formats, aus [31, S.9].

3 Implementierung einer Akustik-Simulation in die Virtual Reality

Der folgende Teil der Arbeit beschreibt einen möglichen Modellaufbau einer Virtual Reality (VR)-Umgebung mit integrierten Impulsantworten aus einer akustischen Simulation. Dieser Aufbau soll es ermöglichen, in einer virtuellen Räumlichkeit Inhalte zu hören und Grundlage für eine akustische Demonstration bieten. Der Grundaufbau wird aus der von D'Amelio beschriebenen Offline-Verarbeitung von Simulationsdaten aus [40] entnommen und an eine aktive Signalverarbeitung angepasst. Darunter ist zu verstehen, dass nicht bereits gefaltete Signale eingebracht werden, sondern der Faltungsprozess durch Effektpozessoren der Signalinfrastruktur bewerkstelligt wird. Die folgenden Funktionsabschnitte werden aufgrund von Anforderungen der Müller-BBM BSO Akustikabteilung detailliert ausgeführt.
Ziel ist es, möglichst realistische Höreindrücke mit den visuellen Eindrücken des Raumes zu verknüpfen. Die VR-Applikation baut auf festen Abhörpositionen auf, da eine Bewegung des Zuhörers im Raum während eines Konzerts eher unüblich ist. Die Kopfbewegung soll jedoch akustisch abgebildet werden.

3.1 Überblick

Essenzieller Bestandteil der durchzuführenden Zusammenführung mehrerer technischer Komponenten, ist ein VR-Headset, das es ermöglicht eine entsprechende Aufbereitung von Raummodellen für Nutzer darzustellen. Zusätzlich soll das Headset eine Audioausgabe für den Tragenden realisieren. Die Erstellung eines Raummodells mit hörbaren Inhalten bedarf einer Plattform, die eine VR-Entwicklungsumgebung für eine VR-Brille bietet, siehe Abbildung 3.4. Dieser umfassende Modellaufbau wird deshalb in der Gaming-Engine Unity aufgebaut, deren nötiger Funktionsumfang unter Abschnitt 3.3 weiter erläutert wird. Durch die zusätzlich benötigte Audiosignalverarbeitung wird sich einer externen Audioengine bedient, die mit Unity verknüpfbar ist. Diese Audioengine nennt sich Wwise und bietet Möglichkeiten zur Signalverarbeitung und Interaktion nach einer entsprechenden Programmierung. Letztendlich wird der Modellaufbau durch objektbasierte Programmierung in C# gesteuert und alle nötigen Interaktionen dadurch realisiert. Der zu erstellende Modellaufbau ist durch Abbildung 3.1 nachzuvollziehen.
Für die Durchführung wird sich eines Raumes bedient, dessen Messdaten, Raummodell sowie Simulationsergebnisse bereits abrufbar sind. Diese Daten werden dem Verfasser für die Implementierung durch die Akustikabteilung der Müller-BBM Building Solutions (BSO) GmbH zur Verfügung gestellt. Das Konzerthaus Blaibach ist ein fertiggestelltes akustisches Planungsprojekt der Raumakustikabteilung mit den nötigen Messdaten und einem nutzbaren Modell. Die Simulationsdaten werden durch Odeon erstellt und als Ambisonics-Impulsantwort exportiert. Der Prozess der Simulation einer Impulsantwort wurde bereits durch Abschnitt 2.1.3 geschildert. Die Messdaten einer Schallintensitätssonde für den Saal werden ebenfalls in das Ambisonics-Format gewandelt.
Das Konzerthaus Blaibach wurde durch die Initiative des Baritons Thomas E. Bauer und Architekten Peter Haimerl unter der Vorgabe geplant und gebaut, eine exzellente Akustik für Zuschauer und Künstler zu schaffen. Haimerl ist für seine extravaganten Bauprojekte mit dem

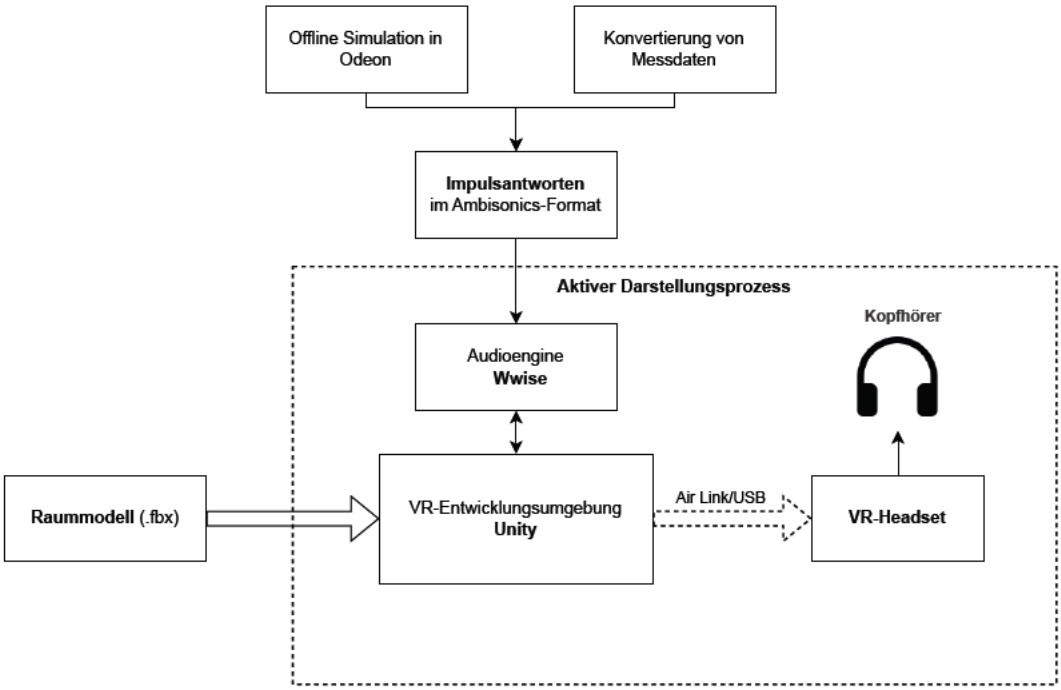


Abbildung 3.1: Überblick des Modellaufbaus mit den nötigen Software- und Hardwarekomponenten.

(a) Innenansicht des Konzertsaals [41]. **(b)** Außenansicht des Konzerthauses [42].

Abbildung 3.2: Ansichten des fertiggestellten Konzertsaals in Blaibach.

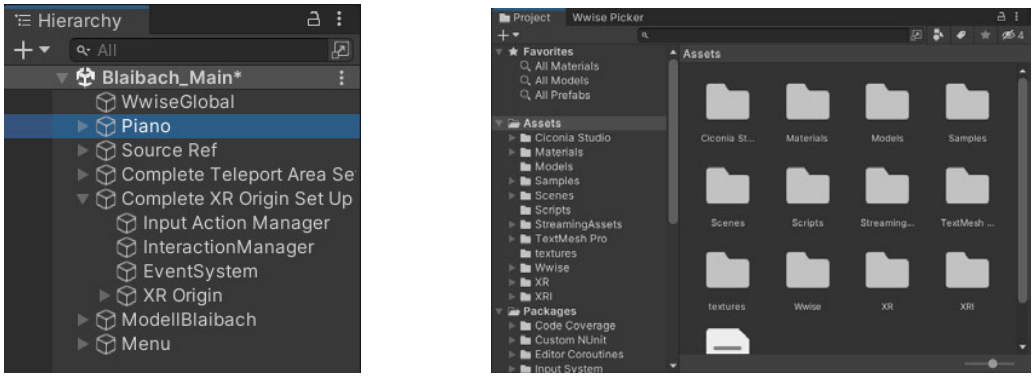
Baustoff Beton bekannt [43]. Die Bauweise des Konzertsaals ist den Grafiken 3.2b und 3.2a zu entnehmen. Die Akustik des Raumes zeichnet sich durch eine gemittelte Nachhallzeit $T_{20} = 1,4\,s$ des besetzten Saals aus, die unter anderem durch eine Schallwegführung anhand einer Facettierung der Betonwände erreicht wird. Die akustische Simulation ermöglichte auf Grundlage einer Strahlenverfolgung das Planen der genauen Ausrichtung der Betonstruktur. Da die Schallstrahlen den Zuhörerbereich erst nach einer bestimmten Anzahl von Reflexionen erreichen, kann so die Nachhallzeit im Saal beeinflusst werden. Der Baustoff Beton ist als schallhart einzuordnen [44], weshalb durch einfallenden Schall mit einer geringen Schallstreuung durch die glatten Betonoberflächen zu rechnen ist.

3.2 Vorbereitung des Raummodells

Für eine Integration in die Gaming-Engine Unity wird eine Wandlung des bestehenden Modells für den Konzertsaal Blaibach durchgeführt. Da das Modell in einem firmeneigenen Programm bearbeitet wurde, sind die Raumbegrenzungsflächen als Polygone hinterlegt. Eine Ansicht der Modellstruktur kann Anhang B entnommen werden. Durch entsprechende Tools werden die Polygone in Rhino [45], einer CAD-Software, in Flächenobjekte gewandelt, da eine Bearbeitung von Flächen in Unity so besser durchzuführen ist. Für den Transfer des Modells bietet sich das Dateiformat .fbx an, das sowohl Rhino erzeugen als auch Unity importieren kann. Das von Autodesk entwickelte Format speichert die Objekte eines Modells und kann von verschiedenen Programmen wie Blender eingebunden werden [46]. Der Import eines 3D-Modells im .fbx Format erlaubt in Unity die Bearbeitung aller einzelnen Flächen, wodurch die Gestaltung mit Texturen und die Ausleuchtung vereinfacht wird. Ähnliche Funktionalität bietet das .obj Format, welches in Unity jedoch nur die Bearbeitung des Raumes als ganzes Objekt zulässt.

3.3 Die Entwicklungsumgebung Unity

Unity ist eine Entwicklungsumgebung für die Gestaltung von Computerspielen und vergleichbaren Anwendungen. Die Arbeitsweise des Programms ist szenenbasiert aufgebaut und bietet über eine Objekthierarchie Zugriff auf alle enthaltenen sogenannten Game Objects der Szene, siehe Grafik 3.3a. Das Hinzufügen von Modellen und Texturen wird durch das Ablegen der Dateien im



(a) Szenenbasierte Hierarchie. **(b)** Projektfenster mit Assets-Ordner.

Abbildung 3.3: Aufbau der Entwicklungsumgebung Unity.

Projektfenster (Abbildung 3.3b) ermöglicht. Jedes erstellte Projekt enthält eine standardisierte Ordnerstruktur, die es erlaubt individuelle Inhalte strukturiert zu hinterlegen. Außerdem wird durch das Einbinden von erhältlichen Softwarepaketen (engl. packages) weitere Funktionalität für Unity eingebunden. Der Funktionsumfang eines Pakets wird in der Projektordnerstruktur in verschiedenen Dateien hinterlegt. Eine beispielhafte Implementierung eines Pakets erläutert der folgende Abschnitt 3.3.1. Die darstellende Instanz der Entwicklungsumgebung findet sich im Szenenfenster, das den Modellaufbau visualisiert und zusätzliche Funktionalität zur Darstellung bietet. Am Rand des Fensters finden sich Werkzeuge zur Bewegung von Objekten, Darstellungsoptionen sowie Filter der darzustellenden Markierungen und Hilfslinien. Ein Inspektorfenster gibt zusätzlich Informationen zu Objekten und zeigt deren Funktionalitäten. Für jedes Objekt können in diesem Fenster Komponenten (engl. components) geladen werden, die Funktionen für

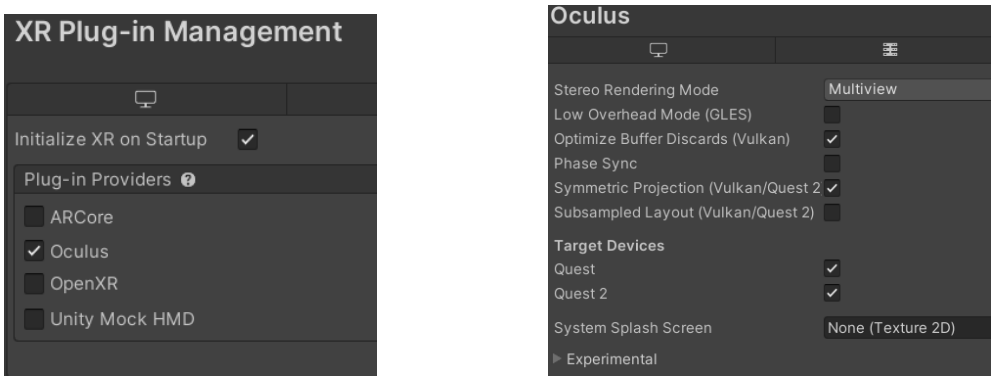
die Realisierung der Anwendungssteuerung hinzufügen. Dies kann beispielsweise in Form einer Renderkomponente erfolgen, die Objekte wie Würfel und Sphären darstellen lässt, oder auch durch angefertigte Skripte, die Funktionen der Anwendung für den Anwender kontrollieren und steuern.

3.3.1 VR-Integration

Unity bietet durch das eigens entwickelte Paket XR Interaktion Toolkit Schnittstellen für die Entwicklung einer VR-Umgebung und kann durch den Package-Manager über die Git-URL *com.unity.xr. interaction.toolkit* eingebunden werden. Das VR-Headset kann so im Entwicklungsprozess aktiv für das Testen der erstellten Modellumgebung genutzt werden. Die Darstellung des Szenenfensters wird entsprechend auf die beiden Bildschirme der VR-Brille übertragen.
Eine Verbindung zu dem Headset wird herstellerabhängig mit einem Kabel oder drahtlos über WLAN hergestellt. Die für den Modellaufbau genutzte Meta Quest 2 [47] ermöglicht beide Verbindungsformen. Das mitgelieferte USB-Kabel baut die Verbindung über den USB-Standard zu dem entsprechenden Rechner auf. Das drahtlose, von Meta gelieferte Verbindungsprotokoll Air Link ermöglicht einen Verbindungsaufbau zu der Entwicklungsumgebung über ein WLAN Netzwerk. Für dieses sind jedoch Vorgaben für einen erfolgreichen Verbindungsaufbau einzuhalten, weshalb die Nutzung der drahtlosen Verbindung mehr Aufwand erfordert. Das WLAN Signal kann nur auf einem 5GHz-Kanal übertragen werden. Zusätzlich muss der Hostrechner der Entwicklungsumgebung über ein Ethernet-Kabel an den Router des Netzwerks angeschlossen sein. Dies dient der Latenz- und Datenbandbreitenminimierung. Eine Installation des Occulus Treibers auf dem Hostrechner ist in beiden Anwendungsfällen notwendig.
Für die einfache Nutzung der aufgebauten Verbindung zwischen VR-Headset und Hostrechner

Abbildung 3.4: Frontansicht des Meta Quest 2 VR-Headsets [47].

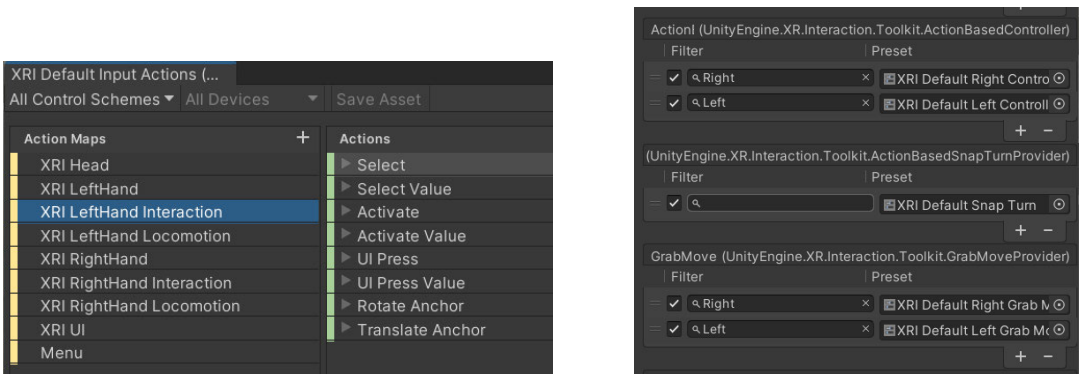
in Unity sind mehrere Arbeitsschritte notwendig, die das XR-Interaction Toolkit weiter in die Entwicklungsumgebung implementieren. Für die Nutzung der Testwiedergabe des Modellaufbaus in Unity muss das XR-Interaction Toolkit in den Projekteinstellungen richtig hinterlegt werden. Die Installation ermöglicht das Debugging von Projekten auf Basis des Open XR Standards. Für die Implementierung der bereits vorgestellten Meta Quest 2 (s. Abbildung 3.4) wird jedoch der proprietäre Standard wie in Grafik 3.5a visualisiert aktiviert. Die Wahl der Standardisierung ist durch den Einsatz des spezifischen VR-Headsets unter Ausschluss anderer Fabrikate für den Modellaufbau begründet. Da die Firma Oculus durch Facebook im Jahr 2014 aufgekauft wurde [48], wird an der ein oder anderen Stelle die alte Firmenbezeichnung für die Benennung von



(a) XR Interaction Toolkit in den Projekteinstellungen. **(b)** Einstellungen der Verbindung eines Meta Headsets.

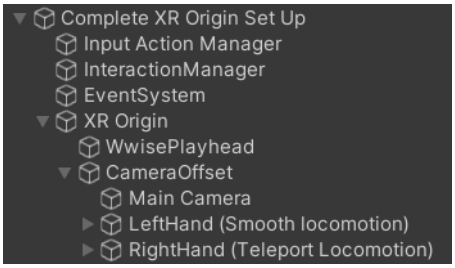
Abbildung 3.5: Einrichtung des XR-Interaction-Toolkits.

VR-Komponenten genutzt. Durch die Auswahl der Unterkategorie des XR-Plug-in Managements in den Projekteinstellungen lassen sich anschließend Einstellungen der Verbindung zu dem VR-Headset beeinflussen. Die in Grafik 3.5b getroffenen Einstellungen haben sich für die Nutzung über eine Kabelverbindung durch den Verfasser der Arbeit erprobt. Wichtig ist die Auswahl des zu verwendenden Endgeräts (engl. Target Device) in dem Einstellungsfenster. Der Stereo Rendering Mode beeinflusst, wie Unity das Bild für die beiden Displays des VR-Headsets generiert. Multiview ist eine ressourcenschonende Möglichkeit, den Renderingprozess von stereoskopischen Bildinhalten durchzuführen und wird deshalb für den Modellaufbau gewählt.
Für den Abschluss der Implementierung des Toolkits werden die sogenannten Input Actions, also Eingabeaktionen importiert, die eine Interaktionssteuerung der Controller eines VR-Headsets hinterlegen. In den zu importierenden Starter Assets des XR Interaction Toolkits finden sich die XRI Default Input Actions Map (Abbildung 3.6a) sowie weitere Actionmaps, die auf Objekten Funktionen der VR-Implementierung hinterlegen. Diese Vorlagen sowie die Zuweisung an Eingaben werden durch den Inspektor der Projektstruktur hinzugefügt. Den Starter Assets ist außerdem eine Demoszene zu entnehmen, die einen beispielhaften Aufbau einer Szene für eine VR-Implementierung bereitstellt. Zuletzt muss dem Preset-Manager über einen Filter die jeweilige Funktionsseite des XR-Presets, wie in Abbildung 3.6b beispielhaft dargestellt, zugewiesen werden.

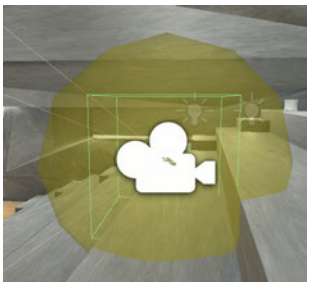


(a) Input-Actions Map. **(b)** Preset-Manager in den Projekteinstellungen mit zugeordneten Filtern.

Abbildung 3.6: Implementierung des XR-Interaction-Toolkits.



(a) Hierarchie des Spielerobjekts.



(b) Visualisierung des Spielerobjekts im Szenenfenster.

Abbildung 3.7: Aufbau eines VR-Spielerobjekts.

3.3.2 Aufbau des VR-Spielerobjekts

Unity ist als Gaming Engine darauf ausgerichtet, ein Spielerobjekt durch eine Szenerie zu bewegen, um so die Illusion einer Bewegung für den Nutzer zu erzeugen. Das Spielerobjekt beinhaltet Komponenten, die eine Visualisierung der Spielerumgebung erfassen. Interaktionselemente sind in Form von Skripten hinterlegt und reagieren beispielsweise auf Berührungen anderer Objekte, oder realisieren die Fortbewegung des Objekts in einer Szenerie durch die Tastatureingaben eines Nutzers. Für die Anwendung in einer VR-Umgebung werden durch das XR-Interaction Toolkit verschiedene bereits erstellte Game Objects hinterlegt, die eine Interaktions- und Visualisierungsschnittstelle für ein VR-Headset integrieren. Das Spielerobjekt repräsentiert in dem beschriebenen Modellaufbau den Hörer, der sich auf unterschiedliche Positionen des Raumes bewegen kann. Der entstehende Höreindruck passt sich also für das Spielerobjekt an.
Die oberste Ebene des Spielerobjekts bilden Skripte und Ordnerobjekte mit Funktionalitäten, die sich in Eingabe (Input Action Manager), Interaktion (Interaction Manager) und Visualisierung (XR Origin) gliedern lassen, siehe Abbildung 3.7a. Das XR Interaction Toolkit hinterlegt Skripte, die den nötigen Funktionsumfang der obersten Hierarchieebene implementieren. Die Skripte werden als Komponenten auf entsprechenden Game Objects hinterlegt und sind für die Nutzung des Modellaufbaus ohne weitere Schritte nutzbar. Das XR Origin-Objekt beinhaltet hingegen weitere Objekte wie die virtuelle Kamera, zwei Controllerobjekte sowie den Wwise-Playhead, der für die Audiowiedergabe relevant wird, siehe 3.4.7. Auf den Controllerobjekten finden sich Komponenten für die Belegung der Controller-Tasten. Ein beispielhaftes Spielerobjekt kann der Demoszene des XR Interaction Toolkits entnommen werden. Der Aufbau der verschiedenen Elemente resultiert in einer Visualisierung im Szenenfenster, siehe Abbildung 3.7b

3.3.3 Aufbereitung des importierten Modells

Die Textur von Oberflächen sowie die Beleuchtung des darzustellenden Raumes spielen eine wichtige Rolle für den realistischen Wahrnehmungseindruck des VR-Erlebnisses. Die Grundlage für das Einsetzen von Texturen und beleuchtenden Objekten ist durch das importierte Modell gegeben. Dieses muss aus Flächenobjekten bestehen, um eine Bearbeitung zu ermöglichen. Unity erlaubt das Hinterlegen von Texturen auf Objekten in Form von Materialien. Diese können unabhängig der zu belegenden Objekte im Nachhinein angepasst werden und wirken sich so auf die Auswahl des hinterlegten Materials aus. Nach dem Erstellen einer Textur auf einem Material kann dieses auf einem oder mehreren Flächenobjekten hinterlegt werden. Dieser erste Schritt der Modelloptimierung ermöglicht eine optische Annäherung an das reale Umfeld des virtuellen Raums, siehe Abbildung 3.8. Für den Größenbezug des Raumes wird das Modell eines

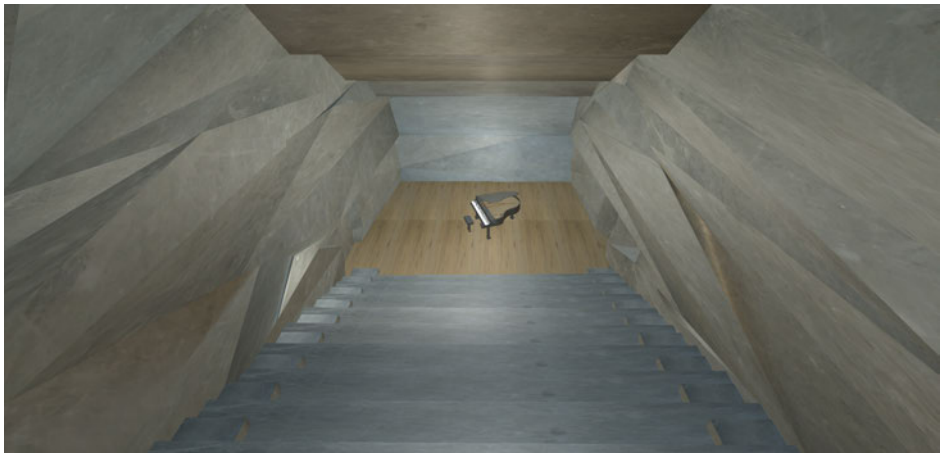


Abbildung 3.8: Der Konzertsaal Blaibach mit hinterlegten Texturen auf dem importieren Raummodell.

Klavierflügels eingefügt, das dem Nutzer eine Orientierung für Dimension und Abmessung des Raums geben soll.
In einem zweiten Schritt der Modellaufbereitung werden Lichtquellen in den Raum eingebracht. Die Beleuchtung von Räumlichkeiten kann in Unity auf zwei unterschiedliche Arten erfolgen. Über die Szenenhierarchie kann eine Lichtquelle als Game Object eingefügt werden. So bieten sich Einstellungen zur Art der Lichtquelle und anderen Parametern wie Lichtfarbe und Lichtstärke. Die resultierende Beleuchtung ist in Echtzeit, jedoch ohne Schattenbildung zu verfolgen. Anstatt von Lichtquellobjekten können ebenso Flächenobjekte Licht emittieren. Dies ist durch die einstellbaren Eigenschaften eines Materials festlegbar. Durch Auswahl eines Materials mit aktivierter Lichtemission und gewünschter Farbe kann Flächenobjekten so die Funktion eines Lichtquellobjekts übertragen werden. Vorteil ist, dass das Licht anhand der Form des Flächenobjekts abgestrahlt wird. Das Modell Blaibach hat aufgrund der geplanten Beleuchtung bereits hinterlegte Beleuchtungsflächen, die so anhand eines entsprechenden Materials direkt im Modell übernommen werden können. Entscheidender Schritt für eine Beleuchtung mit Schattenwurf ist jedoch das Erstellen einer *Baked Lightmap*. Durch eine Simulation ermittelt Unity die Lichtemission in dem Modell und speichert dies in Bildform. Durch diesen Offline-Berechnungsprozess werden keine Ressourcen während dem Ausführen der Anwendung verbraucht. Wie durch Abbildung 3.9 sichtbar, vermittelt das Raummodell eine weitere Annäherung an den realen Raum durch das Einbinden der Lichtemission mit Schattenbildung.

Abbildung 3.9: Raummodell mit erstellter Baked Light Map.

3.4 Audiosignalverarbeitung der Audioengine Wwise

Die Audioengine Wwise bildet das Rückgrat für die Verarbeitung der hörbaren Inhalte des Modellaufbaus. Um die akustischen Gegebenheiten eines Raumes zu beurteilen, können Kunstkopfaufnahmen über Kopfhörer abgehört werden. Jedoch bringt dies, wie in Abschnitt 2.3.2 besprochen, Einschränkungen mit sich. Diese Einschränkungen lassen sich durch den Einsatz des Ambisonics-Formats minimieren. Die Flexibilität eines rotierbaren Schallfelds sowie die unabhängige Faltung mit einer HRTF kann durch eine entsprechend aufgebaute Signalverarbeitung umgesetzt werden. Sowohl die gemessenen, als auch die simulierten Messdaten des Konzertsaals werden in Form von neun-kanaligen Waveform Audio File-Dateien im Ambisonicsformat zweiter Ordnung mit Normalisierungskonvention FuMa geliefert, siehe 2.4. Diese Impulsantworten lassen sich über einen Faltungsalgorithmus mit trockenen Signalen falten. Wie durch D'Amelio in [40] beschrieben, bietet die Audioengine Wwise den nötigen Funktionsumfang der Signalverarbeitung. Der von D'Amelio beschriebene Aufbau verfolgt jedoch eine Implementierung von gefalteten Signalen, die in den Signalfluss von Wwise integriert werden. Die im folgenden Abschnitt beschriebene Signalinfrastruktur wird um aktive Faltungsinstanzen erweitert, um so die Flexibilität des zu erstellenden Modellaufbaus zu maximieren.

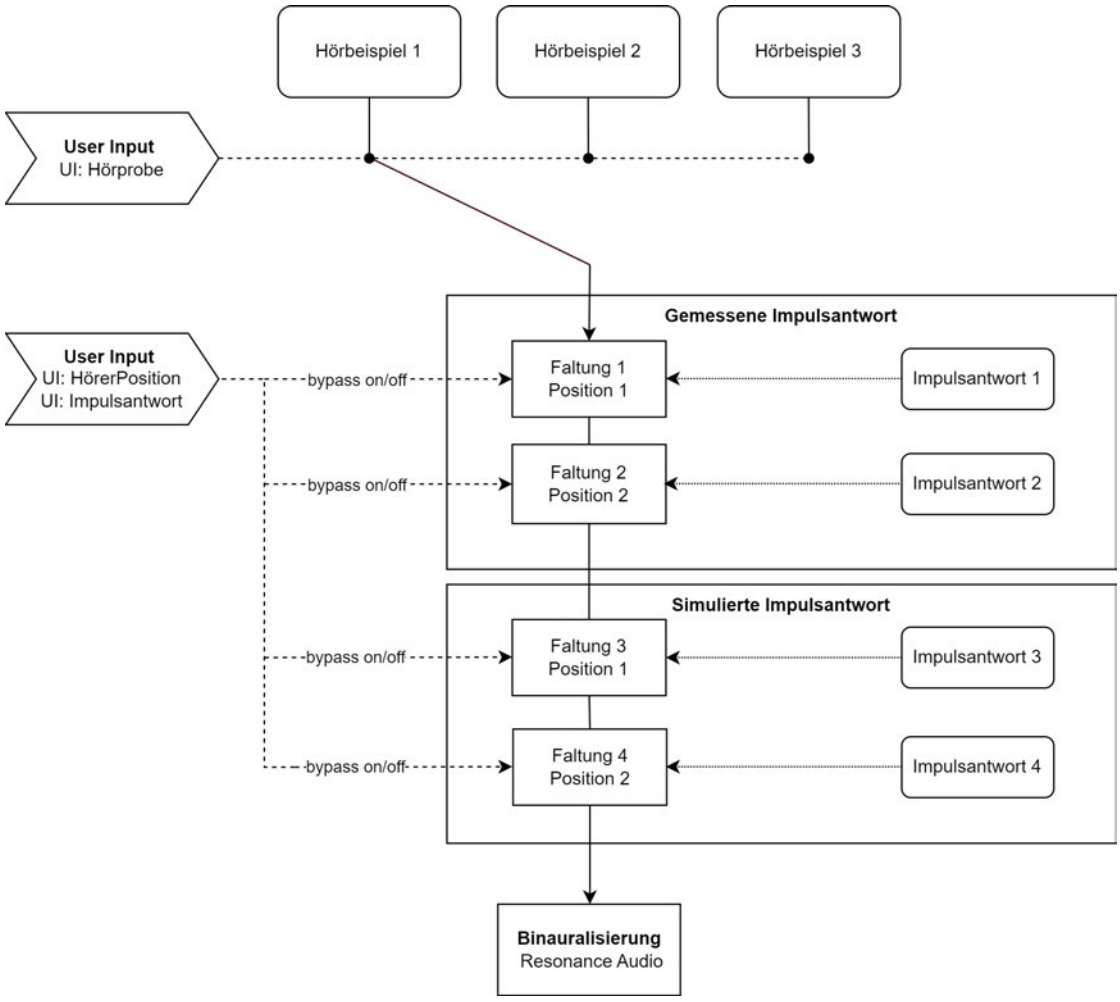


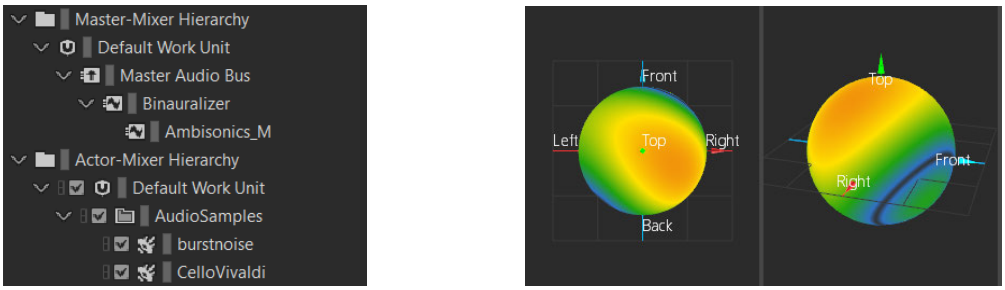
Abbildung 3.10: Signalflussdiagramm der Faltung trockener Signale mit Impulsantworten (IR) in Wwise.

Da sowohl simulierte als auch gemessene Impulsantworten vorhanden sind, kann ein Vergleich zwischen Simulation und Realität implementiert werden. Dafür müssen, wie in dem Signalflussdiagramm 3.10 dargestellt, genügend Instanzen des Faltungsalgorithmus in die Audioverarbeitung eingebracht werden. Ein Planungseinsatz des Modells würde jedoch aufgrund der noch nicht messbaren Daten ausschließlich mit simulierten Impulsantworten durchgeführt werden. Im Folgenden wird die durch das Diagramm 3.10 visualisierte Signalverarbeitung und Implementierung durch Wwise weiter erklärt.

3.4.1 Die Oberfläche der Audioengine

Wwise ist ein durch Audiokinetic vertriebenes Softwareprodukt und beinhaltet einen Funktionsumfang, der für die Programmierung von Computerspielen ausgelegt ist [49]. Wwise verknüpft automatisch eine Unity Umgebung mit dem jeweiligen Wwise-Projekt durch das Integrieren mehrerer Ordner in der Unity-Projektordnerstruktur. Die Ordnerstruktur des Wwise-Projekts wird über den sogenannten Wwise Picker als Registerkarte im Projektfenster von Unity angezeigt. Durch das Doppelklicken auf Elemente werden diese in Wwise angewählt oder angespielt. Die Oberfläche der Audioengine basiert ähnlich wie Unity auf einer Hierarchie, die zusätzlich den Signalfluss von abgespielten Audiodateien repräsentiert. In Ordnern können hier Audiodateien abgelegt, Buswege erstellt und Auxwege genutzt werden. Die Struktur des Projekts wird nutzbar, sobald diese in einer Sound Bank hinterlegt wird. Durch diese werden Daten, einbegriffen der Busstruktur, mit in Unity übernommen. Das Hinzufügen von Teilen der Hierarchie erfolgt in dem Sound Bank Layout der Wwise Oberfläche.

Die gewünschten Hörbeispiele können in der Actor-Mixer Hierarchie hinzugefügt werden. Durch das Anwählen einer Audioquelldatei werden weitere Bearbeitungsmöglichkeiten durch das Designer-Fenster mit mehreren Registerkarten angezeigt. Die Ansicht der Registerkarte *General Settings* erinnert an den Aufbau eines Kanalzugs, bekannt aus digitalen Audioworkstations (DAW) oder digitalen Mischpulten mit zusätzlichen Funktionen wie der Tonhöhe oder auch speziellen Aux-Wegen, die durch Unity ferngesteuert werden können. Über das Festlegen des Ausgangs-Busses wird das Routing des Ausspielwegs der Audioquelldatei beeinflusst. Besonderheit ist hier, dass Wwise Busse verschiedene Kanalformate unterstützen, was durch Abschnitt 3.4.3 weiter vertieft wird. Zusätzlich kann eine Dauerschleife für die Wiedergabe von Audioquelldateien eingestellt werden. Die Registerkarte Effects erlaubt das Hinterlegen von mitgelieferten Insert-Effekten, die hier für die notwendigen Faltungen genutzt werden, (s. Abschnitt 3.4.4). VST-Plug-Ins sind nicht einsetzbar. Jeder Insert-Punkt besitzt einen Bypass-Schalter, der den jeweiligen Effektprozessor in die Signalkette ein- oder ausschließt. Die Positioning-Registerkarte findet Einsatz bei der Konfiguration von 3D-Audiobussen oder bei durch Unity gesteuerten Audioquellen. Die States-Registerkarte ermöglicht das Einbinden einer Gruppe an States für die Steuerung von Parametern einer Audioauswahl. Durch das Aufrufen eines States durch Unity können Lautstärke, der Bypass von Effekten und weitere Parameter gesteuert werden. Höchste Instanz der Bushierarchie bildet der Master Audio Bus, siehe Abbildung 3.11a. Durch diesen wird ebenfalls der genaue Ausspielweg auf der Hardware des Computers festgelegt. Dies dient jedoch rein für Abhörzwecke, da Unity die Audioausgabe ab dem Starten der Anwendung übernimmt. Der Master Audio Bus unterstützt ebenfalls immersive Kanalformate wie Dolby Atmos sowie die Ausgabe im Ambisonicsformat.
Die Event-Registerkarte repräsentiert eine Schnittstelle zwischen Unity und einer Auswahl der



(a) Mixbushierarchie in Wwise.

(b) *3D Meter* aus Wwise für einen gefalteten Sinus-Testton im Ambisonicsformat.

Abbildung 3.11: Aufbau eines VR-Spielerobjekts.

Funktionen von Wwise. Ein Event kann beispielsweise eine Audioquelldatei abspielen oder auch die globale Wiedergabe unterbinden. Zusätzlich lässt sich sequentiell Automation mit einem Event aufrufen, wodurch Lautstärke oder auch Filter mit einer gewissen Übergangszeit die Audiowiedergabe beeinflussen.
Das Metering-Fenster bietet mehrere Funktionen zum Betrachten der Audiobusse eines Projekts. Hervorzuheben ist die 3D-Meter-Ansicht für die Betrachtung von Ambisonics-Bussen, siehe Abbildung 3.11b. Eine Lautheitsbewertung ist ebenfalls durch vier Messinstanzen in die Signalinfrastruktur zu integrieren. Diese können unterschiedliche Audiomixbusse überprüfen, was besonders für das Erstellen der Sounddesigns von Computerspielen relevant sein kann.

3.4.2 Mixbus-Struktur

Die Mixbusstruktur in Wwise ist über den Project Explorer in der Audiohierarchie konfigurierbar. Jeder erstellte Audiobus besitzt ähnliche Bearbeitungsmöglichkeiten zu den Audioquelldateien. Die Busse können für verschiedene Kanalformate von klassischem 1.0 Mono bis 7.1.4 für Dolby Atmos konfiguriert werden. Zusätzlich sind Auro3D Kanalformate auswählbar. Die für den Modellaufbau relevante Kanalform des Ambisonicsformats wird bis zur fünften Ordnung unterstützt. Untergeordnete Busse können sich an das Format des Busses einer Hierarchieebene höher anpassen. Die Oberfläche gibt außerdem über den Bus Status Auskunft zu der Konfiguration von Ein- und Ausgängen. Im Falle einer Konvertierung durch den Bus werden die Kanäle automatisch der nötigen Anzahl angepasst. Rafaely beschreibt die notwendigen Schritte der Übertragung von akustischen Szenen auf ein anderes Abhörmedium in [30, S.3]. Die Busstruktur legt Grundlage für diese durchzuführende Verarbeitung digitaler Audiosignale. Die vorzunehmende Signalführung ist Abbildung 3.12 zu entnehmen.

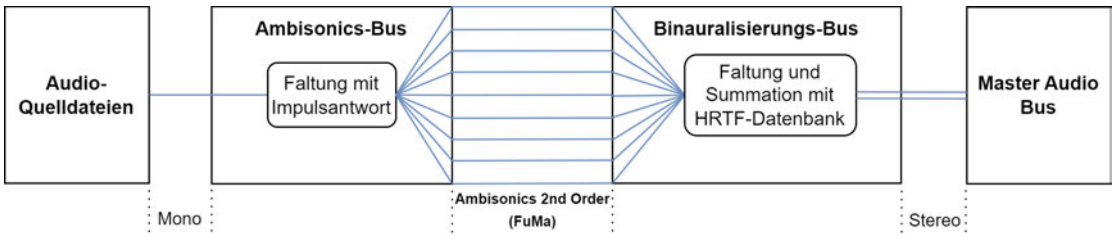


Abbildung 3.12: Mixbusstruktur der Signalverarbeitung.

Die Mixbusstruktur integriert die durch Abschnitte 2.1.4, 2.3 und 2.4 beschriebenen Methodiken und Theorien der Audiosignalverarbeitung. Die Faltung von digitalen Audiosignalen ist ein

zentraler Prozess für die vorzunehmende Signalkette. Anforderung an die Signalverarbeitung ist eine Struktur, die es erlaubt trockene Audiosignale in Mono mit einer Ambisonics Impulsantwort zu falten. Dadurch wird die Notwendigkeit der akustischen Reproduktion der Räumlichkeit abgedeckt. Das Einbinden des dadurch entstehenden Ambisonics-Signals in den Modellaufbau mit VR-Headset wird über eine Binauralisierung, wie in Abschnitt 2.4.4 beschrieben, realisiert. So wird eine Wiedergabe des sphärisch repräsentierten Schallfelds auf die Kopfhörer des VR-Headsets dekodiert, siehe Abbildung 3.12.
Der Modellaufbau basiert auf einem Ambisonics-Bus zweiter Ordnung. Dies ist durch die Messdaten des Konzertsaals zu begründen. Um die Vergleichbarkeit zwischen simulierter und gemessener Impulsantworten zu gewährleisten, wird das Format durch die höchste verfügbare Ordnung von Messdaten und Simulation gewählt. Wie in 2.4 beschrieben, verbessert sich die räumliche Auflösung mit steigender Ordnung. Der Master Audio Bus nimmt lediglich das binauralisierte Signal des Busses einer Hierarchie-Ebene niedriger an, und gibt es an den Audiotreiber des Computers weiter.

3.4.3 Mixbus-Konfiguration

Für die gewünschten Funktionen der Busse müssen diese unterschiedlich in Wwise konfiguriert werden. Die schematische Darstellung der Busstruktur gibt nochmals Auskunft über die Hierarchieordnung, siehe Abbildung 3.13. Der *Ambisonics_M*-Bus wird von den Audioquelldateien direkt beschickt. Dies ist über das Designer-Layout über die Auswahl des Ausgangs-Busses einzustellen. Da durch den *Ambisonics_M*-Bus die Faltung der trockenen Audiosignale mit einer Ambisonics Impulsantwort realisiert werden soll, werden entsprechende Faltungsalgorithmen als Insert-Effekte geladen. Der *Binauralizer*-Bus nutzt ebenso eine Effektinstanz zur Umsetzung der Binauralisierug. Wwise integriert einen Binauralisierung durch Resonance Audio. Das entsprechende Plug-In ist jedoch über die Registerkarte Mixer Plug-In hinzuzufügen. Ein weiterer wichtiger Konfigurationsschritt ist die Positionierung des Busses, wodurch später in Unity eine Rotation des Ambisonics-Signals realisiert wird. Über die Registerkarte Positioning des Designer-Layouts wird die Positionierung des Busses festgelegt. Für den *Ambisonics_M*-Bus wird über die Auswahl des Listener Relative Routings das hinterlegte Ambisonicsformat je nach Orientierung des VR-Headsets rotiert. Durch die Einstellung wird eine sphärische Abstrahlung des Ambisonics-Formats in Unity ermöglicht. Die Rotation des Formats berücksichtigt außerdem der Binauralisierungsprozess. Für die Positionierung muss zwingend eine Attenuation-Kurve für das Spread-Parameter erstellt werden, sodass der eingestellte Wert bei jedem Abstandswert 100% beträgt. So bleibt die Schallverteilung konstant und ist unabhängig von der Bewegung des Spielerobjekts. Es ist außerdem darauf zu achten, ausschließlich Busse zu benutzen. Aux-Wege werden in Wwise anders gehandhabt und erzielen nicht die gewünschten Funktionen für den Modellaufbau.

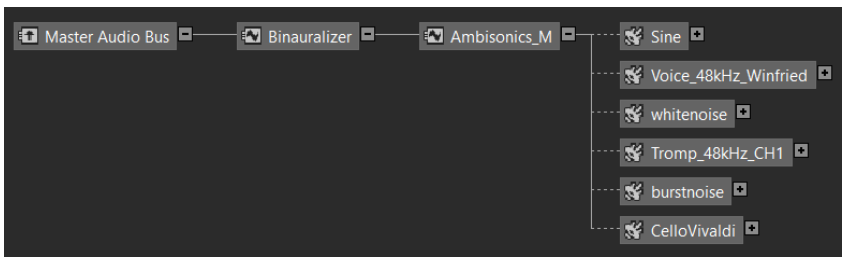


Abbildung 3.13: Audiohierarchie aus dem schematischen Layout in Wwise.

3.4.4 Signalverarbeitung

Die Insert Effekte auf der bereits beschriebenen Busstruktur bewerkstelligen die gewünschte Signalverarbeitung. Für die Faltung der trockenen Audiosignale wird der Audiokinetic Convolution Reverb genutzt, um die Ambisonics Impulsantworten einzubinden. Nach dem Laden

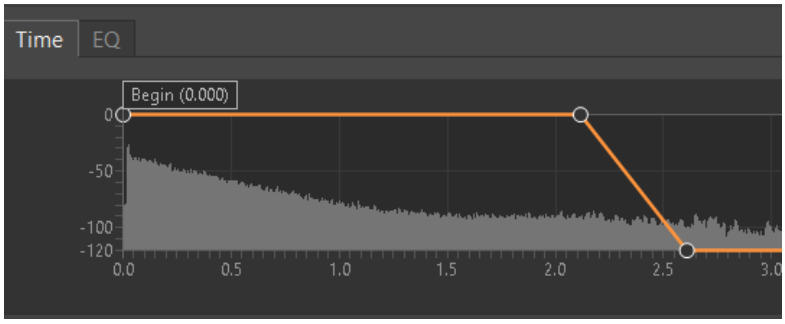


Abbildung 3.14: Ambisonics-Impulsantwort dargestellt durch den Audiokinetic Convolution Reverb.

der gewünschten Impulsantwort kann das Kanalformat über die entsprechende Auswahl der Benutzeroberfläche (Abbildung 3.15) ausgewählt werden. Da die Impulsantwort im Ambisonicsformat zweiter Ordnung mit Normalisierungskonvention FuMa geladen wurde, kann diese Konfiguration ausgewählt werden. Alternativ wird auch die AmbiX Formatkonvention unterstützt. Die Darstellung der Impulsantwort erlaubt über eine Lautstärkeautomation, die Wirkungsdauer einzuschränken, wodurch Rechenleistung eingespart werden kann. Nach dem Verstreichen der vorhergehend gemittelten Nachhallzeit $T_{60} = 1,4\,s$, wird unter zusätzlicher Berücksichtigung des Schallpegelabfalls der Cut-Off Punkt wie durch Abbildung 3.14 visualisiert gesetzt.
Für die Umsetzung mehrerer Hörerpositionen müssen, wie in Signalflussdiagramm 3.10 angedeutet, mehrere Instanzen des Effektprozessors seriell in den Signalfluss eingegliedert werden. Da zusätzlich zu den zwei angedachten Hörerpositionen im Saal der Vergleich zwischen simulierten und gemessenen Impulsantworten umgesetzt wird, werden vier Instanzen des Faltungsalgorithmus benötigt. In jeden Kanalzug eines Mixbusses können vier Insert-Effekte geladen werden. Dies deckt den Bedarf für die notwendigen Effektprozessoren ab. Die Binauralisierung des

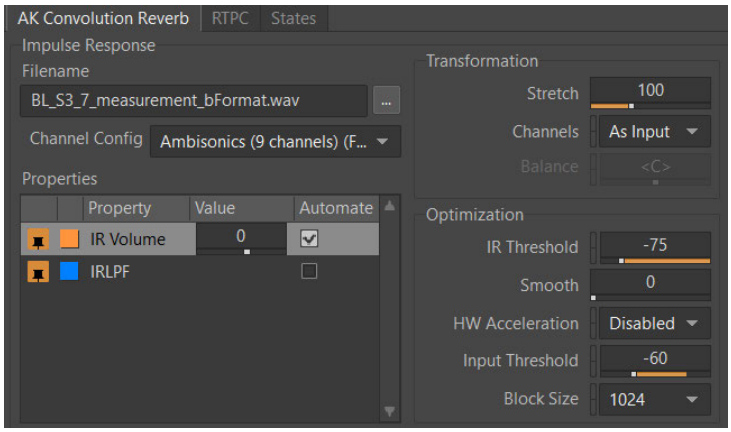


Abbildung 3.15: Benutzeroberfläche des Audiokinetic Convolution Reverbs.

resultierenden Ambisonics Signals wird durch die Resonance Audio-Suite in Wwise realisiert. Resonance Audio integriert Möglichkeiten der Bearbeitung von immersiver Audiowiedergabe für Entwicklungsplattformen wie Unity, Unreal Engine und unterstützt zusätzlich die mobilen

Betriebssysteme Android sowie IOS [50]. Aufgrund einer Zusammenarbeit von Audiokinetic und den Entwicklern der Resonance Audio SDK, kann eine Binauralisierung durch eine Instanz des Resonance Audio Binauralisierers durchgeführt werden [51]. Dieser arbeitet ohne weitere nötige Einstellungen des Nutzers. Die durch Rafaely in [30] geforderten Parameter der auditorischen Wahrnehmung werden durch den Dekodierungsprozess des binauralen Abhörens übergeben. Die Faltung mit einem HRTF-Paar integriert die essenziellen richtungsgebenden Informationen für das Lokalisieren von Hörereignissen. Die aus Abschnitt 2.2 bekannten Kenngrößen der Schallquellenlokalisation werden durch die Faltung mit einem HRTF-Paar auf ein Hörereignis übertragen. Der Prozess der Binauralisierung ist durch Abschnitt 2.3.2 sowie 2.4.4 für den Zusammenhang einer Ambisonics-Szene erläutert.

3.4.5 Aufbau der Audio-Event Struktur

Ein weiterer Schritt der Verknüpfung von Wwise in die Entwicklungsumgebung Unity beinhaltet die Erstellung von Events. Diese Events müssen für jede Audiodatei erstellt werden, um so Unity den Zugriff und das Abspielen zu ermöglichen. Ein Event kann sowohl die Wiedergabe starten als auch stoppen. Zusätzlich können Parameter des Kanalzugs einer gewählten Audiodatei verändert werden. Da für den Modellaufbau mehrere Hörbeispiele in das Modell des Konzertsaals eingebunden werden sollen, wird für die entsprechenden Audioquelldateien ein Event erstellt. Die gewählten Hörbeispiele wurden Sammlungen trockener Instrumentenaufnahmen entnommen. Diese sind über die Website *OpenAIR* [52] und von der *Universitat Pompeu Fabra Barcelona* [53] bezogen worden.

3.4.6 Zustandssteuerung der Signalkette

Für die Anforderungen des Modellaufbaus werden, wie unter Abschnitt 3.4.4 beschrieben, vier Effektprozessoren für die Umsetzung benötigt. Jeder Insert-Punkt der vier Faltungsalgorithmen besitzt einen Bypass-Schalter, der für die Realisierung von Positionswechseln des VR-Headsets genutzt werden kann. Da alle vier Instanzen der Faltungsalgorithmen nicht gleichzeitig in die Signalkette eingegliedert werden dürfen, können die Bypass-Schalter genutzt werden, um die gerade nicht benötigten Prozessoren aus der Signalkette zu nehmen. Bei einem Positionswechsel des Spielerobjekts in Unity muss also der richtige Faltungsprozess in der Signalkette aktiv sein. Die Steuerung der Signalkette wird durch Wwise in Form sogenannter State-Gruppen realisiert. Wie durch D'Amelio in [40] angedeutet, besitzt jede Gruppe mehrere States, also Zustände die unter anderem den Bypass-Schalter der Effektprozessoren steuern können. Für den Modellaufbau wird deshalb die State-Gruppe *EffectBypass* erstellt. Die Gruppe enthält vier Zustände, die durch ihren Namen die Herkunft und Position der Impulsantwort ausdrücken. Wie in Abbildung

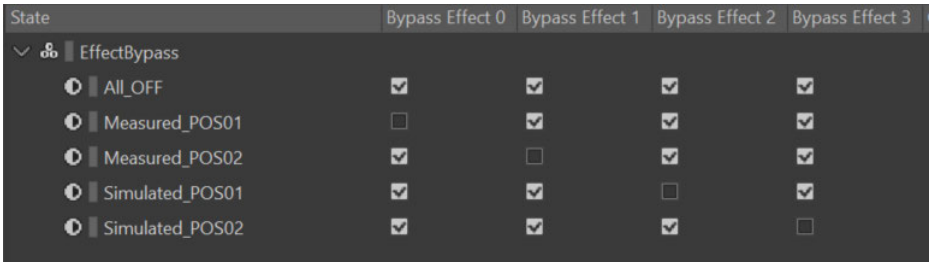


Abbildung 3.16: Zuordnung der Bypass-Schalterstellung.

3.16 dargestellt, kann nach dem Hinzufügen der State-Gruppe für die inkludierten States der jeweilige Bypass-Zustand ausgewählt werden. Zusätzlich wird ein State erstellt, durch den alle Effektprozessoren aus der Signalkette genommen werden. Dieser Zustand dient zu Testzwecken.

3.4.7 Implementierung in Unity

Der Zusammenhang von Unity und Wwise bedarf der in 3.4.1 beschriebenen Projektverknüpfung. Für die Nutzung von Wwise sind mehrere Objekte mit hinterlegten Komponenten in der Unity Objekthierarchie notwendig.

a) Für die Initialisierung der zu ladenden Sound Bank ist es notwendig, die sogenannten *AkBank* und *AkInitializer*-Komponenten auf einem Objekt des Projekts zu hinterlegen. Die Benennung *Ak* ist eine Abkürzung für den Firmennamen Audiokinetic und dient der Identifikation von Komponenten. Für den Modellaufbau wird das **WwiseGlobal** Objekt erstellt, auf dem neben diesen Komponenten unter anderen ein Skript zum Abrufen der Audioevents aus Wwise liegt.
b) Der **WwisePlayhead** liegt als Objekt in der Hierarchieebene über der Kamera, wodurch er nicht von der Rotation des Kameraobjekts beeinflusst wird. Auf dem **WwisePlayhead** wird die *AkRadialEmitter*-Komponente hinterlegt, die für die Wiedergabe des Ambisonicsbusses verantwortlich ist. Durch das Laden der Komponente wird eine Sphäre mit einzustellenden Radien auf dem Spielerobjekt eingefügt, siehe Abbildung 3.7b.
c) Dem Kameraobjekt der Unityszene ist außerdem eine **AkSpatialAudioListener**-Komponente hinzuzufügen. Durch diese werden an Wwise Daten zu der Position und Rotation des Kameraobjekts weitergegeben.

3.4.7.1 Aufbau einer Event-basierten Wiedergabesteuerung

Für die Implementierung der Audiowiedergabe besteht die Notwendigkeit eines Skripts in C# für den Aufruf von Audioevents. Die Ansteuerung von in Wwise erstellten Events beinhaltet mehrere notwendige Anforderungen der Wiedergabesteuerung. Ziel des Skripts ist es, je nach Auswahl des Nutzers aus einem Dropdown-Menü, das entsprechende Event aufzurufen. Zusätzlich soll verhindert werden, ein Hörbeispiel mehrfach zu starten. Ausgehend dieser Anforderungen sind durch eine Klasse für alle eingebrachten Hörbeispiele Boolesche Variablen hinterlegt. Der Startwert der Variablen wird entsprechend der Startauswahl der Menüführung durch Quelltext 3.1 initialisiert.

```
1 void Start()
2 {
3     celloIsActive = true;
4     burstsoundIsActive = false;
5     whitenoiseIsActive = false;
6     trompeteIsActive = false;
7     opernstimmeIsActive = false;
8     playheadHold = false;
9 }
```

Quelltext 3.1: Zuweisung von Wahrheitswerten in der Startanweisung des *EventTrigger*-Skripts.

Durch das Dropdown-Menü der Benutzeroberfläche wird ein Eingabewert an eine *HandleInputData*-Funktion (s. Quelltext 3.2) weitergegeben. Diese ruft je nach eingegebenen Wert die gewünschten Wahrheitswerte der deklarierten Variablen auf. Das Dropdown-Menü gibt bei der Auswahl einer Option einen Zahlenwert von null an aufwärts aus. In der Funktion sind entsprechend der Option die Wahrheitswerte hinterlegt. Zusätzlich wird die *stopAll* Funktion aufgerufen, die jede laufende Wiedergabe unterbindet.

```
1  public void HandleInputData(int val)
2  {
3      if (val == 0)
4      {
5          celloIsActive = true;
6          burstsoundIsActive = false;
7          whitenoiseIsActive = false;
8          trompeteIsActive = false;
9          opernstimmeIsActive = false;
10         stopAll();
11         playHead();
12     }
13 ...}
```

Quelltext 3.2: Auszug der Funktion zur Änderung der Wahrheitswerte.

Über die Funktion playHead (s. Quelltext 3.3) wird der Aufruf des jeweiligen Events bewerkstelligt. Je nach Wahrheitswert wird das Event der zugehörigen Variable ausgelöst, dass den Wert *true* besitzt. Außerdem ist durch den Einsatz der Variable *playbackHold* ein doppeltes Aufrufen von Events nicht möglich. Sobald die *stopAll*-Funktion aufgerufen wird, setzt diese den Wahrheitswert zurück auf *false*, wodurch das Abspielen anderer Hörbeispiele wieder möglich wird. So kann verhindert werden, dass die mehrfache Auswahl einer Option des Dropdown-Menüs die Wiedergabe beeinflusst.

```
1  public void playHead()
2  {
3      if (playheadHold == true)
4      {
5          return;
6      }
7      else if (playheadHold == false)
8      {
9          if (celloIsActive == true)
10         {
11             AkSoundEngine.PostEvent("Play_CelloVivaldi", playSphere);
12             playheadHold = true;
13         }
14 ...}
```

Quelltext 3.3: Auszug der Funktion für den Aufruf von Events.

Zeile 11 des Quelltextes 3.3 ruft ein Event mit dem Namen *Play_CelloVivaldi* auf und referenziert das Objekt der *playSphere* für die Wiedergabe. Das *playSphere*-Objekt ist als globales Objekt deklariert und mit der sphärischen Schallquelle des *WwisePlayheads*-Objekts verknüpft.

3.4.7.2 Zustandssteuerung der Effektprozessoren

Die Umschaltung verschiedener Zustände der Signalkette bedarf ebenfalls eines Skripts in Unity. Dieses wird aufgrund der ebenfalls integrierten Funktion der Teleportation als *TeleportMenu* bezeichnet. Vorerst wird jedoch nur die Schaltung der States durch das Skript geschildert.

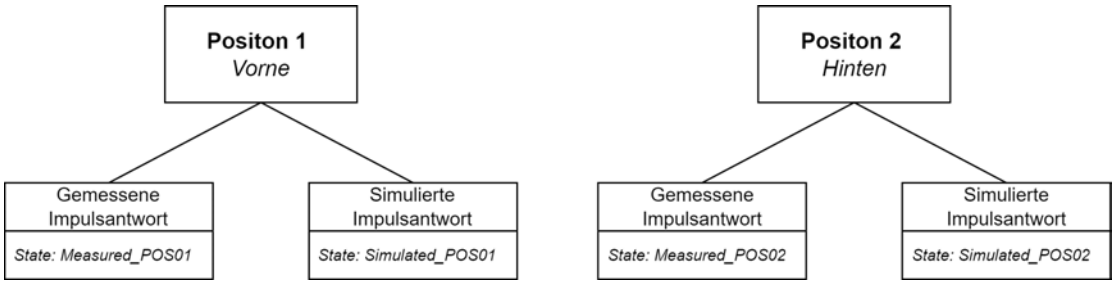


Abbildung 3.17: Aufstellung verschiedener States der Signalinfrastruktur.

Für die Implementierung eines Zustandswechsels sind vorerst die zwei wechselnden Parameter zu erörtern, die durch den Nutzer in der Menüführung beeinflusst werden. Die Auswahl der Position bestimmt die richtige Zuordnung der entsprechenden Impulsantworten. Die Erstellungsform der Impulsantwort wirkt sich ebenfalls durch die Auswahl des Nutzers auf den Zustand der nötigen Signalinfrastruktur aus. Im vorliegenden Fall sind jeweils zwei Positionen mit gemessenen und simulierten Impulsantworten vorhanden, weshalb sich letztendlich vier Zustände ergeben, zwischen denen gewechselt wird. Die Logik des Zustandswechsels kann nochmals über die genaue Benennung der States aus Abbildung 3.17 entnommen werden.
Die Beeinflussung der Zustandswechsel durch den Nutzer wird in Form von Dropdown-Menüs in die Menüführung integriert. Eines der Dropdown-Menüs gibt die Auswahl über gewünschte Hörpositionen. Ein zweites legt fest, ob die gehörten Inhalte mit der simulierten oder der gemessenen Impulsantwort gefaltet werden. Wie bereits vorhergehend beschrieben, geben Dropdown-Menüs in Unity für die Auswahl des Nutzers eine Zahl von null an aufwärts zurück. Dadurch lässt sich mit zwei Funktionen ein Aufrufen des richtigen Zustands realisieren. Die **InputHandleIR-Funktion** des Skripts legt einen Wahrheitswert für die Eingabe des Nutzers fest. Die Boolesche Variable *irMeasured* gibt so den Status der Impulsantwortherkunft für die Klasse des Skripts wieder.

```
1  public void InputHandleIR (int valu)
2  {
3          if (valu == 0)
4          {
5              irMeasured = false;
6              HandleInputData(_PosNumber);
7
8          }
9          else if (valu == 1)
10         {
11                 irMeasured = true;
12                 HandleInputData(_PosNumber);
13         }
14 }
```

Quelltext 3.4: Dropdown-Menüabfrage für die Bestimmung der Impulsantwortherkunft.

Zusätzlich wird eine zweite Funktion benötigt, die einen entsprechenden State unter Berücksichtigung der Positionsauswahl aufruft. Für die Bestimmung der derzeitigen Position des Spielerobjekts und der damit resultierenden Hörerposition wird die globale Variable *_PosNumber* eingeführt. Diese ist unter Abschnitt 3.5.3 weiter beschrieben. Für das Verständnis des Funktionsaufrufs ist relevant, dass diese Variable den tatsächlichen Aufenthaltsort des Spielerobjekts repräsentiert. Durch den in Quelltext 3.4 aufgeführten Funktionsaufruf der *HandleInputData*-Funktion wird die Position in Form der Variable mit übergeben.
Diese Funktion entscheidet anhand der übergebenen Variable über das Aufrufen eines passenden States. Für jede Position werden zwei Zustände nötig, die über eine klassische if-Entscheidung des Codes erstellt werden.

```
1  public void HandleInputData(int val)
2  {
3          if (val == 0 && irMeasured == false)
4          {
5              AkSoundEngine.SetState("EffectBypass", "Simulated_POS01");
6          }
7
8          else if (val == 1 && irMeasured == false)
9          {
10             AkSoundEngine.SetState("EffectBypass", "Simulated_POS02");
11         }
12 ...}
```

Quelltext 3.5: Funktion zur Änderung der aktiven Impulsantwort einer Position.

Durch den Quellcode 3.5 der Funktion ist nachzuvollziehen, wie der Aufruf des Zustands der Signalverarbeitung abläuft. Durch Nennung der State-Gruppe und dem gewünschten State wird die Signalverarbeitung entsprechend aufgerufen. Durch den Aufbau der Funktionen wird die Position des Spielerobjekts und die Einstellung der Herkunft der Impulsantwort berücksichtigt. Zusammenfassend transportieren die beiden Skripte **TeleportMenu** und **EventTriggerScript** die Interaktion aus Unity in die Wwise-Umgebung und ermöglichen so die Nutzung der hinterlegten Signalverarbeitung.

3.5 Aufbau einer Interaktionsoberfläche für Wwise und Unity

Die bereits in Kapitel 3.4.7 aufgegriffenen Schnittstellen für den Nutzer werden durch eine Integration in die Benutzeroberfläche nutzbar. Diese Schnittstelle dient der Steuerung von Optionen des Modellaufbaus, wie beispielsweise der bereits angesprochenen Umschaltung der Impulsantwortherkunft. Unity's Funktionsempfang erlaubt das Erstellen einer Benutzeroberfläche (engl. User Interface, kurz UI) mit Menüführung.

3.5.1 Anforderungen an die Benutzeroberfläche

Für die Implementierung der Menüführung in Unity wird vorerst das Diagramm 3.18 zur Beurteilung des nötigen Funktionsumfangs betrachtet. Die nötigen Funktionsfelder teilen sich in die bereits bekannten Dropdown-Menüs und in zu drückende Knöpfe (engl. Buttons) auf. Diese Elemente werden als UI-Elemente durch Unity bereitgestellt. Die zu verknüpfende Funktionalität kann der rechten Seite des folgenden Diagramms entnommen werden. Die linke Spalte zeigt

das Frontend der Menüführung mit allen notwendigen Optionen, die unter den Benennungen der Steuerungselemente aufgelistet werden. Für die Umsetzung der gewünschten Funktionalität

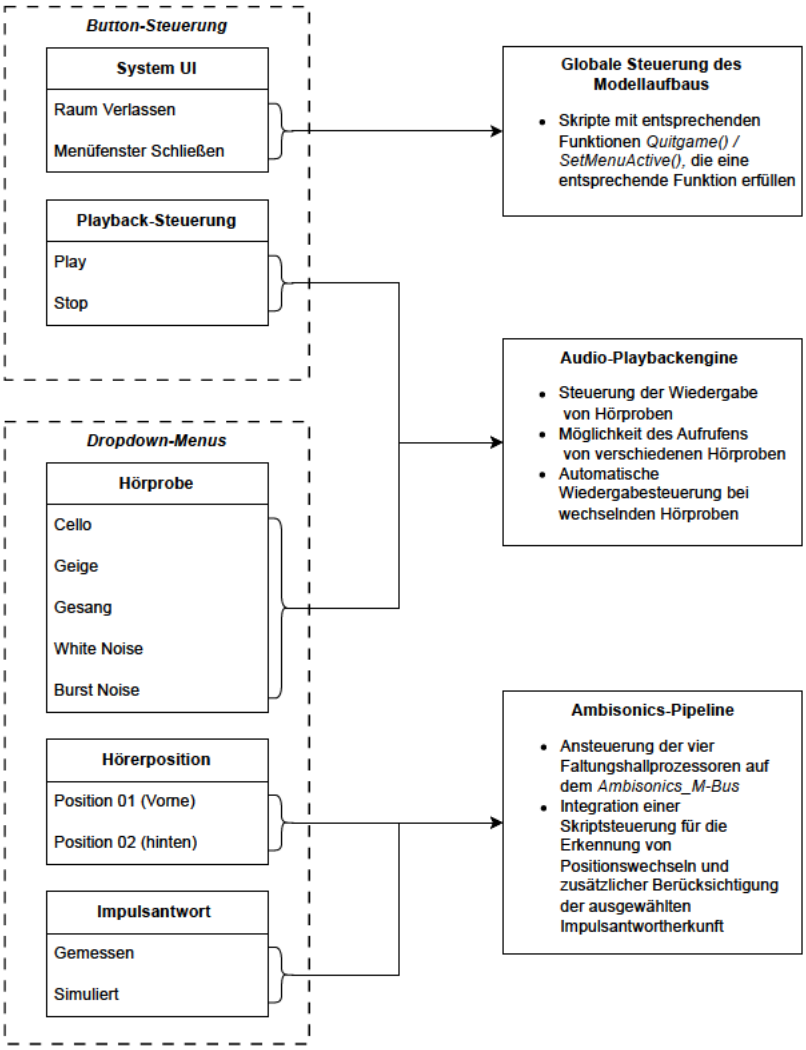


Abbildung 3.18: Verknüpfungen zwischen dem Frontend, der Menüführung und der Modellsteuerung.

wird in Unity ein sogenannter Canvas erstellt, auf dem Objekte der UI zu hinterlegen sind. Dazu zählen zum einen Text sowie Dropdown- und Knopfobjekte. Der grafische Aufbau der Oberfläche soll einen Zusammenhang der Hörbeispielsteuerung verdeutlichen. Eine Bedienung der Benutzeroberfläche soll außerdem keine Vorkenntnisse der Akustik oder VR verlangen, weshalb die Menüführung ausschließlich notwendige Funktionen darstellt.
Die Grafik 3.19 zeigt das Frontend der Planung von nötigen Interaktionsschnittstellen aus Diagramm 3.18. Die Ansteuerung der Menüführung erfolgt über die Controller des VR-Headsets. Diese emittieren einen Laserstrahl in der VR-Umgebung, um Inhalte des Menüs mit einem Knopfdruck des Controllers auszuwählen.

Abbildung 3.19: Benutzeroberfläche des Konzertsaals Blaibach.

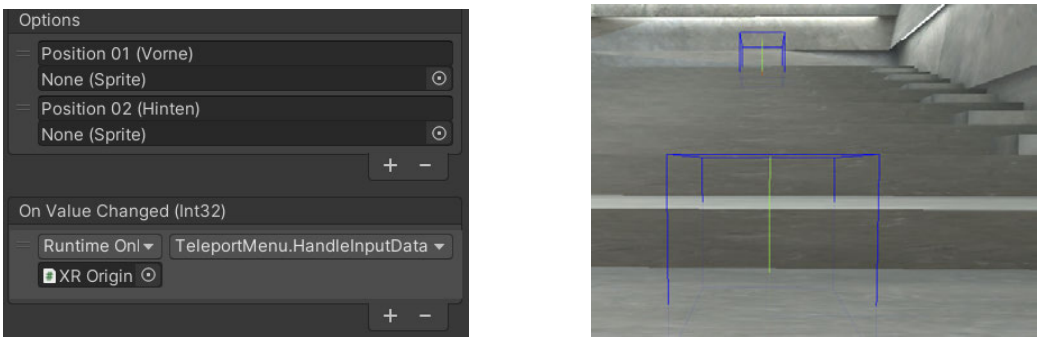
3.5.2 Implementierung der Benutzeroberfläche in die Entwicklungsumgebung

Unity baut eine Objekthierarchie der einzelnen Bestandteile des Menüs auf. Die Optionsauswahl eines Dropdown-Menüs wird über das Inspektorfenster Unitys angezeigt und die Möglichkeit zur Eingabe eigener Bezeichnungen geboten. Für die Interaktion durch den Benutzer kann durch das Öffnen eines Dropdown-Menüs die Anzahl der Optionen eingesehen werden. Ein visuelles Feedback in Form einer Farbüberblendung signalisiert die Auswahl einer Option nach dem Drücken des Controllers. Knöpfe der Wiedergabesteuerung sind farblich ebenfalls eindeutig hinterlegt und beeinflussen die Wiedergabe von ausgewählten Hörbeispielen.

3.5.3 Skriptsteuerung der Menüführung

Die Menüführung des Modellaufbaus wird durch die Erstellung von Skripten mit entsprechendem Funktionsumfang weiter in Unity implementiert. Ähnlich zu der von D'Amelio erstellten Steuerung aus [40], soll die Programmierung Zugriff auf die Zustandsänderung des Modellaufbaus bieten. Die Funktionsweise der beiden Dropdown-Menüs für die Steuerung der Impulsantwort und der Hörprobe wurden bereits durch Abschnitt 3.4.7 beschrieben. Für die generelle Verknüpfung von interaktiven UI-Elementen mit Funktionen der erstellten Skripte, ist eine Auswahl des Skripts im Inspektorfenster zu treffen. Das Verhalten des UI-Elements kann so bei einer eintretenden Interaktion des Nutzers festgelegt werden. Wie in Abbildung 3.20a erkennbar, kann unter dem Punkt *On Value Changed ()* die Funktion eines hinterlegten Skripts verknüpft werden.

Durch die beschriebene Auswahl wird die jeweilige Funktion durch die Auswahl des Nutzers aufgerufen. Je nach gewählter Option eines Dropdown-Menüs wird der Funktion ein ganzzahliger Wert weitergegeben, der von 0 ab aufwärts anhand der Optionsanordnung ausgegeben wird.
Für die Realisierung der Teleportation eines Spielerobjekts sind mehrere Komponenten für eine Implementierung in die Benutzeroberfläche nötig. Da die gemessenen Impulsantworten an bestimmten Koordinaten des Raummodells aufgenommen oder simuliert wurden, soll an diesen Positionen ein Abhören mit korrektem visuellem Eindruck ermöglicht werden. Für eine genaue Positionierung des Spielerobjekts werden deshalb bereits in Rhino die Messpunkte der Impulsantworten in Form von Sphären hinterlegt. Für das Hinterlegen von Komponenten wird ein Teleport-Ankerobjekt erstellt, das die Messpunktmarkierungen in Unity ersetzt. Auf dieses



(a) Inspektorfenster eines Dropdown-Menüs.

(b) Positionen der Teleport-Ankerobjekte im Konzertsaal.

Abbildung 3.20: Implementierung der Teleportationssteuerung.

Ankerobjekt wird ein Skript hinterlegt, dass durch einen sogenannten Collider überprüft, ob sich das Spielobjekt in einem Bereich um den Teleportanker aufhält. Dies wird durch folgenden Code realisiert.

```
1 public void OnTriggerEnter(Collider other)
2 {
3     if (other.gameObject.tag == "Player")
4     {
5         TeleportMenu._PosNumber = 0;
6     }
7 }
```

Quelltext 3.6: Änderung der Variable *_PosNumber*.

Durch einen *tag*, also eine namentliche Markierung die dem Spielerobjekt zugewiesen wurde, kann ein Eintritt desselben in dem festgelegten Bereich der Collider-Komponente registriert werden. Der bestimmende Bereich in Form eines Quaders ist in Abbildung 3.20b visualisiert. Registriert die Funktion *OnTriggerEnter* also den Eintritt des Spielerobjekts, wird dadurch die Variable *_PosNumber* auf eine 0 geändert. In diesem Fall ist dadurch die vordere Position der beiden möglichen aktiv. Auf dem zweiten Teleportanker befindet sich ein Skript, dass für die Variable eine 1 hinterlegt. Da die Variable an das Skript *TeleportMenu* weitergegeben wird, kann durch dieses klar differenziert werden, auf welcher Position das Spielerobjekt derzeit im Saal sitzt. Für die Übernahme der Variable wird diese global in der Klasse des *TeleportMenu*-Skripts deklariert. Die Entwicklungsumgebung ermöglicht das Verknüpfen von globalen Variablen und Funktionen über die Skripteinbindung des Inspektorfensters. Sobald eine globale Deklarierung erfolgt, kann durch eine Referenzierung auf das zugrunde liegende Game Object jede globale Variable oder Funktion in anderen Skripten aufgerufen werden. Ein Objekt erzeugt durch die Deklarierung eines globalen Game Objects eine Senke für ein anderes Objekt der Unityszene. Diesem Verfahren wird sich ebenfalls bei der Realisierung der Teleportsteuerung bedient. Anforderung an eine Teleportation ist die Bewegung des Spielerobjekts an die bestimmten Positionen der Teleportanker, die sich durch Messpunkte auf definierten Positionen befinden. Um die Position der Anker in die Teleportationssteuerung mit einzubeziehen, werden die Objekte in dem *TeleportMenu*-Skript referenziert.

```
1  public class TeleportMenu : MonoBehaviour
2  {
3          private Vector3 FrontPos;
4          private Vector3 MiddlePos;
5          public GameObject TeleportAnchor01;
6          public GameObject TeleportAnchor02;
7
8      void Start()
9      {
10         FrontPos = TeleportAnchor01.transform.position;
11         BackPos = TeleportAnchor02.transform.position;
12         gameObject.transform.position = FrontPos;
13         _PosNumber = 0;
14     }
15     public void HandleInputData(int val)
16     {
17         if (val == 0 && irMeasured == false)
18         {
19             gameObject.transform.position = FrontPos;
20             AkSoundEngine.SetState("EffectBypass", "Simulated_POS01");
21         }
22 ...}
```

Quelltext 3.7: Aufbau der Teleportationssteuerung.

Betrachtet man die Deklarierung von Objekten und Vektoren der Klasse *TeleportMenu* aus Quelltext 3.7, wird ersichtlich, dass die Position der beiden referenzierten Teleportanker als Vektoren in Form der Variablen *FrontPos* und *BackPos* in das Skript implementiert werden. Der Startvorgang der Anwendung weist automatisch den Vektoren die aktuelle Position der Ankerobjekte zu. Zusätzlich wird der Startzustand des Modellaufbaus festgelegt. Da das Spielerobjekt auf der vorderen Position in dem Modell starten soll, wird über Codezeile 12 des Quelltextes 3.7 die Teleportation auf die Position des vorderen Teleportankers bewerkstelligt. Die Variable *_PosNumber* ist mit einer 0 zu initialisieren, um keine ungewollte Teleportation beim Aufruf einer Option aus dem Hörbeispielmenü zu erzeugen. Die bereits bekannte *HandleInputData*-Funktion wird um Komponenten der Teleportation erweitert. Die Position des Teleportankers wird so durch die Auswahl des Teleportmenüs auf das Spielerobjekt angewandt. Dies hat den Vorteil, dass die Programmierung auch nach einer Veränderung der Position von Teleportankern die neuen Koordinaten nahtlos in die Teleportationssteuerung einpflegt. Für eine Anwendung in anderen Räumlichkeiten muss lediglich die Objekthierarchie der Teleportanker sowie das steuernde Skript übernommen werden.

Ähnlich zu der Funktion der Dropdown-Menüs werden die andrückbaren Knöpfe ebenso mit Skripten der Entwicklungsumgebung verknüpft. Die bereits bekannte Funktion *stopAll* unterbindet die globale Wiedergabe durch das Drücken des Stop-Knopfes. Das Starten der Wiedergabe über den Play-Knopf ruft hingegen die *playHead*-Funktion auf, wodurch nach Abfrage des ausgewählten Hörbeispiels die Auswahl wiedergegeben wird.

3.5.4 Globale Steuerungselemente der Benutzoberfläche

Für die Nutzung der vorhergehend beschriebenen Benutzeroberfläche bedarf es an Möglichkeiten die Menüführung aufzurufen und so auf den Funktionsumfang des Modellaufbaus zuzugreifen. Ein Ausblenden des Menüs trägt außerdem zu der Immersion des VR-Erlebnisses bei, da sich so besser auf den Raum und Klang konzentriert werden kann. Darum soll die Möglichkeit implementiert werden, das Menü über ein Pressen des sogenannten Thumbsticks auf den Controllern des VR-Headsets aufzurufen. Die Umsetzung benötigt die Bearbeitung der durch das XR Interaction Toolkit bereitgestellten Action Maps. Durch das Hinzufügen einer eigens erstellten Actionmap mit dem Namen *Menu* wird eine Action hinzugefügt, die das Drücken des Thumbsticks registriert. Der Aufbau der Action Map kann Abbildung 3.21 entnommen werden. Die Integration wird durch Quellcode 3.8 gezeigt. Durch die Vorarbeit kann die Action Map in

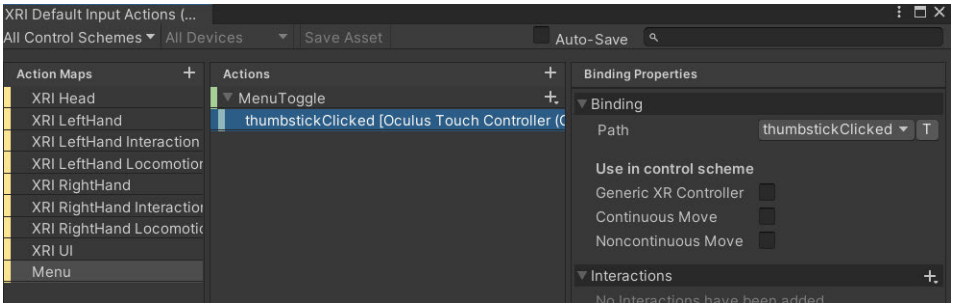


Abbildung 3.21: Action Map für den Druck des Thumbsticks der VR-Controller.

einem Skript referenziert werden. Das Skript *MainMenuLogic* beinhaltet diese Referenzierung und eine zugehörige Funktion *Toggle*, die den Canvas der Benutzeroberfläche nach dem Erkennen eines Tastendrucks auf dem Controller ein- oder ausblendet.

```
1  public class MainMenuLogic : MonoBehaviour
2  {
3          public InputActionReference toggleReference = null;
4          public GameObject Menu;
5
6      private void Toggle (InputAction.CallbackContext context)
7      {
8          bool isActive = !Menu.activeSelf;
9          Menu.SetActive(isActive);
10         MenuTeleport();
11     }
12 ...}
```

Quelltext 3.8: Aufbau der Teleportationssteuerung.

Der Canvas der Menüführung wird durch die Funktion *Toggle* in der Objekthierarchie aktiviert oder deaktiviert. Durch die Initialisierung der booleschen Variable *isActive* wird der derzeitige Zustand des Canvas-Objekts übergeben. Das Aufrufen der Funktion ändert diesen Wahrheitswert in den gegenteiligen Booleschen Wert. Dadurch lässt sich die Benutzeroberfläche per Knopfdruck des Controllers ausblenden und aufrufen.
Die VR-Umgebung bringt außerdem weitere Anforderungen an die Benutzeroberfläche mit sich. Da das Sichtfeld des Benutzers frei rotierbar ist, soll die Menüführung immer in der derzeit aktiven Blickrichtung erscheinen. Dies wird über die *MenuTeleport* Funktion umgesetzt. Das Aufrufen der Funktion richtet das Menü an der aktiven Blickrichtung des Benutzers aus.

```
1 private void MenuTeleport()
2 {
3     Menu.transform.position = head.position + new Vector3(head.forward.x,
      0, head.forward.z).normalized * spawnDistance;
4     Menu.transform.LookAt(new Vector3(head.position.x, Menu.transform.
      position.y, head.position.z));
5     Menu.transform.forward *= -1;
6 }
```

Quelltext 3.9: Menürotation in der VR-Umgebung.

Die Codezeile 3 aus Quelltext 3.9 realisiert die Teleportation des Menüobjekts in das derzeit aktive Blickfeld des VR-Headsets. Ein Vektor entnimmt die X- und Z-Koordinaten der Ausrichtung des Spielerobjekts. Da die X-Koordinate die Blickrichtung repräsentiert, wird durch die Multiplikation mit einer Variable *spawnDistance* der Abstand des Menüs zu dem Benutzer festgelegt. Die Y-Koordinate legt die Höhe von Objekten fest. Eine Eingabe von 0 ergibt ein Erscheinen auf Höhe des Spielerobjekts. Codezeile 4 rotiert die Menüführung parallel zu einer Tangente des sphärisch rotierenden Spielerobjekts. Der Berührungspunkt der Tangente gibt die derzeitige Blickrichtung wieder. Durch Codezeile 5 wird die Blickrichtung des Menüs korrigiert. Ein Aufruf der *MenuTeleport* Funktion bei einer Änderung der Position sowie bei dem Aufrufen der Benutzeroberfläche (s. Quelltext 3.8) führt zu einem korrekten Einblenden der Menüführung.

4 Diskussion

Die erstellte VR-Umgebung erfüllt die angesprochenen Kriterien und ermöglicht eine immersive Darstellung des Raummodells und der erhaltenen Simulations- sowie Messdaten. Der resultierende Modellaufbau wird auf Basis der Anforderungen mit der Verknüpfung einer VR-Entwicklungsumgebung zu einer entsprechend konfigurierbaren Audioengine geplant. Ein realistischer Höreindruck soll durch eine zu erstellende Audiosignalverarbeitung entstehen. Das Ziel einer Implementierung von akustischen Simulations- und Messdaten soll durch den Einsatz der Entwicklungsumgebung Unity und der externen Audioengine Wwise umgesetzt werden. Die Fragestellung der Arbeit verlangt das Aufzeigen einer Möglichkeit der audiovisuellen Betrachtung von akustischen Gegebenheiten in der VR. Die Beantwortung der Forschungsfrage ist durch die Beschreibung des Modellaufbaus in Kapitel 3.4.7 gegeben.

Für die Durchführung der Verknüpfung der beschriebenen Softwarekomponenten wurde vorerst eine detaillierte Recherche zur Aufdeckung von Schnittstellen durchgeführt. Verschiedene Szenarien, wie der Einsatz einer Echtzeit-Auralisation sowie das Einbringen von bereits gefalteten Audiodateien wurden abgewogen, um ein plausibles Hörerlebnis zu generieren. Der letztendliche Aufbau der Signalinfrastruktur beruht auf bekannten Verfahren und beinhaltet Simulationsdaten, die ohne hörbare Artefakte durch die Raumakustikabteilung der Müller-BBM BSO erstellt wurden.

Die Ergebnisse des aufgebauten Frameworks zeigen eine erfolgreiche Umsetzung der gestellten Anforderungen an die audiovisuelle Darstellung in der VR. Wwise stellt die Ambisonicsszene mit der gewünschten Rotation durch eine Kopfbewegung dar. Die Umschaltung zwischen Effektprozessoren ist durch die Funktionalität der State-Gruppen nutzbar. So können der Positionswechsel sowie die Umschaltung zwischen simulierten und gemessenen Impulsantworten richtig wiedergegeben werden. Das Abspielen der eingebrachten Hörbeispiele ist auf Grundlage der Audio-Eventsteuerung nutzbar. Für eine volle Immersion des Raumes kann die Menüführung ausgeblendet werden. Die Audiowiedergabe wird trotzdem nahtlos fortgesetzt.
Unity stellt die nötigen Schnittstellen für das Importieren eines Raummodells bereit und lässt optische Optimierungen zu. Die skriptbasierte Menüführung steuert über Befehlsstrukturen und die Abfrage von Wahrheitswerten den Modellaufbau. Die Umsetzung der Programmierung funktioniert ohne weitere Einschränkungen für das genutzte VR Headset und den damit angebundenen Kopfhörern. Die VR-Brille Meta Quest 2 bietet durch den integrierten sogenannten Quest Link eine Grundlage für das Debugging der zu erstellenden VR-Umgebung und stellt die Inhalte der Entwicklungsumgebung audiovisuell ansprechend dar.

Die Realisierung des Modellaufbaus zeigt eine erfolgreiche Umsetzung der gestellten Ziele. Da besonders die Audioinfrastruktur ausschlaggebend für eine korrekte Darstellung der Schallfeldresynthese ist, werden Möglichkeiten der Verbesserung im folgenden Teil der Arbeit diskutiert.

Der Modellaufbau baut auf einem Faltungsprozess und der Ausgabe eines Ambisonicsformats zweiter Ordnung auf. Die zweite Ordnung bietet eine Steigerung der hörbaren räumlichen Auflösung der Audioinhalte im Vergleich zu der ersten Ordnung. Das in dieser Form synthetisierte Schallfeld reicht für eine plausible Darstellung einer akustischen Szene aus. Jedoch lässt sich

durch die Steigerung der Ordnung eine Schärfung der räumlich hörbaren Auflösung erreichen. Wwise unterstützt eine Signalform des Ambisonicsformats bis zu der fünften Ordnung. Für die dadurch entstehende Verbesserung der akustischen Szene ändern sich die Anforderungen an die eingebrachten Simulations- und Messdaten.

Die Platzierung des Nutzers in der virtuellen Umgebung wird über das Spielerobjekt in Unity gelöst. Dieses Objekt kann zwischen zwei festgelegten Positionen teleportiert werden. An diesen Koordinaten des Raumes wurden die entsprechenden Empfänger und Messsonden der erhaltenen Impulsantworten platziert. Da bei einer Verschiebung des Spielerobjekts auf andere Punkte eine Verfälschung zwischen visueller Darstellung und hörbarer Inhalte aufkommt, muss das Spielerobjekt fest positioniert werden. Durch eine Bewegung des Nutzers in der realen Welt entsteht also keine Verschiebung des Spielerobjekts. Diese Limitation bietet dafür eine audiovisuell plausible Darstellung an den beiden festgelegten Positionen. Durch eine freie Beweglichkeit des Spielerobjekts würde die Immersion für den Anwender weiter gesteigert werden. Zusätzlich könnten so die unterschiedlichsten Positionen akustisch demonstriert werden. Diese freie Beweglichkeit setzt eine Überblendung zwischen unterschiedlichen Impulsantworten oder die akustische Simulation in Echtzeit voraus.

Die Signalinfrastruktur basiert auf einer Binauralisierung des Ambisonicsbusses. Der Binauralisierungsprozess wird über die Faltung und Summation einer HRTF-Datenbank realisiert. Das dadurch entstehende räumliche Hören funktioniert je nach Ohrform des Nutzers unterschiedlich gut. Eine stark abweichende Außenohrform führt zu einer Verschiebung der durch Grafik 2.6 aufgezeigten Blauert'schen Bänder. Dies führt zu einer verfälschenden Lokalisation. Um dieser Problematik entgegenzuwirken, bietet sich das Abhören über ein entsprechendes Lautsprecher-Surroundarray an. Für die Wiedergabe von Hörbeispielen werden Audiodateien aus Bibliotheken trockener Instrumentalaufnahmen entnommen. Die dafür eingesetzte Lautstärke, mit der eine Audiodatei letztendlich auf den Kopfhörern des Benutzers hörbar wird, ist durch eine gemittelte Lautheitsbewertung durchgeführt worden. Um das Hörerlebnis noch realer zu gestalten, fehlt ein Referenzpegel der realen Schallquelle.

Die erfolgreiche Erstellung einer VR-Umgebung auf Basis von akustischen Simulations- und Messdaten unter Verwendung von Unity und Wwise zeigt eine weitere Möglichkeit der akustischen Betrachtung einer Räumlichkeit. Die Kombination eines plausiblen Höreindrucks sowie die ansprechende Darstellung von visuellen Inhalten durch das VR-Headset ergeben eine repräsentative Darstellungsform. Die vorhergehende Diskussion zeigt das noch auszuschöpfende Potenzial zu einer Verbesserung der Audiosignalverarbeitung. Ansätze für eine Verbesserung der angesprochenen Problematiken sind dem Ausblick zu entnehmen.

5 Fazit

Der im vorhergehenden Kapitel beschriebene Modellaufbau erfüllt das Ziel der immersiven Demonstration akustischer Eigenschaften. Der in Abschnitt 3.1 aufgeführte Überblick von einzelnen Software- und Hardwarekomponenten gibt Aufschluss über das zu erstellende Framework des Modellaufbaus. Komponenten wie ein VR-Headset gilt es mit einer Entwicklungsumgebung zu verknüpfen. Eine Vorarbeit in Form einer Simulation und Messung liefert die notwendigen Daten zur Rekonstruktion akustischer Gegebenheiten. Das Modell des Saal Blaibach muss vorerst in Rhino und anschließend in der Entwicklungsumgebung Unity aufbereitet werden. Rhino liefert notwendige Funktionen zur Wandlung des Modells in eine bearbeitbare Version für Unity. In Unity selbst werden zusätzlich Optimierungen vorgenommen. Texturen sorgen auf den Raumbegrenzungsflächen für einen realistischeren Raumeindruck. Die zusätzlich integrierte Beleuchtung mit erstellter *Baked Light Map* erzeugt eine deutlichere Wahrnehmung der Betonstruktur des Raumes und vermittelt dadurch eine entsprechende Raumtiefe. Ein Klavier-Flügel-Modell gibt dem Benutzer ebenso eine Bezugsgröße des Raumumfangs.

Die Aufstellung der zu hinterlegenden Audioinfrastruktur wird durch die Audioengine Wwise bewerkstelligt, die einen Faltungsprozess sowie das Ambisonicsformat unterstützt. Das Rückgrat der Signalverarbeitung, die Mixbus-Struktur bildet Grundlage für die Signaltransformation. Ein trockenes Monosignal wird über die richtige Konfiguration von Bussen in eine Ambisonicsszene zweiter Ordnung gewandelt. Die dafür eingesetzten Signalprozessoren falten mit den durch Simulation und Messung erhaltenen Impulsantworten. Für die Einsparung von Rechenleistung wird das aktive Fenster der jeweiligen Impulsantwort beschnitten. Das dadurch entstehende Ambisonics Format zweiter Ordnung wird in einem nachfolgenden Prozess binauralisiert. Das dafür benötigte Plug-In von Google Resonance liefert die notwendige Datenbank an gemessenen und gemittelten HRTF's. Durch die Faltung und Summation mit diesen Funktionen entsteht ein enkodiertes binaurales Signal. Die Rotation der Ambisonicsszene wird in diesem Fall durch die Wwise-Implementierung in Unity berücksichtigt. Durch Platzierung von Skripten und Komponenten, wie durch 3.4.7 beschrieben, werden Rotation und Position des Spielerobjekts an Wwise weitergegeben. Die Rotation beeinflusst anschließend die Ausrichtung der Ambisonics-Szene vor dem Binauralisierungsprozess. Die Implementierung einer Zustandssteuerung ermöglicht das Umschalten zwischen den vier notwendigen Effektprozessoren, die seriell in die Bus-Struktur geschaltet werden. Eine eingebrachte Audio-Eventsteuerung realisiert das Abspielen von Hörbeispielen.

Die Menüführung des Modellaufbaus dient als Schnittstelle zwischen Nutzer und Programmierung. Die in Abbildung 3.19 gezeigte Benutzeroberfläche ermöglicht Zugriff auf die Zustandssteuerung der Effektprozessoren, wodurch je nach Auswahl der gewünschten Position sowie Impulsantwortherkunft, der entsprechende Effektprozessor in der Signalinfrastruktur aktiv ist. Die Auswahl kann in Form eines Dropdown-Menüs getroffen werden. Die Auswahl eines Hörbeispiels startet die Wiedergabe. Ein Start- und Stop-Knopf bieten zusätzliche Kontrolle über die Audiowiedergabe. Globale Funktionen, wie das Ausblenden des Menüs oder das Verlassen des Modellaufbaus ist durch entsprechende Knöpfe hinterlegt. Hervorzuheben ist die Positionsauswahl durch den Spieler. Das Teleportieren des Spielerobjekts zwischen den beiden Teleport-Ankern

ist durch entsprechenden Code so hinterlegt, dass eine Bewegung des Teleport-Ankers ab dem nächsten Teleportvorgang weiter berücksichtigt wird. So kann die Programmierung ohne weitere Änderungen des Programmcodes für andere Raummodelle angewandt werden.

Der erreichte Modellaufbau integriert die von der Müller-BBM BSO gestellten Anforderungen. Ein Einsatz für zu planende Projekte ist an verschiedenen Stellen denkbar. Die immersive Betrachtung einer Räumlichkeit in der VR bietet sowohl Vorteile für erfahrene Akustik-Ingenieure als auch für andere Beteiligte. Der Modellaufbau ermöglicht einem Akustikingenieur die Konzentration auf die audiovisuelle Darstellung von Simulationsdaten und zugehörigem Raummodell. Hörbare Inhalte basieren auf bekannten Verfahren der Signalverarbeitung, wodurch ein realistischer Höreindruck gewährleistet wird. Die Bewegung des Kopfes und die dadurch rotierende Ambisonicsszene vermitteln eine Annäherung an ein realistisches Schallfeld. Die zugrunde liegende Anwendungsmechanik lässt sich problemlos auf andere Raummodelle übertragen, wodurch auch ein Präsentationsszenario durch die Darstellung in VR denkbar wird. Sowohl für Kunden als auch Firmenangehörige kann so ein Raum aus der Zuschauerperspektive betrachtet werden. Das Auswerten von Graphen und Simulationsdaten bleibt weiter als Notwendigkeit der Planungspraxis bestehen. Ein simuliertes Schallfeld kann jedoch durch die Darstellung des Modellaufbaus anders erlebt werden.

Die Umsetzungsphase des Modellaufbaus zeigte zusätzlich auf, dass die Gaming Engine Unity eine gute Grundlage für die Entwicklung einer VR-Umgebung mit den bekannten Anforderungen bietet. Die Anwendung funktioniert nur mit einer zu kaufenden Lizenz. Wwise erweitert die Audiofunktionalität um den nötigen Faltungsprozess sowie die Ambisonicsformat-Unterstützung. Wwise ist grundsätzlich kostenlos nutzbar, die einzelnen Effekte wie der Wwise Convolution Reverb müssen in Form einer Lizenz erworben werden. Für Studenten sind Unity und Wwise zu Nachforschungszwecken kostenlos nutzbar.

Die Arbeit befasste sich mit dem Aufzeigen und Erarbeiten einer Darstellungsmöglichkeit akustischer Inhalte in der VR und zeigt lediglich eine Möglichkeit der Implementierung auf, beinhaltet keine Wertung der einzelnen verknüpften Komponenten und schließt andere Ansätze nicht aus.

5.1 Ausblick

Ein Einsatz von akustischen Simulationsdaten in der virtuellen Realität bietet Potenzial für zukünftige Entwicklungen für die Akustikplanung. Die VR ist ein Themengebiet, das durch akustisch Forschende immer wieder in Verbindung mit einer Auralisation genannt wird. Vorländer spricht in [3] bereits 2009 von einer akustischen virtuellen Realität. Er benennt Probleme wie eine Auralisierung in Echtzeit. Aus der akademischen Community werden immer mehr Quellen über eine Simulation und Auralisation in Echtzeit bekannt. Projekte wie das Auralisations-Framework *Virtual Acoustics* der RWTH Aachen finden Anwendung in den Entwicklungsumgebungen Unity oder der Unreal Engine. Das Framework bietet so die Möglichkeit für Entwickler akustische Phänomene wie Absorption, Reflexion, Beugung und den Dopplereffekt zu modellieren. Die Berechnung der Simulation von Schallverteilung erfolgt in Echtzeit und wird von Materialien auf den Objekten der Entwicklungsumgebung beeinflusst. Durch die Echtzeitanwendung lässt sich die Beweglichkeit des Spielerobjekts uneingeschränkt von Messungspunkten erweitern. Die Anwendung einer Echtzeit-Auralisation setzt jedoch das Hinterlegen akustischer Materialien im

Raummodell voraus.
Der beschriebene Modellaufbau baut auf einer im Vorhinein durchgeführten Simulation auf, um so auf Basis bekannter Formate wie Ambisonics, und der durch Müller BBM durchgeführten Simulation in Odeon einen plausiblen Höreindruck zu generieren. Für eine künftige Integration einer Echtzeit-Simulation in dem beschriebenen Modellaufbau, stellt sich die Frage nach einer sinnvollen Integration, wenn das durch diese Arbeit beschriebene Grundprinzip von Wwise und Unity weiter zu nutzen ist. Deshalb werden im folgenden letzten Abschnitt der Arbeit Ansätze für eine zukünftige Optimierung des Modellaufbaus genannt.

Für eine weitere Optimierung der Abhörsituation würde sich das Einbringen eines Lautsprecher-Surroundarrays anbieten. Dieses muss den Vorgaben des genutzten Dekodierungsprozesses entsprechen. Die Ausgabe eines Audiosignals im Ambisonicsformat setzt die entsprechende Konfiguration auf dem Master Audio Output in Wwise voraus. Das mehrkanalige Signal kann durch einen Audiotreiber entgegengenommen werden, um eine Verbindung zu einem entsprechenden Dekodierer herzustellen. Dieser sorgt für eine Verteilung der Ambisonicsszene auf die einzelnen Lautsprecher. So wird die bereits besprochene Problematik der Binauralisisierung umgangen.

Für die zukünftige Benutzung des beschriebenen Modells ist außerdem eine Methodik zu entwickeln, die Wiedergabelautstärke an die realen Lautstärken der Hörbeispiele anzugleichen. Hierfür bieten sich eigens angefertigte Aufnahmen der Hörbeispiele in einem trockenen Raum an. Während der Aufnahme wird der resultierende Lautstärkepegel gemessen. Durch eine entsprechende Messeinrichtung soll der resultierende Lautstärkepegel angepasst und an die realen Schallquellen angeglichen werden. So kann ein realer, vergleichbarer Höreindruck zwischen unterschiedlichen Hörbeispielen gebildet werden.

Für eine Verbesserung der räumlichen Auflösung bei einer Anwendung des Modellversuchs für andere Räumlichkeiten würde sich die Erhöhung der Ordnung des Ambisonicssignals empfehlen. Dafür muss das entsprechende Messmikrofon die Ausgabe eines Ambisonicsformats fünfter Ordnung unterstützen. Die simulierten Impulsantworten können von Odeon bis zur dritten Ordnung des Ambisonicsformats ausgegeben werden. Für die Umwandlung in die fünfte Ordnung existieren Methodiken der Verbesserung von Richtungsauflösung der Ambisonicsszene.

Die aufgelisteten Optimierungsmöglichkeiten des Ausblicks bieten Grundlage für die künftige Forschung an einzelnen Komponenten des Modellaufbaus.

Literaturverzeichnis

[1] *Apple Vision Pro*. Adresse: https://www.apple.com/apple-vision-pro/ (besucht am 22. 06. 2023).

[2] L. Cremer und H. A. Müller, *Die wissenschaftlichen Grundlagen der Raumakustik: Geometrische Raumakustik, statistische Raumakustik, Psychologische Raumakustik*. Stuttgart: S. Hirzel Verlag, 1978, Bd. 1.

[3] M. Vorländer, „Auralization of spaces“, *Physics Today*, Jg. 62, Nr. 6, S. 35–40, Juni 2009. DOI: 10.1063/1.3156831.

[4] *ODEON's calculation methods*. Adresse: https://odeon.dk/learn/video-tutorials/calculation-methods/ (besucht am 15. 03. 2023).

[5] *Diffusion series*. Adresse: https://www.catt.se/diffseries/index.htm (besucht am 03. 05. 2023).

[6] V. Stauskis, „THE NEAR AND FAR ACOUSTIC FIELD AND ITS RELATIONSHIP TO THE HALL ACOUSTICS“, *Statyba*, Jg. 2, Nr. 6, S. 59–67, Jan. 1996. DOI: 10.1080/13921525.1996.10531645.

[7] N. Xiang und J. Blauert, *Acoustics for Engineers: Troy Lectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021. DOI: 10.1007/978-3-662-63342-7.

[8] T. D. Rossing und N. H. Fletcher, *Principles of Vibration and Sound*. New York: Springer New York, 2004. DOI: 10.1007/978-1-4757-3822-3.

[9] K. Ikeuchi, Hrsg., *Computer Vision: A Reference Guide*. Boston, MA: Springer US, 2014. DOI: 10.1007/978-0-387-31439-6.

[10] M. Vorländer, *Auralization: Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality* (RWTHedition). Cham: Springer International Publishing, 2020. DOI: 10.1007/978-3-030-51202-6.

[11] M. R. Schroeder, „New Method of Measuring Reverberation Time“, *The Journal of the Acoustical Society of America*, S.409–412, 1964.

[12] E. A. Lehmann, A. M. Johansson und S. Nordholm, „Reverberation-Time Prediction Method for Room Impulse Responses Simulated with the Image-Source Model“, in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Okt. 2007, S. 159–162. DOI: 10.1109/ASPAA.2007.4392980.

[13] J. Warner, „Reflection coefficients and Mapping“, 2013.

[14] J. B. Allen und D. A. Berkley, „Image Method for Efficiently Simulating Small-room Acoustics“, *The Journal of the Acoustical Society of America*, Jg. 65, Nr. 4, 1978.

[15] I. Bork, „Auralisation bei tiefen Frequenzen“, 2006.

[16] M. Möser, Hrsg., *Messtechnik der Akustik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-540-68087-1.

[17] *Odeon Users's Manual*. Adresse: https://odeon.dk/downloads/user-manual/ (besucht am 04. 05. 2023).

[18] B. Girod, R. Rabenstein und A. Stenger, *Einführung in die Systemtheorie*. Wiesbaden: Vieweg+Teubner Verlag, 2003. DOI: 10.1007/978-3-322-99346-5.

[19] O. Gazi, *Understanding Digital Signal Processing* (Springer Topics in Signal Processing). Singapore: Springer Singapore, 2018, Bd. 13. DOI: 10.1007/978-981-10-4962-0.

[20] F. Adriaensen, „LAC2006 Proceedings“, Apr. 2006.

[21] J. Blauert und J. Braasch, „Räumliches Hören“, in *Handbuch der Audiotechnik*, S. Weinzierl, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 87–121. DOI: 10.1007/978-3-540-34301-1_3.

[22] K. Iida, *Head-Related Transfer Function and Acoustic Virtual Reality*. Singapore: Springer Singapore, 2019. DOI: 10.1007/978-981-13-9745-5.

[23] M. Morimoto und Y. Ando, „On the simulation of sound localization.“, *Journal of the Acoustical Society of Japan (E)*, Jg. 1, Nr. 3, S. 167–174, 1980. DOI: 10.1250/ast.1.167.

[24] W. Ellermeier und J. Hellbrück, „Hören – Psychoakustik – Audiologie“, in *Handbuch der Audiotechnik*, S. Weinzierl, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 41–85. DOI: 10.1007/978-3-540-34301-1_2.

[25] D. Stotz, *Computergestützte Audio- und Videotechnik: Multimediatechnik in der Anwendung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019. DOI: 10.1007/978-3-662-58873-4.

[26] Mellert, K. F. Siebrasse und S. Mehgradt, „Determination of the transfer function of the external ear by an impulse response measurement“, 1974. DOI: 10.1121/1.1903534.

[27] „Head-Related Transfer Function (HRTF) in Audition“, in *Encyclopedia of Neuroscience*, M. D. Binder, N. Hirokawa und U. Windhorst, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 1812–1812. DOI: 10.1007/978-3-540-29678-2_2150.

[28] L. Biscainho, P. Diniz und F. Freeland, „Efficient HRTF Interpolation in 3D Moving Sound“, Mai 2002.

[29] B. Gardner und K. Martin, „HRTF Measurements of a KEMAR Dummy-Head Microphone“, 1994.

[30] B. Rafaely u. a., „Spatial audio signal processing for binaural reproduction of recorded acoustic scenes – review and challenges“, *Acta Acustica*, Jg. 6, S. 47, 2022. DOI: 10.1051/aacus/2022040.

[31] F. Zotter und M. Frank, *Ambisonics: A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality* (Springer Topics in Signal Processing). Cham: Springer International Publishing, 2019, Bd. 19. DOI: 10.1007/978-3-030-17207-7.

[32] *The Michael Gerzon Story*. Adresse: https://intothesoundfield.music.ox.ac.uk/michael-gerzon-story (besucht am 20. 04. 2023).

[33] M. Kronlachner und F. Zotter, *Spatial transformations for the enhancement of Ambisonic recordings*. Jan. 2014.

[34] M. Gerzon, *Practical Periphony: the reproduction of full sphere sound*, 1980.

[35] W. Ritsch, M. Chapman, I. Zmölnig und F. Zotter, „A Standard for Interchange of Ambisonic Signal Sets. Including a file standard with metadata.“, 2009.

[36] E. M. Hoffbauer, „Evaluation of Surround Sound Setups based on Ambisonic Room Impulse Response Measurements“, 2021.

[37] C. Nachbar, F. Zotter, E. Deleflie und A. Sontacchi, „AMBIX - A SUGGESTED AMBISONICS FORMAT“,

[38] D. Malham, *Higher order Ambisonics systems, abstracted from "Space in Music - Music in Space"*. 2003.

[39] M. Mcloughlin, „Overview of Virtual Ambisonic Systems“, in *Encyclopedia of Computer Graphics and Games*, N. Lee, Hrsg., Cham: Springer International Publishing, 2018, S. 1–4. DOI: 10.1007/978-3-319-08234-9_275-1.

[40] F. D'Amelio, *Blog | Audiokinetic*. Adresse: https://blog.audiokinetic.com/en/virtual-acoustic-reality-in-architectural-design/ (besucht am 13. 01. 2023).

[41] *ARD-Themenwoche "Stadt.Land.Wandel": Das Kultur-Ufo in Blaibach | BR-Klassik*, Nov. 2021. Adresse: https://www.br-klassik.de/aktuell/news-kritik/ard-themenwoche-stadt-land-wandel-konzerthaus-blaibach-100.html (besucht am 19. 06. 2023).

[42] H. G. Paul, *Blaibach: "Stornierungswellen werden auf dem Rücken der Musiker ausgetragen"| Da Hog'n - Onlinemagazin ausm Woid*. Adresse: https://www.hogn.de/2020/10/23/2-kultur-im-bayerischen-wald/1-ausm-woid/blaibach-stornierungswellen-werden-auf-dem-ruecken-der-musiker-ausgetragen/141129 (besucht am 19. 06. 2023).

[43] F. G. Weiland, *Home*. Adresse: https://www.peterhaimerl.com (besucht am 06. 06. 2023).

[44] *Schallreflexion: diffuses Klangbild im Raum*. Adresse: https://www.akustikform.ch/blog/reflexion-schall (besucht am 10. 06. 2023).

[45] R. M. Associates, *Rhinoceros 3D*. Adresse: https://www.rhino3d.com/ (besucht am 19. 06. 2023).

[46] *FBX - File Extension*. Adresse: https://fileinfo.com/extension/fbx (besucht am 10. 06. 2023).

[47] *Meta Quest 2: Immersives all-in-one VR-Headset*. Adresse: https://www.meta.com/de/quest/products/quest-2/ (besucht am 12. 06. 2023).

[48] L. O'Reilly, *Facebook acquires virtual reality headset maker Oculus Rift for $2bn*, März 2014. Adresse: https://www.marketingweek.com/facebook-acquires-virtual-reality-headset-maker-oculus-rift-for-2bn/ (besucht am 19. 06. 2023).

[49] *Wwise | Audiokinetic*. Adresse: https://www.audiokinetic.com/en/products/wwise/ (besucht am 20. 06. 2023).

[50] *Resonance Audio - Discover Resonance Audio*. Adresse: https://resonance-audio.github.io/resonance-audio/discover/overview.html (besucht am 20. 06. 2023).

[51] *Resonance Audio - Wwise*. Adresse: https://resonance-audio.github.io/resonance-audio/develop/wwise/getting-started.html (besucht am 20. 06. 2023).

[52] *OpenAIR – Open Air Library*. Adresse: https://www.openair.hosted.york.ac.uk/ (besucht am 06. 06. 2023).

[53] *MTG - Music Technology Group - PHENICX-Anechoic: note annotations for Aalto anechoic orchestral database*. Adresse: https://www.upf.edu/web/mtg/phenicx-anechoic (besucht am 06. 06. 2023).

Anhang A: Nachhallzeit des Konzerthauses Blaibach

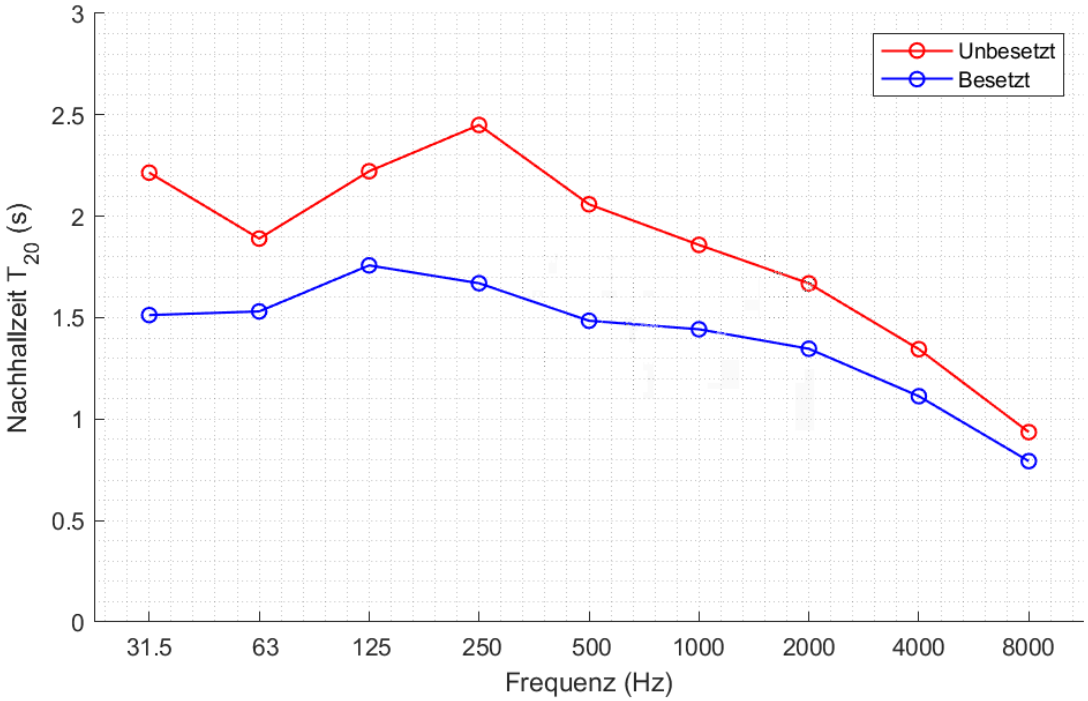


Abbildung A.1: Gemessene Nachhallzeit T_{20} des besetzten und unbesetzten Konzertsaals.

Anhang B: Ansichten des Raummodells aus Rhino

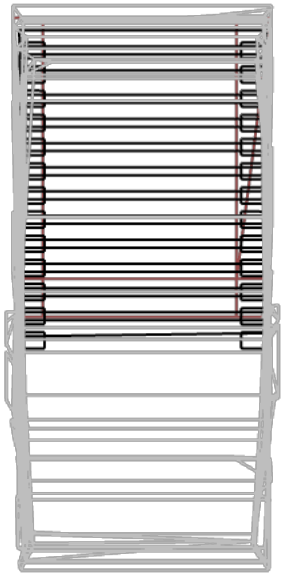


Abbildung B.1: Ansicht aus der Vogelperspektive.

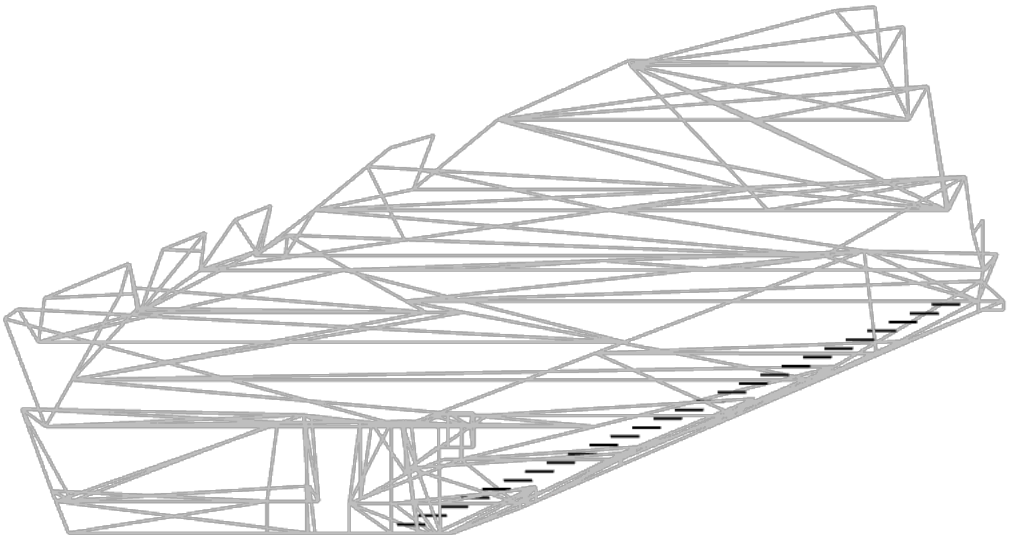


Abbildung B.2: Seitliche Ansicht des Modells.

Eidesstattliche Erklärung

Hiermit versichere ich – Vincent Ritter – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die
anderer Autoren entnommen sin

Diese Arbeit wurde in gleicher
vorgelegt oder anderweitig veröfl

Garmisch-Partenkirchen, 03. Juli

Ort, Datum

Vincent Ritter