

Evaluation von Akustik-Frameworks in Bezug auf Lokalisierung von Schallquellen

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Lucas Hilbig

Erstgutachter: Prof. Dr. Stefan Müller
Institut für Computervisualistik/Computergrafik

Zweitgutachter: M.Sc. Alexander Maximilian Nilles
Institut für Computervisualistik/Computergrafik

Koblenz, im November 2021

Erklärung

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

Ja Nein

Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.

Koblenz, 20.11.2021
.....
(Ort, Datum)

L. Hilbig
.....
(Unterschrift)

Zusammenfassung

In dieser Arbeit wird untersucht, ob ein realistisches Audio-Rendering, bei welchem Schalleffekte wie Verdeckung oder Hall simuliert werden, auch ein Vorteil zur Lokalisierung von Schallquellen liefert. Die kommerziellen Frameworks *Steam Audio* und *Project Acoustics* werden als Vertreter von realistischem Audio-Rendering gegen *FMOD* verglichen, welches simples Panning ohne zusätzliche Effekte nutzt. Zusätzlich wurde ein auf Graphen basierendes Framework *Graph Audio* selbst implementiert. Auch dieses simuliert zusätzliche Schalleffekte und wird mit den drei anderen Frameworks verglichen. Die aufgestellte Leitfrage, ob eine realistischere Schallausbreitung auch in einer erhöhten Lokalisierbarkeit von Schallquellen resultiert, wird Anhand einer im Rahmen dieser Arbeit durchgeführten Evaluation mit Ja beantwortet. Für diese Evaluation wurde eine Art Minispiel entwickelt, in welchem die Probanden unsichtbare Schallquellen finden mussten. Zusätzlich sollten sie dazu einen Fragebogen ausfüllen.

Inhaltsverzeichnis

1 Einleitung	1
2 Grundlagen	2
2.1 Schall	2
2.2 Räumliche Schallsimulation	5
2.3 Realistische Schallsimulation	6
3 Verwandte Arbeiten	8
4 Audio-Frameworks für räumlichen Schall	9
4.1 FMOD	9
4.1.1 Funktionsweise	9
4.2 Steam Audio	11
4.2.1 Funktionsweise	12
4.3 Project Acoustics	13
4.3.1 Funktionsweise	14
4.3.2 Vor- und Nachteile	16
4.4 Graph-Based Real-Time Spatial Sound Framework	16
4.4.1 Funktionsweise	17
4.4.2 Vor- und Nachteile	21
5 Konzeption Graph Audio	22
5.1 Graph-Erstellung	23
5.2 Mögliche Erweiterung	25
6 Implementation Graph Audio	26
6.1 Baking	27
6.2 Laufzeit	29
7 Evaluation von Akustik-Frameworks in Bezug auf Lokalisierung von Schallquellen	32
7.1 Aufbau Testumgebung	33
7.2 Aufbau Fragebogen	34
7.3 Testablauf	35
7.4 Ergebnisse	36
8 Fazit	41
8.1 Graph Audio	41
8.2 Evaluation und Leitfrage	42
8.3 Zukunftsausblick	42

1 Einleitung

Das Audiodesign ist eine wichtige Komponente beim Entwickeln von Videospielen. Denn der Sound ist ein wichtiger Teil von verschiedenen Aspekten eines Videospiels. Er hat großen Einfluss auf die im Spiel erzeugte Stimmung, was meist im Zusammenhang mit der genutzten Musik steht. Aber nicht nur Musik, sondern auch jegliche andere Schallquelle in einem Spiel, wie Fußstapfen oder fahrende Autos, spielen eine große Rolle. Damit diese zur Immersion einer virtuellen Welt beitragen können, ist es wichtig, dass sie einen räumlichen Klang besitzen. Sieht ein Spieler beispielsweise links von sich auf einem Tisch einen angeschalteten Fernseher stehen, dann soll der Schall auch von der linken Seite kommen. Dieses Lokalisieren von Schallquellen ist besonders im First-/Third-Person-Shooter-Genre sehr wichtig. Denn bei Spielen dieser Genres stellt es neben der Immersion auch einen wichtigen taktischen Anteil vom Gameplay dar. Häufig kommt es zu der Situation, dass der Spieler ein Geräusch hört, aber nicht die Quelle sehen kann. Der Spieler muss dann teilweise innerhalb von Sekundenbruchteilen die Richtung ausmachen können, aus der das Geräusch gekommen ist. Handelt es sich bei diesem beispielsweise um Fußstapfen, können durch eine gute Lokalisierbarkeit der Schallquelle Informationen, wie die Position des Gegners, erlangt werden ohne diesen sehen zu müssen. Diese Informationen bieten einen immensen taktischen Vorteil und können über Sieg oder Niederlage entscheiden. In Titeln wie „Counter-Strike:Global Offensive“ oder „Valorant“, welche eine große kompetitive E-Sport Szene haben, in der es um viel Geld geht, möchte man also eine so gute Lokalisierbarkeit von Schallquellen haben wie möglich. In der Praxis verwenden die beiden genannten Titel dennoch relativ simple Verfahren für das Audio-Rendering. Sie nutzen einfaches Panning, welches wahlweise durch HRTF-Filter erweitert werden kann, um einen räumlichen Schall zur 3-dimensionalen Lokalisierbarkeit von Schallquellen zu ermöglichen. Dabei wird kein Wert auf eine realistische Akustik gelegt. So können Schallquellen beispielsweise durch jegliche Wände hindurch gehört werden ohne dabei je nach dicke der Wände stärker gedämpft zu werden. Auch generieren Schallquellen keinen Hall, wenn sie beispielsweise in leeren Räumen sind.

Die technischen Möglichkeiten und Frameworks, um realistische Schallausbreitung zu simulieren, sind in der Industrie bereits erhältlich. Sie werden auch in Titeln wie *Sea of Thieves*, *Escape from Tarkov* oder *Borderlands 3* verwendet. Aus welchem Grund verwenden zwei der größten Shooter-E-Sport Titel „Counter-Strike:Global Offensive“ und „Valorant“ ein simples, statt realistisches Audio-Rendering? Ist dies für die Lokalisierbarkeit von Schallquellen vielleicht besser? Diese Fragen sollen mithilfe einer im Rahmen dieser Arbeit durchgeführten Evaluation beantwortet werden.

Struktur. In dieser Arbeit werden vier Audio-Frameworks im Bezug auf die Lokalisierbarkeit von Schallquellen verglichen. Die Frameworks *FMOD*, *Steam Audio* und *Project Acoustics* sind aktuelle, in der Industrie genutzte Frameworks. Das vierte Framework ist das im Rahmen dieser Arbeit mit Unity entwickelte

Graph Audio, welches auf dem *Graph-Based Real-Time Spatial Sound Framework* von Cowan et al. [1] basiert. Zunächst werden die benötigten Grundlagen sowie verwandte Arbeiten in diesem Themenbereich vorgestellt. Anschließend werden im Hauptteil die Funktionsweisen und verwendeten Methoden der drei kommerziellen Frameworks, sowie von Cowans Framework, welches Graph Audio zugrunde liegt, erläutert. Es folgen Details zur Konzeption und Implementierung von Graph Audio, welches an einigen Stellen Unterschiede zu Cowans Framework aufweist. Daraufhin wird die im Rahmen dieser Arbeit durchgeführte Evaluation von Konzept bis hin zum Ergebnis diskutiert. Die Leitfrage anhand derer die Evaluation der Frameworks konzipiert wurde ist die Frage, ob eine realistischere Schallausbreitung auch in einer erhöhten Lokalisierbarkeit der Schallquellen resultiert. Dazu wurde ein interaktiver Test von mehreren Probanden durchgeführt. Die Ergebnisse der Evaluation werden in dieser Arbeit analysiert und im Bezug auf obige Leitfrage kommentiert.

2 Grundlagen

In den nachfolgenden Abschnitten werden grundlegende Konzepte vorgestellt, die für das tiefere Verständnis dieser Arbeit notwendig sind.

2.1 Schall

Die Erläuterungen des folgenden Abschnitts basieren auf der Arbeit von Jacobsen et al. [2]. Die zugehörige Wissenschaft, welche sich mit Schall und dessen Ausbreitung beschäftigt, wird Akustik genannt. Beim Schall handelt es sich um mechanische Schwingungen, welche sich in Gasen, Flüssigkeiten oder Festkörpern ausbreiten. Dabei bewegen sich diese Schwingungen in Form von Schallwellen fort. Da für diese Ausarbeitung lediglich die Schallausbreitung in Fluiden, also Gasen und Flüssigkeiten, von Bedeutung ist, bezieht sich folgendes auf diesen Fall. In Fluiden handelt es sich bei Schallwellen um longitudinale Wellen, welche durch Vibrationen, genauer Druck- und Dichteschwankungen, hervorgerufen werden. Dabei gilt die *linearisierte Wellengleichung*, Gleichung 1, in einem kartesischen Koordinatensystem (x, y, z) [2] :

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} \quad (1)$$

Hierbei ist p der Schalldruck , angegeben in Pascal (kurz Pa) wobei $1 Pa = 1 \frac{N}{m^2}$. Es handelt sich dabei um „die Differenz zwischen dem Momentanwert des Gesamtdrucks und des statischen Drucks“ (vgl. [2]) und ist die Größe, die der Mensch hört. Des weiteren ist t die Zeit und c die Schallgeschwindigkeit.

Auftretende Phänomene. Da es sich beim Schall um Wellen handelt, kommt es bei dessen Ausbreitung zu verschiedenen Phänomenen, die den Charakteristiken von Wellen entsprechen. Das grundlegendste Phänomen ist zunächst der Verlust

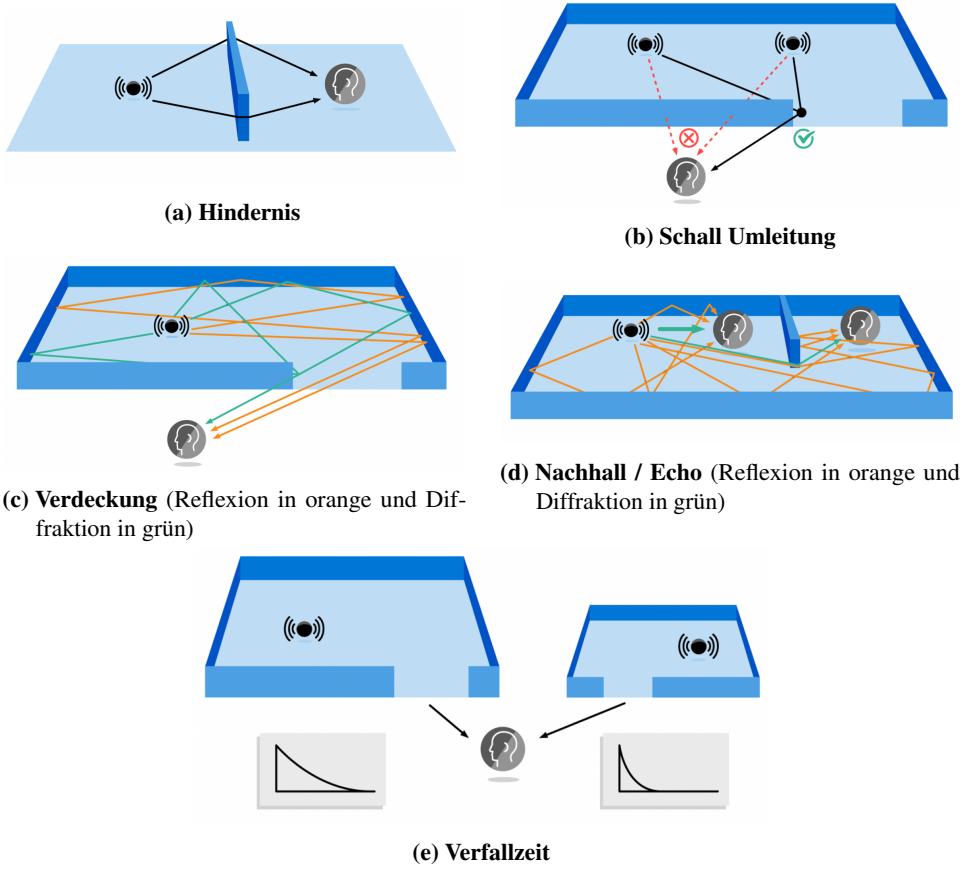


Abbildung 1: Effekte durch Schallausbreitung. Bilderquelle [3]

von Schalldruck bei zunehmender Distanz, auch Schalldämpfung genannt. Dieser tritt aufgrund der Ausbreitung der Wellenfronten, sowie der atmosphärischen Absorption auf und hat einen höheren Einfluss auf hohe Frequenzen [4]. Über lange Distanzen kommt so ein Effekt wie durch einen Lowpass-Filter zu Stande, bei dem die niedrigeren Frequenzen weiter zu hören sind, als die Hohen. Weitere Phänomene, die für diese Arbeit von Bedeutung sind, sind die folgenden:

- **Interferenz:** Schallwellen, die sich in verschiedene Richtungen ausbreiten, überlagern und stören sich gegenseitig.
- **Reflexion:** Schallwellen werden von festen Oberflächen reflektiert.
- **Refraktion:** Auch Schallbrechung genannt. Schallwellen ändern ihre Richtung, wenn das Medium in dem sie sich befinden inhomogen ist. Ein Beispiel dafür ist Luft, die starke Temperaturschwankungen aufweist.
- **Absorption:** Von weichen Oberflächen können Schallwellen teilweise absorbiert werden.
- **Streuung:** Wenn Schallwellen auf kleinere Hindernisse treffen, dann werden sie von diesen gestreut und bewegen sich weiter.
- **Diffraktion:** Auch Beugung genannt. Treffen Schallwellen auf die Ränder von Hindernissen, werden sie dort teilweise gebrochen. Dadurch gelangen Teile der Schallwellen am Hindernis vorbei.

Durch diese Phänomene ergeben sich verschiedene akustische Effekte die wir in unserem Alltag hören und wahrnehmen. Zunächst gibt es den Effekt, dass Schall abgeschwächt wird, wenn sich ein Hindernis zwischen Schallquelle und Hörer befindet. Der Weg des direkten Schalls ist also blockiert. Aufgrund der Diffraktion kann dennoch abgeschwächt etwas gehört werden, da sich die Schallwellen an den Kanten des Hindernisses beugen und so zum Hörer gelangen, wie in Abbildung 1a zu sehen ist. Der nächste Effekt ist die Umleitung des Schalles an Öffnungen, zu sehen in Abbildung 1b. Befinden sich Schallquellen in einem Zimmer und der Hörer steht im Flur, so kann er durch eine offene Tür den Schall hören. Allerdings kann die genaue Position der Schallquelle im Raum nicht lokalisiert werden, denn sie wird nicht direkt durch die Wand gehört (rot in Abb. 1b). Stattdessen kommt es einem so vor, als wäre die Tür die Schallquelle. Der Schall wird also umgeleitet. Ein weiterer Effekt, der anhand dieses Beispieles verdeutlicht werden kann, ist die Verdeckung. Denn als Hörer im Flur ist die Lautstärke, mit der der Schall gehört werden kann, deutlich geringer, als wenn man sich direkt in dem Raum befinden würde. Der Grund für die verringerte Lautstärke liegt in den vielen auftretenden Phänomenen der Schallwellen, wie der Reflexion (orange in Abb. 1) und Diffraktion (grün in Abb. 1) und den daraus resultierenden Interferenzen untereinander [5]. Verdeutlicht ist dies in Abbildung 1c. Im Zusammenhang mit der Verdeckung steht auch die Transmission. Erreicht ein verdeckter Schall den Zuhörer, ist es möglich,

dass sich ein Teil des Schalls durch verschiedene Materialien hindurch ausgebreitet hat. Je nach Eigenschaften der Materialien können dabei verschiedene Frequenzen des Schalls unterschiedlich stark beeinflusst werden. Meist werden hohe Frequenzen stärker absorbiert als Niedrige, wodurch sich der Schall dumpfer anhört [4]. Ein weiterer häufig auftretender Effekt ist der Nachhall, auch Echo genannt. Da der direkte Schall in Abb. 1d (dicker grüner Pfeil) lauter als die Reflexionen ist, wird die Quelle klar und mit wenig Nachhall gehört. Der gebeugte Schall hinter dem Hindernis hingegen ist abgeschwächt und übertönt nicht mehr die Reflexionen. Dadurch entsteht ein hoher Nachhall mit geringer Klarheit. Der letzte hier zu nennende Effekte, ist die in Abb. 1e gezeigte Verfallzeit. Große Räume haben einen länger anhaltendes Echo, das heißt die Verfallzeit ist größer. Dieser Effekt kann in einer leeren Kirche gut beobachtet werden. Wird dort laut gesprochen, hallt der Schall lange nach. Wenn hingegen in einem kleinen Zimmer laut gesprochen wird, gibt es sehr wenig hörbaren Nachhall.

2.2 Räumliche Schallsimulation

In der freien Natur kann der Mensch die Position von Schallquellen alleine mithilfe des Gehörs bestimmen. Für diese Lokalisierung kann er sowohl die Richtung als auch die Distanz zur Schallquelle feststellen, wenngleich die Distanz beim Fehlen von Schallreflexionen nur über die wahrgenommene Lautstärke bestimmt wird [6]. Die von Menschen empfundene Richtung aus der ein Geräusch kommt, hängt hauptsächlich von zwei Faktoren ab: den interauralen Pegelunterschieden (engl. *interaural level difference (ILD)*) und den interauralen Laufzeitunterschieden (engl. *interaural time delay (ITD)*) [7]. Im Falle von ILD hört ein Mensch eine Schallquelle, welche auf dem linken Ohr eine höhere Amplitude (Lautstärke) besitzt als auf dem rechten Ohr, aus der linken Richtung. Bei ITD wird bei gleicher Lautstärke ein Zeitunterschied beim Hören zur Lokalisierung genutzt. Erreicht der Schall das linke Ohr zuerst und das rechte Ohr ein wenig verzögert, lokalisiert der Mensch die Schallquelle links. In einer virtuellen 3-dimensionalen Umgebung nennt man Akustik, die es dem Nutzer ermöglicht Schallquellen zu lokalisieren, räumlichen Klang oder räumliches Hören. Die Art und Weise, wie eine solche räumliche Schallsimulation erreicht werden kann, hängt auch von den genutzten Ausgabegeräten ab. Für Stereo-Lautsprecher, Surround-Lautsprecher oder Kopfhörer gibt es unterschiedliche Verfahren [6]. In dieser Arbeit wird sich nur auf die Ausgabe über Zweikanal-Stereo-Kopfhörer bezogen, dass heißt es gibt einen Kanal für das rechte Ohr und einen für das linke Ohr.

Panning. Die grundlegendste Methode um aus einem einspurigen Mono-Audiosignal einen lokalisierbaren räumlichen Klang zu erzeugen ist das *Panning*. Dieses wird standardmäßig in Audio-Mixing-Software verwendet um ein Stereo-Klangbild zu erhalten. Hierbei wird sich die ILD zunutze gemacht, dass heißt ein Ton der links vom Zuhörer klingen soll, wird mit einem höheren Pegel bzw. höherer Amplitude auf dem linken Kanal als auf dem rechten Kanal wiedergegeben [8]. Die so

entstehende virtuelle Schallquelle ist dann eher auf der linken Seite des Zuhörers platziert. Eine Methode, wie dies umgesetzt werden kann, ist das von Pulkki in [7] vorgestellte *Vector Base Amplitude Panning (VBAP)*. Bei diesem können eine beliebige Anzahl an Lautsprechern zwei- oder dreidimensional um den Zuhörer platziert werden und für den räumlichen Klang genutzt werden. Im Fall von Kopfhörern haben wir also 2 Lautsprecher und somit 2 Kanäle. Die Amplitude der Signale auf den einzelnen Kanälen wird von den Verstärkungsfaktoren g_1 und g_2 kontrolliert. Bei diesen handelt es sich um nicht-negative Skalare. Damit die Lautstärke bei einer sich bewegenden Schallquelle kontrollierbar ist, müssen die Verstärkungsfaktoren normalisiert werden. Es wird der Parameter $C > 0$ als Lautstärkeregelung der virtuellen Schallquelle genutzt, welcher wie folgt approximiert werden kann:

$$g_1^2 + g_2^2 = C \quad (2)$$

Die Lautstärke der virtuellen Schallquelle wird auch dazu genutzt, um die empfundene Distanz zu dieser simulierbar zu machen. Ist die Distanz unverändert, kann die virtuelle Schallquelle nun auf dem „aktiven Bogen“, zu sehen in Abb. 2, platziert werden um die empfundene Richtung zu beeinflussen. Hierfür wird zunächst die Vektorbasis durch die Einheitsvektoren l_1 und l_2 definiert. Diese zeigen vom Ursprung des Koordinatensystems, welcher den Zuhörer simuliert, in Richtung der Lautsprecher. Dies ist ebenfalls in Abbildung 2 zu sehen. Aus der Linearkombination der Lautsprecher-Vektoren ergibt sich der Einheitsvektor p , welcher in Richtung der virtuellen Schallquelle zeigt:

$$p = g_1 l_1 + g_2 l_2 \quad (3)$$

In der Praxis ist die Position der virtuellen Schallquelle und somit p gegeben. Ebenfalls sind die Positionen der Lautsprecher gegeben, also l_1 und l_2 . Damit kann die obige Gleichung in eine Matrix-Form gebracht und nach $g = [g_1 g_2]$ aufgelöst werden. Wenn sich die virtuelle Schallquelle nicht komplett Links oder Rechts befindet, müssen die Verstärkungsfaktoren noch skaliert werden, damit sie Gleichung 2 erfüllen:

$$g^{skaliert} = \frac{\sqrt{C}g}{\sqrt{g_1^2 + g_2^2}} \quad (4)$$

2.3 Realistische Schallsimulation

Für eine immersive und realistische Simulation von Schall ist es wichtig, möglichst viele der in der Realität auftretenden Effekte aus Abschnitt 2.1 nachzubilden. Die Schallausbreitung wird dafür oft in die 3 Phasen *direkter Schall* (engl. *Direct Sound DS*), *frühe Reflexionen* (engl. *Early Reflections ER*) und *später Nachhall* (engl. *Late Reverberation LR*) aufgeteilt [5]. Hier macht DS den Schall aus, der direkt von der Schallquelle ins Ohr des Zuhörers gelangt. ER behandelt die Schallwellen, welche nur wenige Male reflektiert wurden und dann zum Ohr gelangen. Alle übrigen Effekte der Schallausbreitung, hervorgerufen durch Diffraction, Streuung, Refraktion und

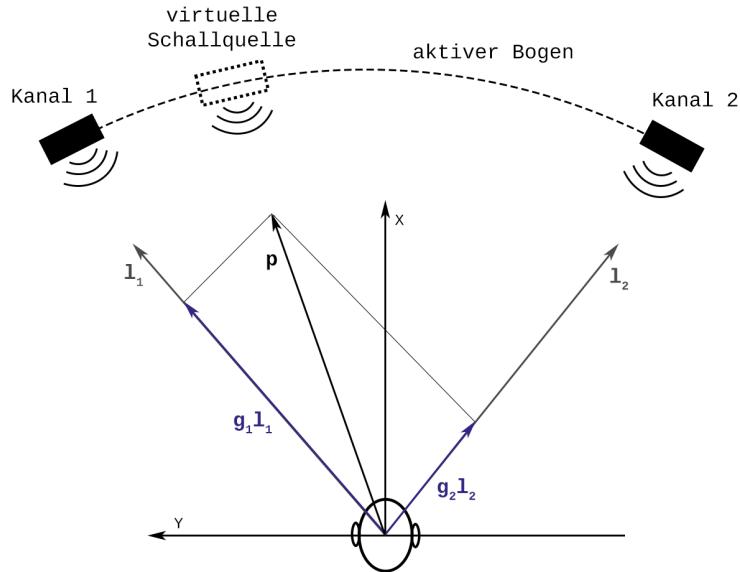


Abbildung 2: Stereo-Lautsprecherkonfiguration mit Vektoren formuliert . Grafik in Anlehnung an: [7]

häufiger Reflexion ergeben das Echo und werden unter LR zusammengefasst. Die Techniken, welche zur Simulation der Schallausbreitung genutzt werden, können klassifiziert werden in *geometrische Akustik* und *numerische (statistische) Akustik* [5].

Geometrische Akustik. Hierbei handelt es sich um Verfahren, welche Raytracing nutzen um die Schallausbreitung zu simulieren. Dies bedeutet der Schall wird mithilfe von Strahlen verfolgt. Diese werden dann für Effekte wie Nachhall rekursiv weiterbearbeitet, wenn sie auf Szenen-Geometrie treffen und von dort z.B. reflektiert werden. Da Strahlen weder Wellenlängen noch Phasen besitzen, werden diese beiden Aspekte der physikalischen Schallausbreitung vernachlässigt [2]. Bei den geometrischen Verfahren wird lediglich die Richtung, in die sich die Schallenergie ausbreitet, betrachtet und zur Simulation genutzt. Es können also nicht alle bei Wellen auftretenden Phänomene aus 2.1 nachgebildet werden, wodurch einige der zuvor erwähnten Schalleffekte nicht physikalisch korrekt simuliert werden können. Je nach Verfahren werden Effekte wie Diffraktion nicht abgebildet [5]. Dabei skalieren diese Verfahren mit der Komplexität und Anzahl an Polygonen der Szene, sowie der Rekursionstiefe der verfolgten Strahlen.

Numerische Akustik. Das Ziel dieser Verfahren ist es, die Wellengleichung 1 direkt zu lösen. Somit wird die Schallausbreitung physikalisch korrekt berechnet [5]. Daraus ergibt sich der Vorteil, dass alle natürlich auftretenden Phänomene aus Abschnitt 2.1 in der Theorie nachgebildet werden können. Außerdem skalieren

diese Verfahren nicht mit der Komplexität der Szenen-Geometrie, sondern mit den physikalischen Dimensionen der Szene [5]. Dadurch fällt für die Modellierer einer virtuellen Szene eine Restriktion weg, wodurch mehr Freiheiten in Szenen-Erstellungen möglich sind. Der Nachteil besteht darin, dass dies sehr rechenintensiv und somit schwieriger in Echtzeit umsetzbar ist, als die geometrischen Verfahren.

3 Verwandte Arbeiten

Da die Verfahren der geometrischen Akustik in der Regel deutlich weniger rechenintensiv sind, als die der numerischen Akustik, gibt es zu diesen Verfahren mehr Echtzeitfähige Arbeiten. Im folgenden werden zunächst Vorreiter in der geometrischen Akustik und anschließend ein weiteres Beispiel zur numerischen Akustik vorgestellt. Außerdem gibt es ein Beispiel zu einer zuvor getätigten Evaluation mit ähnlichen Leitfragen, wie die in dieser Arbeit verfolgten.

Der *Beam Tracing* Algorithmus von Laine et al. [9] aus dem Jahr 2009 ist eines der ersten Verfahren, welches sowohl dynamische Zuhörer-Positionen, als auch Schallquellen-Positionen unterstützt. Es wird dabei der aus der Computergrafik bekannte Algorithmus, welcher Strahlen eine Dicke hinzufügt um Wellenausbreitung zu simulieren, für die Schallausbreitung genutzt. Somit können die Pfade der frühen Reflexionen berechnet werden, um die aus diesen resultierenden Effekte zu erhalten. Ein anderes auf *Raytracing* basierendes Framework ist *GSound* von Schissler et al. aus dem Jahr 2011 [10]. Dieses wurde speziell für Computerspiele entwickelt und versucht somit möglichst wenig Ressourcen durch Berechnungen zu verbrauchen. Gleichzeitig werden durch die Nutzung von Raytracing mit der Szenen-Geometrie die Reflexions- und Diffektionspfade der Schallquellen berechnet, um realistische Akustikeffekte zu erhalten.

Eine Alternative der numerischen Akustik ist der Ansatz von Mehra et. al. [11] aus dem Jahr 2014. Dieses auf Wellenphysik basierende Verfahren zur Schallausbreitung verteilt in einem Vorverarbeitungsschritt *spherical harmonics* (SH) Schallquellen in der gesamten Szene. Für diese werden dann Schallfelder berechnet und codiert. Später zur Laufzeit können diese dann wieder dekodiert werden. Durch die genutzte Wellenphysik können Effekte, die durch Reflexion und Diffraction hervorgerufen werden simuliert werden.

In einer von Ibanez et al. getätigten Evaluation [12] aus dem Jahr 2017, wurde untersucht, inwiefern das Hinzufügen von Lowpass-Filters zur Simulation von Verdeckung einen Einfluss auf das Lokalisieren einer Schallquelle hat. Dazu wurden zwei Varianten des Audio-Rendering verglichen: Auf der einen Seite das standardmäßig in *Unity* und *Unreal Engine* genutzte Audio-System, welches simples Panning und eine einfache kurven-basierte Schalldämpfung liefert und auf der anderen Seite das durch Lowpass-Filter erweiterte System, welches auch Verdeckung simuliert. Für

den Vergleich wurden zwei Tests durchgeführt. Eine Online-Umfrage, in welcher die Probanden reine Audio-Clips der beiden Varianten anhören sollten und anhand dieser entscheiden sollten, aus welcher Richtung ein Alarmton-Geräusch gekommen ist. Für den zweiten Test ist ein minimalistisches 3D Videospiel erstellt worden. In diesem bewegten sich um die Testperson in einem leeren Raum acht verschiedene Sphären. Eine dieser Sphären hat in Dauerschleife ein Alarmton von sich gegeben. Die Testpersonen mussten nun raten welche dieser Sphären den Ton von sich gab. Auch hierfür gab es eine Variante mit der ersten Methode des Audio-Rendering und eine mit der zweiten Methode. Die Ergebnisse beider Tests sind identisch gewesen: Die durch Lowpass-Filter erweiterte Methode des Audio-Rendering hat in einer besseren Lokalisierbarkeit der Alarmtöne resultiert.

4 Audio-Frameworks für räumlichen Schall

In diesem Kapitel werden die vier verschiedene Frameworks und deren verwendete Verfahren zum Simulieren von räumlichem Schall vorgestellt. Begonnen wird mit den in der Industrie genutzten Frameworks *FMOD* [13] von Firelight Technologies, *Steam Audio* [14] von Valve Corporation und *Project Acoustics* [15] von Microsoft. Daraufhin wird das *Graph-Based Real-Time Spatial Sound Framework* [1] von Cowan et al. betrachtet, welches die Grundlage für das im Rahmen dieser Arbeit entwickelte Framework *Graph Audio* liefert.

4.1 FMOD

Es handelt sich bei FMOD um Audio Middleware, welche als adaptive Gesamtlösung für Videospiele von Firelight Technologies entwickelt wurde [13]. FMOD wird in vielen bekannten Titeln, wie *World of Warcraft*, *Dark Souls 3* oder der *Just Cause*-Serie [16], für das gesamte Audio Design eingesetzt. Die Gesamtlösung ist unterteilt in drei Teile: FMOD API, FMOD Studio und FMOD Integration. Die API ist das Grundstück des Frameworks und kann unabhängig von den anderen Teilen verwendet werden. Sie ist für die gesamte Audio-Funktionalität zuständig. Um es den Designern zu erleichtern den Sound für das Spiel zu erstellen und zu modifizieren, gibt es mit FMOD Studio eine eigene DAW¹-ähnliche Applikation. Diese bietet eine grafische Oberfläche, zu sehen in Abbildung 3, mit der das gesamte Sound Design umgesetzt werden kann. Für die Game-Engines *Unity* und *Unreal Engine 4* bietet FMOD außerdem jeweils eine komplette Integration der API und FMOD Studio in den bestehenden Workflow der Engines.

4.1.1 Funktionsweise

Es handelt sich bei FMOD nicht um ein Framework, welches speziell für die Simulation von räumlichem Schall konzipiert wurde. Es bietet aber verschiedene

¹Digital Audio Workstations (kurz DAW) sind Programme, die zur Musikproduktion und Abmischung verwendet werden.

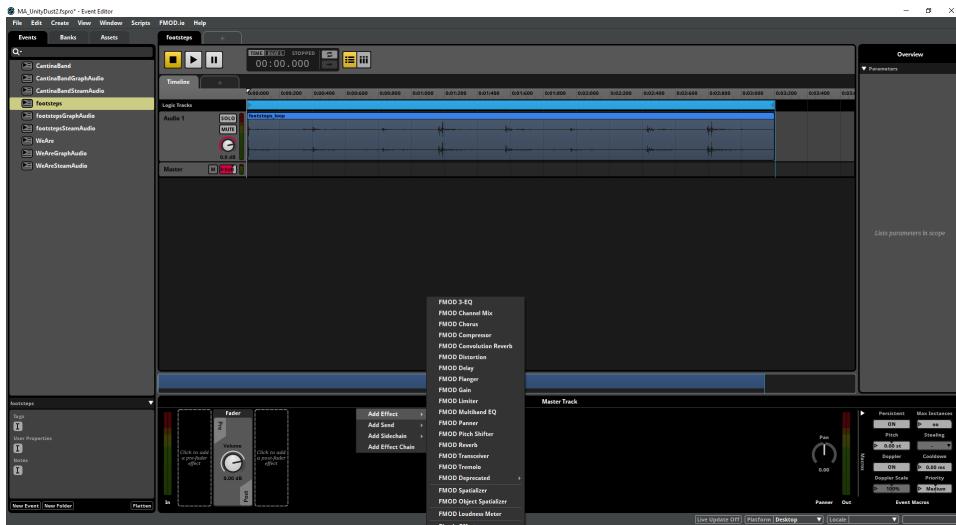


Abbildung 3: Grafische Oberfläche inklusive verfügbarer Audio-Effekte von FMOD Studio.

Effekte, welche traditionell in der Tontechnik genutzte Parameter für *Digital Signal Processors* (DSP) nutzen, mit denen räumlicher Schall simuliert werden kann. In Abbildung 3 sind die standardmäßig verfügbaren Effekte zu sehen, welche durch Plugins noch erweitert werden können. Einer dieser Effekte ist der *Spatializer*. Er bietet eine der grundlegendsten Methoden um räumlichen Schall zu erzeugen. Dazu wird das VBAP-Verfahren, siehe Kapitel 2.2, genutzt, um dem Audio-Signal eine Richtung und Distanz für die Lokalisierung in einer 3D-Umgebung hinzuzufügen [17]. Benötigt wird für das Panning die Positionen des Zuhörers und der virtuellen Schallquelle, welche die Sounds im Spiel abspielen soll. Die Position des Zuhörers ist im Falle eines First-/Third-Person Spieles die Kameraposition. Die Position der virtuellen Schallquelle ist das Objekt, welches die Sounds abspielen soll. Beide Informationen bezieht FMOD automatisch nachdem das System eingerichtet worden ist. In Abbildung 4 ist der Spatializer-Effekt in FMOD Studio abgebildet. Der Parameter *Distance Attenuation* approximiert die Schalldämpfung mit zunehmender Entfernung zwischen Schallquelle und Zuhörer. Dazu wird hier keine physikalische Basis, wie die atmosphärische Absorption (siehe Kapitel 2.1), verwendet, sondern eine in Spielen üblicherweise eingesetzte *Rolloff*-Kurve. Die Art der Kurve kann zwischen linear quadriert, linear, invers und invers spitz zulaufend gewählt werden oder die Schalldämpfung kann deaktiviert werden. Zusätzlich muss eine minimale und maximale Distanz angegeben werden. Während das Audio-Signal unterhalb der minimalen Distanz nicht gedämpft wird, erreicht die Lautstärke $-\infty$ dB sobald die maximal Distanz vorhanden ist [18]. Die Parameter *Sound Size* und *Min Extent* werden automatisch oder manuell gesetzt. Im automatischen Modus ist ersterer das zweifache der minimalen Distanz und gibt die Größe der Schallquelle an. Dies verhindert ein „flippen“ der Richtung von einer Seite zur Anderen, wenn sich der



Abbildung 4: *Spatializer* Audio-Effekt in FMOD Studio.

Zuhörer sehr nahe an der Schallquelle aufhält. *Min Extent* gibt eine minimal Ausdehnung zwischen 0 und 360 Grad für das Panning an und besitzt den Standardwert 0 Grad.

Neben der Schalldämpfung werden somit keine der in Kapitel 2.1 vorgestellten Schallphänomene vom FMOD Spatializer automatisch simuliert. So gibt es beispielsweise keine Verdeckungsberechnung durch Raytracing weshalb Sounds durch jegliche Szenen-Geometrie hindurch gehört werden können. Daraus folgt, dass der Spatializer weder zur geometrischen noch zur numerischen Akustik gezählt werden kann. Dennoch ist es möglich die Schallphänomene manuell über die von FMOD zur Verfügung gestellten Effekte zu simulieren. Ein Raumhall kann durch den *FMOD Reverb* erzeugt werden und Verdeckungen können mithilfe des *FMOD Lowpass*-Effekt, welcher hohe Frequenzen filtert, simuliert werden. Diese Effekte sind dann allerdings nicht physik-basiert, denn ein Sound Designer muss diese für jede Schallquelle der Szene manuell einstellen. Eine HRTF, um die 3D Lokalisation mit Stereo-Kopfhörern zu verbessern, wird standardmäßig nicht in FMOD angeboten.

4.2 Steam Audio

Steam Audio ist ein Audio Framework, welches speziell für die Erzeugung von räumlichem Schall von Softwarehersteller Valve erschaffen wurde [19]. Vorgestellt worden ist es im Jahr 2017 und wird seitdem konstant weiter entwickelt. Es bietet physik-basierte Akustik für eine immersive und realistische Sound Kulisse sowohl in 3D- als auch in VR-Videospielen. Steam Audio kann zum einen mithilfe der C-API eigenständig genutzt werden und zum anderen integriert werden in die Game-Engines *Unity* und *Unreal Engine 4*. Des Weiteren gibt es eine Integration in FMOD Studio in Form eines Plugins, zu sehen in Abbildung 5, welches ähnlich zum FMOD Spatializer genutzt werden kann. Steam Audio wird im Valve eigenen VR-Titel *Half-Life Alyx* oder auch im FPS Battle-Royal *Escape from Tarkov* verwendet [20]. Neben einem durch HRTF-Filtern verbessertem räumlichem Audio-Rendering, bietet Steam Audio die Möglichkeit automatisch die Akustik-Effekte Schalldämpfung, inklusive frequenzabhängiger Luftabsorption, Verdeckung und



Abbildung 5: Steam Audio Spatializer Plugin in FMOD Studio.

Schall-Umleitung zu berechnen. Außerdem können durch Hindernisse auftretende indirekte Schallwellen-Pfade berechnet werden. Die dafür nötigen Berechnungen mit der Szenen-Geometrie werden mithilfe von Raytracing durchgeführt [14]. Aus diesem Grund zählen die in Steam Audio genutzten Verfahren zu denen der geometrischen Akustik. Es werden außerdem sowohl dynamische Schallquellen- und Zuhörerpositionen, als auch dynamische Szenen-Geometrie, wie sich öffnende Türen oder Fenster, unterstützt.

4.2.1 Funktionsweise

Für den grundlegenden 3-dimensionalen Sound verwendet Steam Audio ein auf HRTFs basierendes binaurales Rendering [21]. Binaural bedeutet, dass es für das linke und rechte Ohr verschiedene Filter gibt. Die genutzten HRTF-Filter hängen von der Richtung ab, in der sich die Schallquelle im Vergleich zum Zuhörer befindet. Dabei sind in Steam Audio für eine feste Anzahl an Richtungen verschiedene HRTFs hinterlegt. Je nach Einstellungen wird entweder zwischen den vier der Richtung am nächsten liegenden HRTFs interpoliert oder es wird lediglich die einzelne nächste gewählt. Die Interpolation resultiert in saubererem Audio-Rendering mit weniger Artefakten, kostet aber auch bis zu zwei mal so viel Prozessorleistung [22]. In Abbildung 5 sind alle einstellbaren Effekte zu sehen. Die Werte dieser können entweder automatisch physik-basiert simuliert werden oder manuell vom Sound Designer eingestellt werden. Zunächst gibt es eine Schalldämpfung (engl. *Distance Attenuation*), die in Form einer Kurve eingestellt werden kann, identisch zu denen des FMOD Spatializer. Zusätzlich wird hier die Möglichkeit der physik-basierten Schalldämpfung geboten. Diese nutzt einen inversen Distanzabfall. Die Erweiterung dazu ist die frequenzabhängige Luftabsorption (engl. *Air Absorption*), welche einen exponentiellen physik-basierten Abfall simuliert. Bei diesem fallen über die Distanz die hohen Frequenzen stärker ab als die niedrigen. Im manuellen Modus können

die niedrigen, mittleren und hohen Frequenzen separat eingestellt werden. Über die Richtcharakteristik (engl. *Directivity*) kann eine Schallquelle so modifiziert werden, dass sie in bestimmte Richtungen mehr Schall abgibt, als in andere. Der nächste Effekt ist die Verdeckung (engl. *Occlusion*), welche wahlweise einen einzelnen (*single*) oder mehrere (*volumetric*) Strahlen vom Zuhörer zur Schallquelle verschießt. Befindet sich Szenen-Geometrie zwischen Beiden, dann erhöht sich der Verdeckungswert, je nach Anzahl an Strahlen, welche die Schallquelle nicht erreichen. Damit dies funktioniert muss zunächst die Szenen-Geometrie, welche Einfluss auf die Schallausbreitung haben soll, markiert und exportiert werden. Dies kann einmalig zur Designzeit getan werden. Wenn die Verdeckungsberechnung aktiviert ist, kann optional die Transmission genutzt werden. Diese veranlasst die Verdeckungsberechnung dazu, ein Akustik-Material auf der Szenen-Geometrie einzukalkulieren. Die Transmission kann wahlweise frequenzunabhängig, als ein einzelner Faktor, oder -abhängig in niedrigen, mittleren und hohen Frequenzen, modelliert werden. Schall, welcher den Zuhörer indirekt über Reflexionen (engl. *Reflections*) an der Szenen-Geometrie erreicht, kann ebenfalls simuliert werden. Dies führt besonders zu einem wahrnehmbaren Hall-Effekt. Der letzte Effekt von Steam Audio ist die Pfadfindung (engl. *Pathing*). Diese berechnet, mit Einbeziehung der Szenen-Geometrie, den kürzesten Pfad zwischen Zuhörer und Schallquelle. Diese indirekten Schallpfade werden dann für den räumlichen Schall mit einbezogen.

4.3 Project Acoustics

Bei *Project Acoustics*² [15] handelt es sich um eine Akustik-Engine von Microsoft für interaktive 3D-Software wie Computerspiele. Vorgestellt wurde das Framework 2019 und wird in AAA-Titeln wie *Gears of War 5*, *Sea of Thieves* oder *Borderlands 3* verwendet [3]. Es nutzt als Grundlage die „parametrische Wellenfeld-Kodierung für vorberechnete Schallausbreitung“ [23] und erweitert dies konstant, wie zum Beispiel mit einer richtungs-basierten Kodierung [24]. Da physikalische Wellen zur Berechnung genutzt werden, handelt es sich um ein Verfahren der numerischen Akustik 2.3. Dadurch können, im Gegensatz zu raytracing-basierten Verfahren, Effekte wie Schall-Umleitung, Verdeckung, Nachhall und durch Hindernisse oder Verfallszeit auftretende Effekte (zu sehen in Abb. 1) physikalisch korrekt modelliert werden. Somit hört sich die gesamte Akustik innerhalb einer virtuellen Szene realistischer an, was zu einer höheren Immersion für Spieler führt. Ein weiterer Vorteil von *Project Acoustics* ist die Integration sowohl in bestehende Game-Engines, wie *Unity* oder *Unreal-Engine*, als auch in Audio-Middleware. Das System wandelt die erstellten Parameter für Verdeckung, Hall und Schall-Umleitung in traditionell genutzte DSP-Parameter um. Dadurch können Toningenieure direkt mit diesen arbeiten und müssen sich nicht erst auf das neue System umstellen. Das Framework unterstützt dynamische Schallquellen- und Zuhörerpositionen, allerdings muss die Szenen-Geometrie statisch sein. Die Pipeline des gesamten Systems ist in Abbildung

²Auch *Project Triton* auf der offiziellen Forschungswebsite [3] genannt.

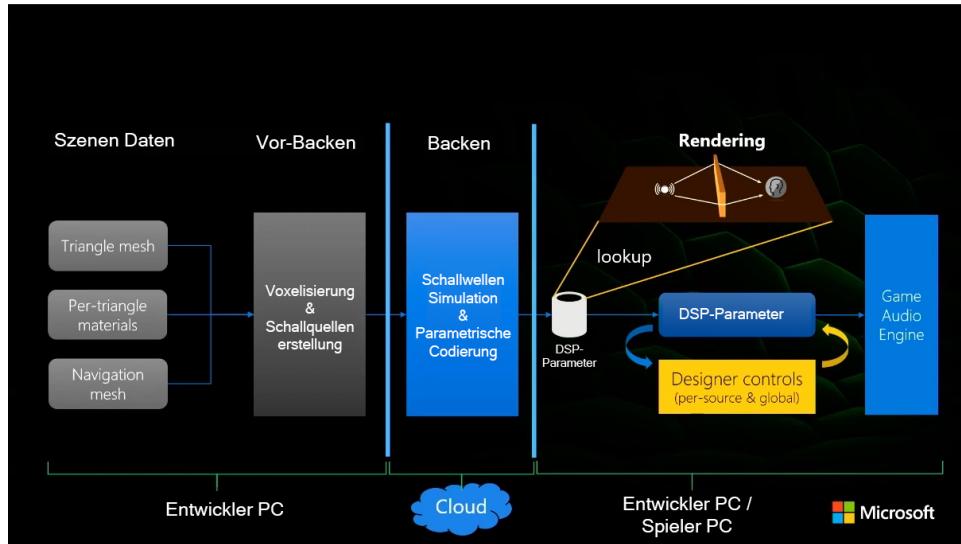


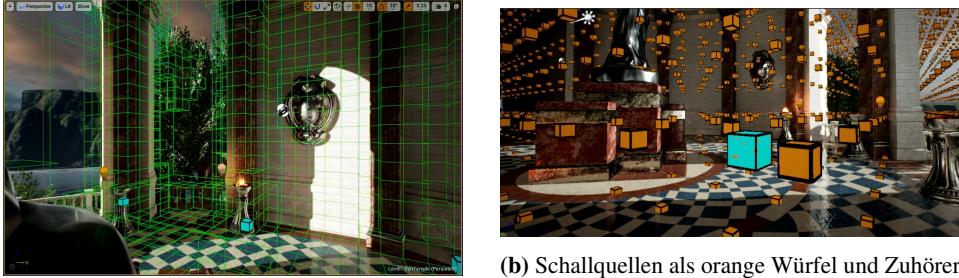
Abbildung 6: Gesamte Pipeline des *Project Acoustics* System. Bildquelle: [25]

6 zu sehen und wird im folgenden Abschnitt erläutert.

4.3.1 Funktionsweise

Wie bereits im vorherigen Abschnitt erwähnt, basiert *Project Acoustics* auf dem Verfahren der „parametrischen Wellenfeld-Kodierung für vorberechnete Schallausbreitung“ [23]. In Kapitel 2.3 wurde gesagt, dass die Berechnung der Wellengleichung rechenintensiv ist. Dies wird in diesem Verfahren durch einen Vorverarbeitungsschritt, dass so genannte *Baking* gelöst.

Baking. Das *Baking* muss zunächst lokal vorbereitet werden, was auch *Pre-Baking* genannt wird. Als erstes wird im Editor der 3D-Szene die statische Geometrie ausgewählt, welche auf die Schallausbreitung Einfluss nehmen soll. Dadurch ist es möglich kleinere Objekte die wenig bis keinen Einfluss auf die Schallausbreitung haben, wie Messer oder Gabeln, zu ignorieren. Auf der ausgewählten Geometrie werden lokal auf dem Rechner geometrische Analysen, wie eine Voxelisierung, durchgeführt. Die Unterteilung der Szene in Voxel ermöglicht es unter anderem Geometrie, die auf die Schallausbreitung keinen Einfluss hat, für die Berechnungen zu ignorieren und so Ressourcen zu sparen. In Abbildung 7 ist das Ergebnis dieser lokalen Vorverarbeitung sichtbar, wobei die Voxel in 7a als grüne Gitterboxen sichtbar sind. Außerdem sind in 7b orangene und blaue Würfel zu erkennen. Bei den orangefarbenen handelt es sich um Schallquellen, die automatisch von *Project Acoustics* in der Szene verteilt worden sind. Die blauen Würfel sind mögliche Zuhörerpositionen. Gemacht wird dies, damit beim *Baking* für jede mögliche Positions-Kombination von Schallquellen und Zuhörern Berechnungen getätigten werden können. Hierdurch entsteht der Vorteil, dass später zur Laufzeit auch dynamische Schallquellen ge-



(a) Voxel als grüne Gitterboxen.

(b) Schallquellen als orange Würfel und Zuhörer als blaue Würfel.

Abbildung 7: Ergebnis der lokalen Vorberechnung (*Pre-Baking*). Bilderquelle: [25]

nutzt werden können. Die dafür benötigte Schallausbreitung muss dann nicht mehr neu berechnet werden, sondern kann aus den Ergebnissen des *Bakings* abgerufen werden. Die Ergebnisse des *Pre-Baking* werden daraufhin zu Microsofts Cloud-Computing-Dienst *Azure* geschickt, wo auf leistungsstarken Clustern das *Baking* durchgeführt wird. Die tatsächliche Position der Schallquelle und des Zuhörers, jeweils 3 Dimensionen, und die Zeit spannen ein 7-dimensionales Druckfeld auf. Im *Baking* wird dieses Feld diskretisiert, indem für eine feste Schallquelle verschiedene 4-dimensionale Schnitte mit Zuhörerposition und Zeit erstellt werden. Diese nutzen die Testpositionen aus dem *Pre-Baking* und führen für jeden Schnitt eine Wellensimulation durch. Es entstehen in diesem Schritt, je nach Szenengröße, eine große Menge an Daten. Zu groß um sie direkt von Azure zurück zu schicken. Deshalb werden die Daten zunächst durch einen proprietären parametrischen Kompressor geschickt, welcher die zu Beginn des Kapitels genannten DSP-Parameter erstellt. Dieser Schritt nennt sich *parametrische Codierung*. Zurück erhält man ein „Akustik-Game-Asset“, welches in die Szene geladen werden kann und zur Laufzeit vom System genutzt wird.

Laufzeit und Rendering. Beim laden des Akustik-Game-Asset werden die Test-Schallquellen mit ihren codierten Parametern in ein Uniform-Grid geladen [23]. Für eine tatsächliche Schallquelle aus der Szene können nun diejenigen 8 Test-Schallquellen aus dem Grid geladen werden, die eine Box um diese bilden. Von diesen 8 Test-Schallquellen werden nun alle entfernt, die in Wänden und außerhalb der aktuellen Region-Of-Interest liegen oder nicht im Sichtfeld der tatsächlichen Schallquelle sind. Aus dem resultierenden Set von Test-Schallquellen können nun für die aktuelle Zuhörerposition die tatsächlich genutzten DSP-Parameter mithilfe von tri-linearer Interpolation berechnet werden.

Für den finalen Stereo Soundoutput o eines jeden Schallquelle-Zuhörer-Paares wird parametrisches Rendering mit einer festen Anzahl globaler kanonischer Filter verwendet. Betrachte man die i -te tatsächliche Schallquelle mit Mono-Sound-Signal $s_i(t)$ zum Zeitpunkt t . Auf dieses Signal wird mittels Faltung ein Stereo-Filter (Binauraler-Filter) $h_i(t)$ angewandt, welcher die zuvor berechneten DSP-Parameter

nutzt. Daraus resultiert der Stereo Output:

$$o_i(t) = s_i * h_i \quad (5)$$

Wie bereits in Kapitel 2.3 erläutert kann Schallausbreitung in die 3 Phasen *direkter Schall* (DS), *frühe Reflexionen* (ER) und *später Nachhall* (LR) unterteilt werden. Wendet man dies auf h_i an erhält man:

$$h_i = h_i^{DS} + h_i^{ER} + h_i^{LR} \quad (6)$$

Insgesamt kann das Rendering damit als Summe von 3 Faltungen ausgedrückt werden:

$$o_i(t) = o_i^{DS} + o_i^{ER} + o_i^{LR} = h_i^{DS} * s_i + h_i^{ER} * s_i + h_i^{LR} * s_i \quad (7)$$

4.3.2 Vor- und Nachteile

Bei *Project Acoustics* handelt es sich um ein innovatives Framework für Echtzeit Schallsimulation, welches dynamische Schallquellen und Zuhörerpositionen unterstützt. Der größte Vorteil liegt in der akkuraten und realistischen Simulation der Schallausbreitung durch die Nutzung echter Wellenphysik. Zuvor ist dies in diesem Ausmaße nicht möglich gewesen für Echtzeit-Systeme, wie in Kapitel 3 zu sehen. Dadurch werden die für den realistischen Sound verantwortlichen Effekte aus Abbildung 1 realitätsnah abgebildet. Wenn zur Entwicklung *Unity* oder *Unreal Engine* genutzt wird, stehen außerdem Plugins zur Integration in beide Spiel-Engines zur Verfügung. Durch die im System vorhandene Umwandlung in übliche Audio-DSP-Parameter ergibt sich der weitere Vorteil, dass Toningenieure und Entwickler die Akustik auch nach ihren Wünschen anpassen können. Dies ist interessant, wenn ein absichtlich unrealistischer Sound gewünscht ist, um verschiedene Effekte oder Schallquellen hervorzuheben. So können die Entwickler selber entscheiden, wie das Sound-Design in ihrer Anwendung sein soll.

Nachteile des Frameworks sind zum einen, dass während der Entwicklung der Schritt des *Baking* extern über Microsoft *Azure* durchgeführt werden muss. Somit ist der Entwickler darauf angewiesen, dass dieser Dienst auch zur Verfügung steht. Außerdem erhält Microsoft so Zugriff auf die Szenen-Daten. Zum anderen ist die Entwicklung beschränkt auf die Spiel-Engines *Unity* und *Unreal Engine*. Eine Unterstützung für dynamische Geometrie ist in der aktuellen Version nicht vorhanden. Es gibt allerdings eine Beta-Version für die *Unreal Engine*-Variante von *Project Acoustics*, in der dynamische Geometrie auch genutzt werden kann [26].

4.4 Graph-Based Real-Time Spatial Sound Framework

Cowan stellt in [1] aus dem Jahr 2020 ein Framework für die Simulation von räumlichem Schall vor. Entwickelt wurde dieses für die Nutzung in Computerspielen oder virtuellen Umgebungen, also für Echtzeitanwendungen. Das Framework nutzt einen Graphen, um die Ausbreitung von Schallenergie zu modellieren. Dies vermeidet

zum einen die Berechnungen der Schallausbreitung mit der Szenen-Geometrie zur Laufzeit und zum anderen wird eine parallele Berechnung der einzelnen Knoten des Graphen durch eine GPU ermöglicht. Das Framework baut dabei auf *FMOD* auf, welches für das finale Audio-Rendering genutzt wird. Das Framework fokussiert sich somit darauf, räumlichen Schall zu simulieren. Hierfür berechnet das Framework für verschiedene Effekte DSP-Parameter, welche dann von FMOD genutzt werden, um durch Faltung mit Filtern die Effekte zu rendern. Dynamische Schallquellen und Zuhörerpositionen werden ebenfalls unterstützt, allerdings gibt es keine Möglichkeit dynamische Geometrie in die Berechnungen mit einzubeziehen.

Die in Kapitel 2.1 beschriebene Schalldämpfung wird durch FMOD umgesetzt, allerdings wird nicht die direkte Distanz zwischen Schallquelle und Zuhörer genutzt, sondern die mit dem Graphen berechnete Pfadlänge. Zusätzlich umgesetzte Effekte sind die *Verdeckung* und der *Nachhall*. Diese werden mithilfe der Szenen-Geometrie approximiert, weshalb die dafür genutzten Verfahren zu denen der *geometrischen Akustik* gehören. Weitere Effekte aus Abbildung 1, welche größtenteils durch Diffraktion hervorgerufen werden, sind nicht speziell implementiert. Die Diffraktion wird nicht einzeln berechnet, da laut Cowan für die Verdeckung und Diffraktion dieselben Filter zur Simulation von FMOD verwendet werden. Dadurch würden separat kalkulierte Werte die Simulation vermutlich nicht verbessern [1]. Ein weiteres Merkmal des Frameworks liegt in der „Mehrdeutigkeit“. Um eine räumliche Lokalisierung der Schallquelle durch direkten Schall zu ermöglichen, werden Panning und HRTF-Filter genutzt. Wenn der Weg des direkten Schalles aber verdeckt ist, müssen dessen gerenderte Hinweise zur Lokalisierung abgeschwächt werden, um eine realistische Simulation zu erreichen. Die „Mehrdeutigkeit“ modelliert diese Beeinflussung durch Verdeckung.

4.4.1 Funktionsweise

Die Grundlage des Frameworks liegt in dem Aufbau eines Graphen [1]. Dieser beschreibt eine Umgebung, indem er angibt ob sich Schall zu Nachbarknoten ausbreiten kann oder nicht. Bei der Umgebung kann es sich um eine 2D- oder 3D-Umgebung handeln. Für Umgebungen mit geometrischen Daten kann das Framework diesen Graphen automatisch aus der Szene erstellen lassen. Dies geschieht indem die Szene in ein Gitter mit einer diskreten Anzahl an Blöcken unterteilt wird. Jeder dieser Gitter-Punkte wird ein Knoten im Graph. Ist die Umgebung allerdings nicht durch geometrische Daten gegeben, ist es auch möglich den Graphen manuell zu erstellen. Nachdem der Graph einmal aufgebaut ist läuft das gesamte Audio-Rendering über diesen Graphen und die Interaktion mit Geometrie der Szene wird obsolet. Das gesamte Verfahren kann unterteilt werden in einen *Baking*-Schritt und einen *Rendering*-Schritt. Beim Baking wird der Graph erstellt und mit statischen zur Szene gehörenden Werten, welche im Verlauf noch genannt werden, gefüllt. Zur Laufzeit beim *Rendering*-Schritt werden dann die statischen Werte genutzt, um zu einem konkreten Zuhörer-Schallquellen-Paar die Parameter-Werte für die FMOD Effekte zu konstruieren. In Abbildung 8 ist ein manuell erstellter Graph abgebildet.

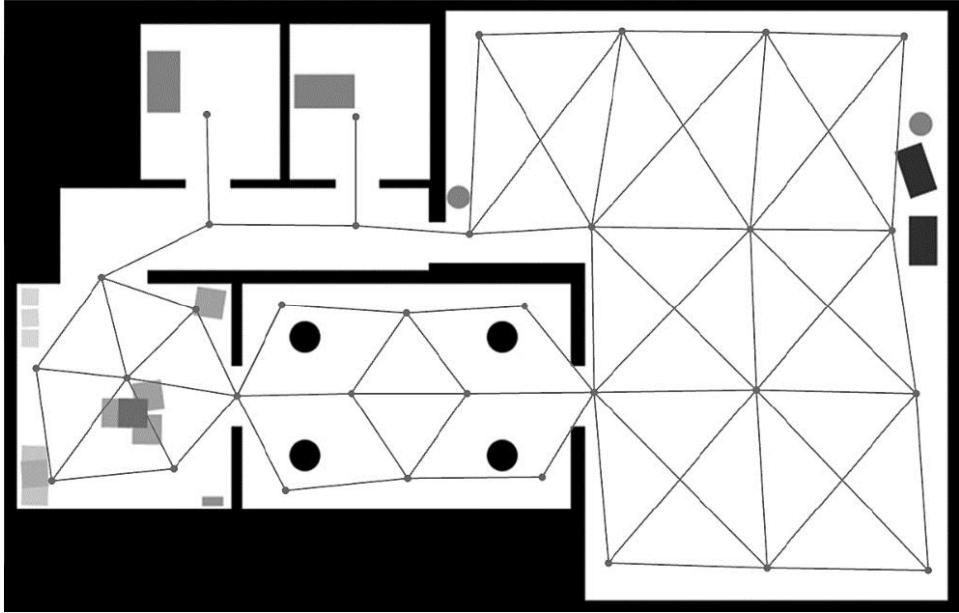


Abbildung 8: Graph in einer 2D-Umgebung, welcher angibt wohin Schall sich ausbreiten kann und wohin nicht. Graustufen der Objekte geben die Verdeckungs-Werte an, wobei Schwarz einer 100%igen Verdeckung entspricht. Bildquelle: [1]

Jeder Knoten hat dabei eine 3D-Position mit X-, Y- und Z-Koordinaten. Zu sehen ist außerdem, dass Knoten immer dann mit ihren Nachbarn verbunden sind, wenn es dem Schall möglich ist sich zu diesem Auszubreiten. Einige Objekte sind in Graustufen gehalten, durch welche auch Kanten durchlaufen. Dies hängt mit den Verdeckungs-Werten zusammen, denn nicht jedes Objekt verhindert die Schallausbreitung zu 100 Prozent. Ist der Verdeckungs-Wert geringer kann sich der Schall immer noch zwischen den beiden Knoten ausbreiten, weshalb eine Kante zwischen diesen gezeichnet wird. In dieser Kante wird der Verdeckungs-Wert gespeichert. Bei Knoten, die keine Hindernisse zwischen sich haben, ist der Verdeckungs-Wert der Kante 0. Berechnet werden kann der Wert mithilfe eines beliebigen geeigneten Verfahrens, wie *Raymarching*- oder *Raytracing*-Verfahren. Cowan schlägt in [27] ein GPU basiertes Verfahren zur akustischen Verdeckungs-Modellierung vor, welches *Raycasting* nutzt. Zusätzlich werden in den Knoten Parameter-Werte für den Nachhall gespeichert. Auch für deren Berechnung schlägt Cowan in [28] eine Methode vor. Auf beide von Cowan vorgeschlagenen Methoden wird nicht näher eingegangen, da diese in der Implementation dieser Arbeit nicht verwendet werden.

Baking. Beim ersten Schritt des Verfahrens handelt es sich um einen *Baking*-Schritt. Das heißt hier wird zur Entwicklungszeit der Graph erstellt und alle statischen Werte, die im Zusammenhang mit der Szenen-Geometrie, aber nicht eines konkreten Schallquellen-Zuhörer-Paars stehen, berechnet. Für Knoten ist dies die Position sowie der Wert für den Nachhall. Für Kanten ist dies die Kantenlänge sowie

der Verdeckungs-Wert. Um Speicherplatz zu sparen und nicht zwei separate Werte in Kanten speichern zu müssen, wird stattdessen ein einzelner Schalldämpfungs-Wert gespeichert. Dazu wird die Verdeckung der Kante in die Kantenlänge mit einkalkuliert, indem ein hoher Verdeckungs-Wert zu einer künstlichen Erhöhung des Schalldämpfungs-Wertes führt. Der Schalldämpfungs-Wert D_a zwischen zwei Knoten ist also deren durch Verdeckung erhöhte Distanz zueinander. Er wird wie folgt berechnet [1]:

$$D_a = D \times \left(1 + \frac{O_c^{1.5}}{4}\right) \quad (8)$$

Dabei ist D die physikalische Distanz und O_c der Verdeckungs-Wert dieser Kante. Der Verdeckungs-Wert nimmt Werte zwischen 0 und 255 an. Es folgt aus Formel 8, dass bei einem Verdeckungs-Wert von 0 D_a die tatsächliche physikalische Distanz annimmt. Ist die Verdeckung maximal, dann wächst D_a um das 1018-fache der physikalischen Distanz an. In diesem Fall wird also kein Schall zwischen beiden Knoten zu hören sein.

Laufzeit und Rendering. Um die Schallausbreitung mithilfe der Schallenergie zur Laufzeit zwischen einer Schallquelle und einem Zuhörer zu kalkulieren, muss auf dem Graphen der effizienteste Pfad gefunden werden, den der Schall nehmen kann. Da in einem Computerspiel aber oft mehrere Schallquellen gleichzeitig aktiv sind, wird die Pfadberechnung nicht für ein konkretes Schallquelle-Zuhörer-Paar durchgeführt. Stattdessen wird beim Zuhörer begonnen und von dort ein modifizierter Dijkstra-Algorithmus angewandt, welcher keinen Endknoten besitzt. Dieser wird jedes Frame durchgeführt und berechnet zu jedem anderen Knoten die Menge an Schallenergie, die diese jeweils erreicht. Jeder Knoten speichert dann seine geschätzte Gesamtmenge an Schallenergie vom Zuhörer ausgehend zu diesem Knoten und die Richtung zum Zuhörer. Da der Zuhörer selbst kein Knoten im Graphen besitzt, muss zunächst der ihm am nächsten liegende Knoten gefunden werden. Dieser Knoten ist der Startknoten für den Dijkstra-Algorithmus. Er und alle seine Nachbarknoten werden behandelt, als hätten sie eine direkte Verbindung zum Zuhörer. In Abbildung 9 ist ein Graph zu sehen, dessen Knoten direkt mit Zuhörer und Schallquelle verbunden sind. Für diese Knoten wird die Schallenergie anders berechnet, als für alle restlichen Knoten. Dazu wird folgende Formel 9 genutzt:

$$N_d = \| N_p - L_p \| \quad (9)$$

Hier ist N_d die Distanz zwischen dem Knoten und Zuhörer. N_p ist die Position des Knoten und L_p die Position des Zuhörers. Neben der Schallenergie wird auch eine Richtung in den direkt verbundenen Knoten gespeichert. Diese wird mit Formel 10 kalkuliert:

$$\widehat{V_n} = \frac{N_p - L_p}{N_d} \quad (10)$$

V_n ist ein Einheitsvektor, welcher vom Zuhörer in Richtung des Knoten zeigt. Für alle restlichen Knoten wird deren Schallenergie nach Dijkstras-Algorithmus berech-

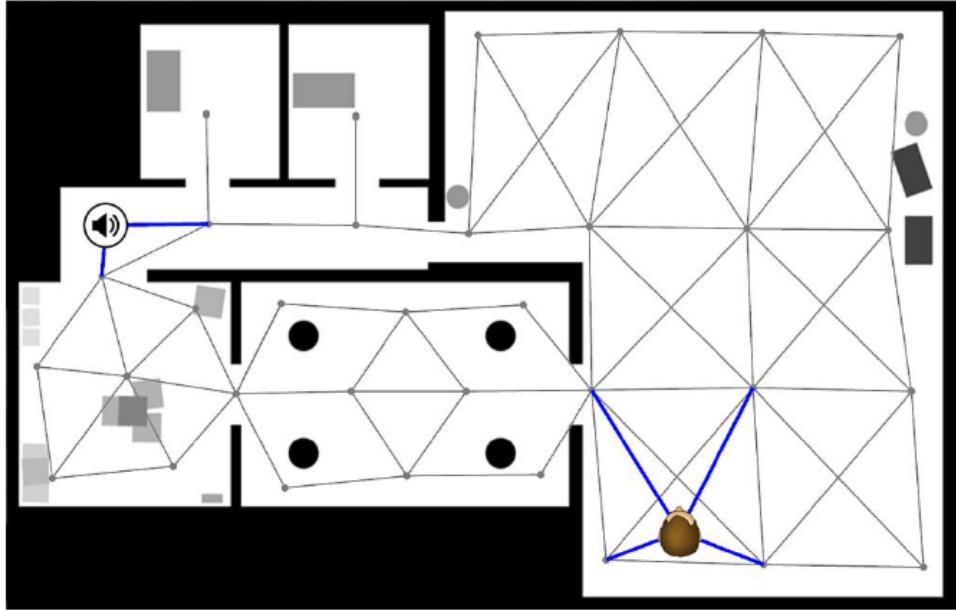


Abbildung 9: Direkte Verbindungen von Knoten und Zuhörer/Schallquelle. Bildquelle: [1]

net, indem die totale Schallenergie des Vorgängerknoten zum Schalldämpfungswert der Kante addiert wird.

Zu jeder vorhandenen Schallquelle kann nun der Knoten gefunden werden, welcher ihr am nächsten liegt. Der Knoten hat den kürzesten Pfad gespeichert, den der Schall zum Zuhörer nehmen kann. Diese Pfadlänge sowie die gespeicherte Richtung werden in eine virtuelle Schallquellen-Position umgewandelt. In Abbildung 10 ist diese in gelb eingezeichnet. Außerdem sind die durch Dijkstra berechneten Pfadlängen als Farben codiert sichtbar. Da die Kalkulationen aus Sicht des Zuhörers getätigten wurden, sind die Pfadlängen um den Zuhörer herum am geringsten. Die virtuelle Schallquelle wird nun in FMOD als Schallquelle angegeben. Das weitere Audio-Rendering inklusive Panning wird von FMOD übernommen.

Der Wert für den Verdeckungs-Effekt des Pfades kann nach Cowan mit folgender Formel 11 approximiert werden:

$$O = 1 - \left(\frac{D}{P} \right)^2 \quad (11)$$

Die Verdeckung O zwischen Schallquelle und Zuhörer nimmt Werte zwischen 0 und 1 an. D ist die direkte Distanz zwischen Schallquelle und Zuhörer, welche jegliche Szenen-Geometrie ignoriert. Die durch den Graph berechnete Pfadlänge ist P . Der Verdeckungs-Wert wird dann mithilfe der FMOD Effekte angewandt.

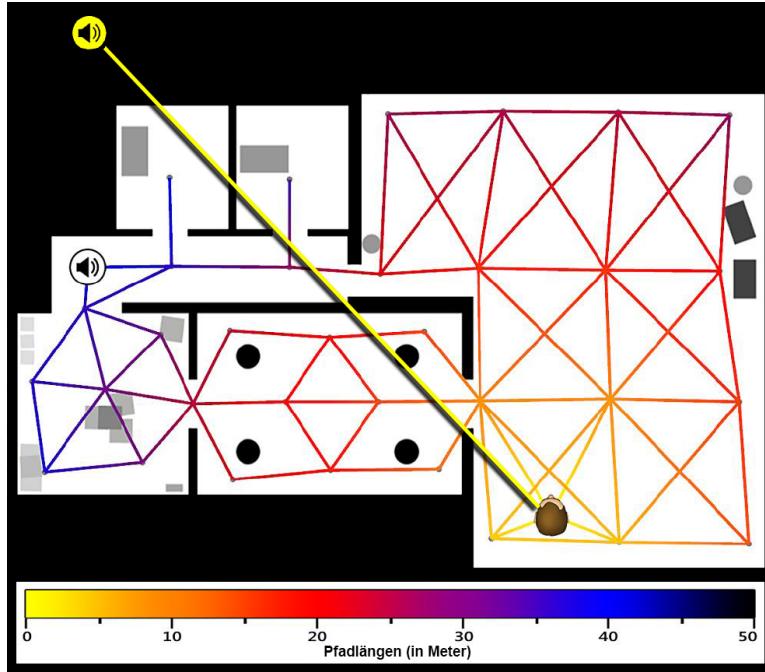


Abbildung 10: Virtuelle Schallquelle in Gelb, berechnet aus der Pfadlänge und Richtung im Graphen. Pfadlängen sind als Farben codiert. In Anlehnung an: [1]

4.4.2 Vor- und Nachteile

Der Vorteil des Frameworks von Cowan liegt in dessen universeller Anwendbarkeit. Das Framework kann sowohl für 2D als auch für 3D Umgebungen angewandt werden und unterstützt dabei auch dynamische Schallquellen- und Zuhörerpositionen. Außerdem muss die Umgebung nicht zwingend durch Geometrie-Daten, wie in den meisten Computerspielen, repräsentiert sein. Zusätzlich ist der Entwickler, im Gegensatz zu *Project Acoustics*, nicht auf eine bestimmte Spiele-Engine bei der Entwicklung oder zur Laufzeit angewiesen. Durch das Einbeziehen der Verdeckung und des Nachhalls, sowie der tatsächlich kürzesten Richtung, welche der Schall nimmt, ist eine räumliche Lokalisierung der Schallquelle möglich. Laut Cowan [29] erhöht sich dadurch die Performance von Spielern in Computerspielen, welche dieses Framework nutzen.

Nachteile des Frameworks sind, dass es die Schallausbreitung nicht physikalisch korrekt berechnet, sondern diese nur approximiert. Es wird nicht wie bei *Project Acoustics* die tatsächliche Wellenphysik zur Kalkulation verwendet. Somit werden weniger natürlich auftretende Phänomene durch das Framework hörbar sein, als bei *Project Acoustics*. Zusätzlich wird immer nur der kürzeste Pfad des Schalles zum Zuhörer für das Rendering verwendet. Der Nachhall wird zwar mittels DSP-Parameter simuliert, allerdings ist dies nur eine Annäherung zum tatsächlichen Nachhall. Ein weiterer Nachteil besteht darin, dass nur statische Geometrie unterstützt wird.

5 Konzeption Graph Audio

Graph Audio ist das im Rahmen dieser Arbeit implementierte Audio-Framework. Grundlegend basiert es auf dem im vorherigen Kapitel 4.4 vorgestellten „Graph-Based Real-Time Spatial Sound Framework“ von Cowan [1]. Dies bedeutet es werden dynamische Schallquellen- und Zuhörerpositionen, aber lediglich statische Szenen-Geometrie unterstützt. Es wurden an einigen Stellen Änderungen oder Erweiterungen gegenüber Cowans Framework vorgenommen, welche in diesem Kapitel präsentiert werden. Umgesetzt worden ist das ganze in *Unity* mit FMOD als Audio Engine, sowie dem *Steam Audio Spatializer*-Plugin für FMOD Studio. Das Plugin bietet die Möglichkeit für alle vorhandenen Parameter, außer den *Reflections* und dem *Pathing*, benutzerdefinierte Werte anzugeben. Zur Schalldämpfung wird ein Abfall genutzt, der durch eine linear quadrierte Kurve, wie es in Abbildung 5 zu sehen ist, modelliert wird. Wie durch Cowan beschrieben wird aber nicht die direkte Distanz zwischen Schallquelle und Zuhörer verwendet, sondern die durch den Graph berechnete Pfadlänge, welche die Szenen-Geometrie mit einbezieht. Für die Verdeckung und Transmission werden sich die manuell im Plugin eintragbaren Werte zunutze gemacht. Über den Graphen wird der Gesamtwert der Verdeckung für ein aktuelles Zuhörer-Schallquellen-Paar nach Cowan, Formel 11, berechnet und dann im *Steam Audio Spatializer* verwendet. Die Werte für niedrige, mittlere und hohe Frequenzen der Transmission werden mit dem Gesamtwert der Verdeckung verknüpft. Die mittleren und hohen Frequenzen skalieren dabei linear eins-zu-eins mit dem Gesamtwert der Verdeckung. Die niedrigen Frequenzen skalieren hingegen verzögert exponentiell. Anschaulich bedeutet dies, dass sie bei höheren Verdeckungs-Werten noch deutlich besser hörbar sind als andere Frequenzen, aber dann rapide abnehmen. Die Verdeckungs-Werte der einzelnen Kanten, welche zur Entwicklungszeit berechnet werden müssen, werden nicht nach Cowans Methode [27], sondern mithilfe von Steam Audio berechnet. Der Nachhall wird im Gegensatz zu Cowan allerdings nicht mithilfe des Graphen berechnet. Hierzu wird der *Reflections*-Parameter des Plugins verwendet, welcher den Hall mithilfe von Raytracing berechnet. Die weiteren Parameter zur Luftabsorption (engl. *Air Absorption*), Richtcharakteristik (engl. *Directivity*), sowie der indirekten Pfade der Pfadfindung (engl. *Pathing*) werden in Graph Audio nicht verwendet.

Neben dem Graph wird zusätzlich eine *Uniform-Grid*-Datenstruktur aufgebaut. Dabei handelt es sich um eine räumliche Datenstruktur, welche die gesamte Szene in ein 3-dimensionales Raster unterteilt, wie es in Abbildung 11 zu sehen ist. Jede Zelle des Uniform-Grid enthält die Knoten des Graphen, dessen Position sich innerhalb dieser Zelle befindet. Genutzt wird diese Datenstruktur, um zur Laufzeit effizient denjenigen Knoten finden zu können, der dem Zuhörer oder den Schallquellen am nächsten liegt. Mit dem Uniform-Grid müssen lediglich die Knoten geprüft werden, die sich innerhalb der Zelle und den Nachbarzellen des Zuhörers befinden, denn diese sind die dem Zuhörer räumlich am nächsten liegenden Knoten. Ohne Datenstruktur müsste die Distanz zu allen Knoten des gesamten Graphen geprüft werden.

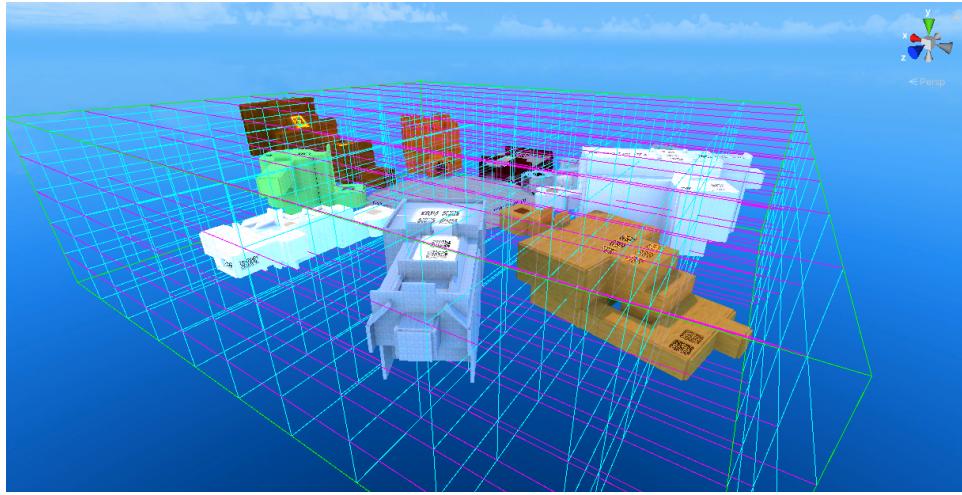


Abbildung 11: Unterteilung der *Project Acoustics Demo*-Szene in ein *Uniform-Grid* mit der Auflösung 10x5x10.

Der letzte Unterschied zu Cowan besteht darin, dass der Graph nicht genutzt wird, wenn es eine direkte Verbindung zwischen Schallquelle und Zuhörer gibt. Dafür gibt es zwei Gründe: Als erstes spart dies Rechenzeit, in dem Fall, dass es keine andere Schallquelle in der Szene gibt, die momentan den Graphen nutzt. Denn dann muss der Graph nicht neu berechnet werden, wenn sich der Zuhörer bewegt. Der zweite Grund liegt in der Auflösung des Graphen. Da es nur eine begrenzte Anzahl an Knoten gibt, kommt es dazu, dass selbst der kürzeste Pfad im Graph länger ist als der direkte Pfad. In Abbildung 12 sind beide Pfade im Vergleich zu sehen. Damit daraus resultierend die Schalldämpfungswerte nicht verfälscht werden, wird der Graph in diesem Fall nicht genutzt, sondern nur der direkte Schall mittels FMOD. Die Folge davon ist allerdings, dass Graph Audio zur Laufzeit abhängig von der Szenen-Geometrie ist und nicht mehr allein mithilfe des Graphen Audio rendern kann, wie es in Cowans Framework möglich ist.

5.1 Graph-Erstellung

Bevor Graph Audio zur Laufzeit genutzt werden kann, muss zunächst zur Entwicklungszeit der Graph für die Szene erstellt werden. Dies kann entweder manuell geschehen oder automatisiert werden. Cowan schlägt in [1] vor ein Raster von Knoten über die Szene zu legen. Dadurch wird zwar die gesamte Szene mit Knoten aufgespannt, allerdings gibt es auch viele Knoten in leeren Stellen oder Wänden, an welche der Schall nicht gelangen kann. Somit werden zur Laufzeit im Dijkstra-Algorithmus viele für die Schallausbreitung unmögliche Pfade berechnet. Um dies zu verbessern wird das Navmesh der Unity-Szene verwendet. Ein Navmesh markiert diejenigen Bereiche in der Szene, die von Charakteren betreten werden können. Um ein solches Navmesh zu erstellen bietet Unity von Haus aus automatisierte

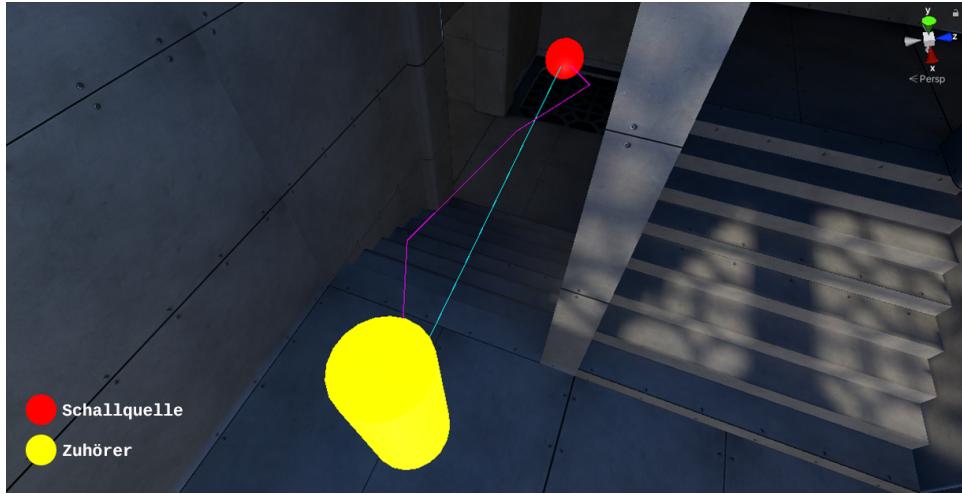


Abbildung 12: Der direkte Pfad zwischen Schallquelle und Zuhörer ist in türkis zu sehen und der kürzeste Pfad des Graphen in pink.

Verfahren an. Während nun in einem Raster über die Szene gegangen wird, wird ein Knoten nur dann erstellt, wenn er sich in der Nähe des Navmesh befindet. Da das Navmesh sich immer auf dem Boden befindet, müssen die Knoten noch ein wenig nach oben verschoben werden. Diese erste, niedrigste Ebene an Knoten bietet dem Schall zur Ausbreitung bis jetzt aber nur Pfade auf Kniehöhe eines Charakters an. Da sich der Schall auch höher in der Luft ausbreiten können muss, wird nun noch eine zweite oder auch dritte Ebene an Knoten hinzugefügt. Die Knoten dieser neuen Ebenen befinden sich immer direkt an denselben Positionen der ersten Ebene und werden nur höher, also mit größerem Y-Achsenwert, positioniert. Anschließend werden Knoten, welche sich in einem festgelegten Radius zueinander befinden, durch Kanten miteinander verbunden. Ein mit dieser Methode erstellter Graph mit zwei Ebenen ist in Abbildung 13 zu sehen. Die Kanten wurden in der Abbildung zwecks einer besseren Übersicht nicht eingezeichnet. Die drastisch reduzierte Anzahl an Knoten ist deutlich erkennbar, denn mit Cowans Methode wäre die gesamte Bounding-Box der Szene, zu sehen als grüner Käfig im Hintergrund, mit Knoten gefüllt gewesen.

Mit der vorgestellten, effizienteren Methode, kann die Anzahl an Knoten je nach Größe der Szene um mindestens das 30-fache reduziert werden. Ein Nachteil dieser Methode besteht allerdings darin, wie die Knoten miteinander verbunden werden. Man möchte den Radius, in dem Knoten durch Kanten miteinander verbunden werden, so wählen, dass nur die direkten Nachbarn verbunden werden. Wählt man den Radius zu groß, entstehen Kanten nicht nur zwischen dem Knoten und seinem ersten Nachbarn in einer Reihe, sondern auch zwischen dem Knoten und seinem zweiten Nachbarn in der Reihe. Dadurch entstehen viele Kanten, welche zwar mitberechnet werden müssen im Dijkstra-Algorithmus, aber keinen Mehrwert liefern, da derselbe Pfad schon von anderen Kanten abgedeckt wird. In Abbildung

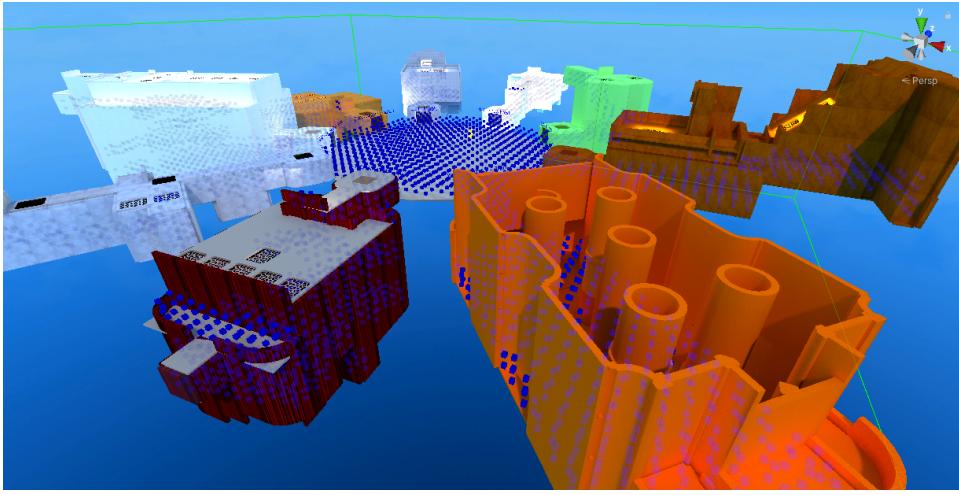


Abbildung 13: Die Knoten der gesamten *Project Acoustics Demo*-Szene als blaue Würfel (insgesamt 5016). Der grüne Käfig im Hintergrund markiert die Bounding-Box der Szene.

14 ist ein Graph mit Knoten und Kanten zu sehen, bei dem der Radius für die Kantenerstellung so eingestellt ist, dass jeweils nur direkte Nachbarn verbunden werden. Das Problem an dieser Variante ist an den beiden rot markierten Knoten zu sehen. Zwischen diesen ist keine Kante vorhanden und dementsprechend kann der Schall über den Graph dort keinen Pfad nehmen und sich nicht ausbreiten über diesen direkten Weg. Physikalisch ist dies allerdings nicht das richtige Verhalten, denn der Schall müsste sich über diesen Weg ausbreiten können. Verbessert werden kann dies, indem die fehlenden Kanten manuell zum Graph hinzugefügt werden.

Bei der Erstellung des Graphen muss die Rechenzeit gegen die Genauigkeit der Simulation abgewogen werden. Je mehr Knoten und Kanten die Szene abdecken, desto genauer kann auch der kürzeste Pfad der Schallausbreitung berechnet werden. Dies geschieht allerdings auf Kosten der benötigten Rechenzeit zur Laufzeit. Denn jede weitere Kante muss vom Dijkstra-Algorithmus in jedem Frame mitberechnet werden.

5.2 Mögliche Erweiterung

In dieser Version von Graph Audio gibt es noch einige Limitationen, welche zukünftig verbessert werden könnten. Die erste Limitation ist, dass lediglich der einzelne kürzeste Pfad für das Audio-Rendering verwendet wird. Dies kann zu Problemen führen, wenn es zwei unterschiedliche Pfade mit nahezu derselben Länge gibt. Betrachte man die Situation in Abbildung 1a. Der obere und untere Pfad haben beide dieselbe Länge und Verdeckungs-Werte. In der Realität würde der Zuhörer nun die Schallquelle gedämpft mittig hinter der Wand wahrnehmen, also dort wo sie tatsächlich steht. In Graph Audio aber würde sich beim Audio-Rendering für

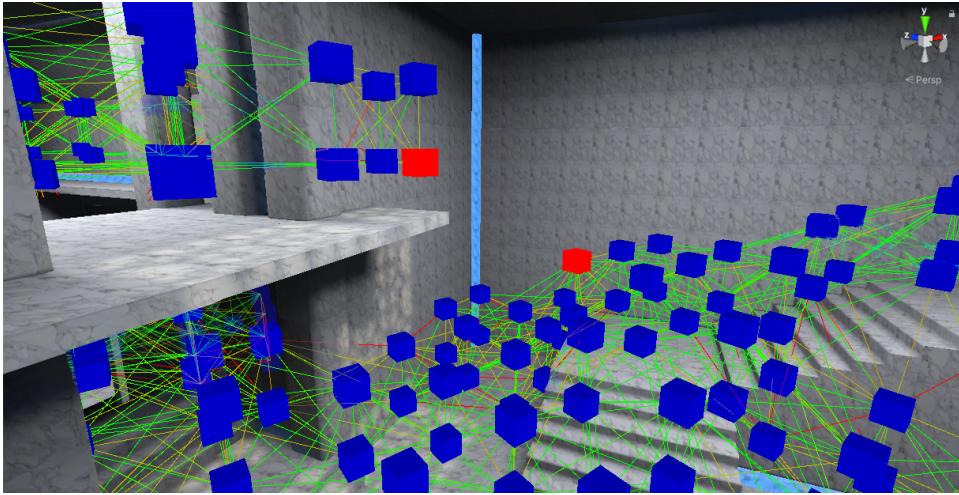


Abbildung 14: Die Kanten zwischen den blauen Knoten sind über einen Farbgradienten nach ihrem Verdeckungs-Wert codiert. Grüne Kanten sind gar nicht verdeckt, während rote Kanten komplett verdeckt sind und keinen Schall durchlassen.

einen der beiden Pfade entschieden werden und die Hinweise zur Lokalisierung würden entweder komplett von rechts oder von links kommen. Des weiteren kann es in dieser Situation dazu kommen, dass die Richtung der Schallquelle ständig zwischen rechts und links wechselt, wenn der Zuhörer immer nur einen Schritt nach links oder rechts macht. Der kürzeste Pfad würde sich dann immer komplett auf die andere Seite ändern und dieser zum Audio-Rendering verwendet werden. Verbessert werden kann das Problem, indem zusätzlich zum besten Pfad, auch der zweit beste Pfad (oder mehr) berechnet werden. Alle berechneten Pfade müssen anschließend gewichtet und gemittelt werden. Die Gewichtung kann abhängig gemacht werden von der Pfadlänge, so dass der kürzeste Pfad den stärksten Einfluss hat.

Eine weitere Limitation ist die, dass keine dynamische Geometrie verwendet werden kann. Der Grund liegt darin, dass die Verdeckungs-Werte der Kanten sich ändern, wenn beispielsweise eine Tür sich öffnet, durch die eine Kante verläuft. Da die Verdeckungs-Werte zur Entwicklungszeit berechnet werden, müsste also der komplette Graph neu kalkuliert werden. Eine Variante dies zu verbessern, ist nur diejenigen Kanten neu zu berechnen, die auch durch die sich verändernde Geometrie beeinflusst werden. Da in den meisten Fällen nur wenige Kanten betroffen wären, könnte dies auch zur Laufzeit geschehen.

6 Implementation Graph Audio

Dieses Kapitel geht auf die Details ein, die zur Implementation von Graph Audio in Unity nötig gewesen sind. Das gesamte Audio-Rendering wird dabei auf der CPU durchgeführt. Um die benötigte Rechenzeit pro Frame dabei so gering wie

möglich zu halten, wird *Unity DOTS* (Datenorientierter Technologie-Stack) [30] über das *Unity Job*-System [31] verwendet. DOTS bietet die Möglichkeit sicheren Multithread-Code zu schreiben, welcher die Rechenleistung aktueller Mehrkern-Prozessoren ausnutzt und so beachtliche Leistungssteigerungen ermöglicht. Dazu wird ein datenorientierter Ansatz verwendet, was bedeutet, dass keine Referenztypen (zb. Objekte von Klassen) mehr verwendet werden, sondern nur reine Datentypen (zb. integer, floats). Innerhalb des Unity-Workflows kann DOTS über das Job-System verwendet werden. Ein Job führt meist einzelne zeitkritische Methoden aus und kann innerhalb der normalen Objektorientierten Unity-Pipeline gestartet werden. In Graph Audio werden Jobs genutzt, um zur Laufzeit die rechenintensiven pro Frame durchgeführten Methoden zu berechnen. Was genau mithilfe des Job-Systems berechnet wird, ist im Verlauf folgenden Kapitel 6.2 erläutert. Auf der *Project Acoustics Demo*-Szene konnten während der Nutzung von Graph Audio durchschnittliche 558 Frames pro Sekunde mit einer *Intel 8700K* CPU, welche auf 5 Ghz getaktet ist, einer *Nvidia Geforce 1080TI* Grafikkarte sowie 32 GB DDR4 Arbeitsspeicher erreicht werden. Der genutzte Graph besteht in dieser Szene aus insgesamt 5016 Knoten und 36916 Kanten. Die genauen Ergebnisse des Benchmark-Test können in Anhang 8.3 eingesehen werden. Im folgenden wird die gesamte Funktionsweise von Graph Audio Teil für Teil durchgegangen. Wie bereits in Kapitel 4.4 erläutert, kann diese in den einmalig zur Entwicklungszeit durchgeführten *Baking*-Schritt und den zur Laufzeit pro Frame durchgeführten *Rendering*-Schritt aufgeteilt werden.

6.1 Baking

Zur Entwicklungszeit müssen zwei Datenstrukturen erstellt werden. Zunächst der Graph, welcher nach der in Abschnitt 5.1 erläuterten Methode automatisch erstellt wird. Bei dieser werden allerdings zunächst nur die Knoten generiert und mit Kanten verbunden. Jeder Knoten speichert seine 3D-Position in Weltkoordinaten, sowie eine Liste aller Kanten mit denen er verbunden ist. Die Kanten speichern beide Knoten die sie verbindet, sowie ihre Länge (Distanz zwischen beiden Knoten) und ihr Verdeckungs-Wert. Die Verdeckungs-Werte müssen allerdings noch kalkuliert werden. Diese geben an, wie stark der Weg zwischen den beiden durch die Kante verbundenen Knoten durch Szenen-Geometrie verdeckt ist. Ein Wert von 0 bedeutet, dass keine Verdeckung auf diesem Pfad vorhanden ist, während 1 ein komplett verdeckter Pfad ist. Berechnet werden diese Verdeckungs-Werte mithilfe der Unity-Integration von Steam Audio. Dieses kann die Verdeckung zwischen einer Schallquelle und dem Zuhörer berechnen. Dazu wird ein volumetrisches Raytracing verwendet [32] bei dem 32 Strahlen vom Zuhörer in Richtung Schallquelle, welche einen Radius von 1 besitzt, verschossen werden. Der Anteil an Strahlen, der durch Szenen-Geometrie verdeckt ist und die Schallquelle nicht erreicht, bestimmt den resultierenden Verdeckungs-Wert. Für die Berechnung des Wertes einer einzelnen Kante wird sich Steam Audio nun zunutze gemacht. Dazu wird der Zuhörer an Position des ersten Knoten der Kante und die Schallquelle an Position des zweiten

Knoten gesetzt. Für diese Konstellation wird nun der Verdeckungs-Wert berechnet. Anschließend werden die Zuhörer und Schallquellen Position getauscht und noch einmal der Verdeckungs-Wert berechnet. Beide Werte werden dann gemittelt. Hier zu beachten ist, dass im Gegensatz zu Graph Audio, bei Steam Audio ein Wert von 0 komplette Verdeckung bedeutet. Der Graph Audio Verdeckungs-Wert O_G wird deshalb über den Steam Audio Verdeckungs-Wert O_S berechnet mit folgender Umrechnung:

$$O_G = 1 - O_S \quad (12)$$

Dieses Vorgehen wird nun für jede Kante wiederholt. Der Graph wird anschließend als *Unity-Asset* gespeichert.

Da zur Laufzeit das Job-System mit Unity DOTS verwendet wird, muss der so gespeicherte Graph beim Laden der Szene aber noch in eine DOTS kompatible Variante konvertiert werden. Für diese dürfen lediglich Datentypen verwendet werden. Das bedeutet, dass die Nachbarn eines Knoten nicht mehr über einfache Referenzen, wie im Unity-Asset, gespeichert werden können. Stattdessen werden in der DOTS-Graph-Variante Indizes verwendet. Der DOTS-Graph besteht aus 3 Datenstrukturen: Einem *Native-Array* mit allen Knoten, einem *Native-Array* mit allen Kanten und einer *Native-Multi-Hash-Map*, die zu jedem Knoten-Index (Schlüssel) alle Kanten-Indizes (Werte) speichert, mit denen er verbunden ist. Die Indizes beziehen sich dabei immer auf die *Native-Arrays*. Ein DOTS-Knoten ist eine Struktur, welche 3D-Position und ihren Index im Native-Array speichert. Zusätzlich werden folgende Werte durch den Dijkstra-Algorithmus zur Laufzeit berechnet und gespeichert: Den Richtungsvektor zum Vorgänger-Knoten, den Index des Vorgänger-Knoten und die totale Pfadlänge zum Startknoten, in diesem Fall also die Schallenergie, die diesen Knoten erreicht. Eine DOTS-Kante ist eine Struktur, welche die beiden Indizes der Knoten speichert, die durch diese Kante verbunden werden. Außerdem wird der Schalldämpfungs-Wert D_a gespeichert, welcher der durch Verdeckung erhöhten Distanz entspricht und wie in Abschnitt 4.4.1 erläutert berechnet wird. Dabei muss beachtet werden, dass die in den Kanten im Unity-Asset gespeicherten Verdeckungs-Werte O_c als Werte zwischen 0 und 1 gespeichert sind. Diese müssen noch umgerechnet werden auf den Wertebereich 0 bis 255.

Die zweite Datenstruktur die erstellt werden muss ist das *Uniform-Grid*. In der verwendeten *Project Acoustics Demo*-Szene hat das Uniform-Grid eine Auflösung von $64 * 6 * 64$ in XYZ-Richtung. Das Grid wird beim Laden der Szene einmalig mit den Knoten des Graphen gefüllt. Dafür muss zunächst eine *Axis-Aligned-Bounding-Box* (kurz AABB) um die gesamte Szene gespannt werden. Der minimale Punkt der AABB stellt gleichzeitig auch den Ursprung des Uniform-Grid Koordinatensystems da. Innerhalb dieses hat jede einzelne Zelle des Uniform-Grid eine einmalige Zellen-ID. In einem Grid der Größe s kann zu einer Position p , welche sich innerhalb der AABB und im Grid-Koordinatensystem befindet, die Zellen-ID über folgende Formel berechnet werden:

$$ID = p_z^{grid} * s_y * s_x + p_y^{grid} * s_x + p_x^{grid} \quad (13)$$

Vom Unity Weltkoordinatensystem kann ein Punkt p wie folgt in das Grid-Koordinatensystem umgewandelt werden:

$$p^{grid} = \lfloor (p^{welt} - u^{welt}) / z^{welt} \rfloor \quad (14)$$

Hier gibt u den Ursprung des Grid-Koordinatensystem und z die Größe einer Zelle, beide in Weltkoordinaten, an. Mit Hilfe beider Formeln kann über jeden Knoten iteriert und die Zellen-ID berechnet werden, in die dieser Knoten gehört. Für jede Zellen-ID gibt es eine Liste, zu welcher der Knoten hinzugefügt wird. Nachdem alle Knoten durchgegangen wurden, enthält die Liste einer Zelle alle Knoten, welche sich innerhalb der Zelle befinden. Die Formeln für Zellen-ID und Umrechnung sind aus einer GPU Implementation eines Uniform-Grid von Nvidia [33] entnommen.

6.2 Laufzeit

Für die Basis des Audio-Renderings wird das vom FMOD *Steam Audio Spatializer*-Plugin bereitgestellte HRTF-basierte binaurale Rendering verwendet. Dieses bezieht die fürs Rendering benötigte Distanz zwischen Schallquelle und Zuhörer, sowie die Richtung vom Zuhörer zur Schallquelle direkt über das Unity *GameObject* der Schallquelle. Da Graph Audio aber nicht die tatsächliche Schallquellen-Position, sondern eine wie in Abschnitt 4.4.1 beschriebene virtuelle Schallquelle verwendet, muss ein zusätzliches *GameObject* für die virtuelle Schallquelle erstellt werden. Von diesem bezieht FMOD dann die Positions- und Richtungsdaten. Da die Position der virtuellen Schallquelle unter Umständen jedes Frame verschoben wird, denn sie ist abhängig von der Zuhörer-Position, kann es zu einem „Springen“ der wahrgenommenen Richtung kommen. Um dies weitestgehend zu vermeiden, wird zwischen einer alten und neuen virtuellen Schallquellen-Position linear interpoliert.

Der *Graph Audio Manager* ist für alle Einstellungen und Berechnungen, die außerhalb von FMOD getätigten werden müssen, zuständig. Dazu wird die Unity *FixedUpdate*-Methode genutzt. Diese ist die in die Unity-Pipeline integrierte, Bildraten unabhängige Renderschleife für Physiksimulationen. Im Gegensatz zur normalen, unlimitierten *Update*-Methode, wird *FixedUpdate* immer in festen, gleichbleibenden Zeitintervallen von 50 Aufrufen pro Sekunde aufgerufen [34]. Im Pseudo-Code 15 ist der Ablauf der *FixedUpdate*-Methode dargestellt.

Zunächst wird geprüft, ob sich der Zuhörer seit dem letzten Frame bewegt hat (Zeile 2). Ist dies nicht der Fall gilt immer noch der kürzeste Pfad aus dem letzten Frame und es muss nichts neu berechnet werden. Als nächstes wird für jede Schallquelle in der Szene mithilfe eines einfachen *Linecast*³ geprüft, ob sich zwischen dieser Schallquelle und dem Zuhörer Szenen-Geometrie befindet (Zeile 7). Ist dies der Fall, wird die entsprechende Schallquelle markiert (Zeile 8), um später mithilfe des Graphen das Audio-Rendering durchzuführen. Andernfalls besteht eine direkte Verbindung zwischen Schallquelle und Zuhörer, weshalb der Graph nicht verwendet wird (Zeile 10-11). Gibt es in der gesamten Szene keine Schallquelle, welche

³Verschießt einen Strahl, um zu prüfen ob sich Geometrie zwischen Start- und Endpunkt befindet.

```

1   function fixedUpdate()
2       if (distanz zwischen zuhörerAlt und zuhörerNeu kleiner gleich 0,1)
3           return
4
5       initialisiere leere schallquellenFürGraph Liste
6       foreach (schallquelle in Szene)
7           if (direkter Weg zwischen schallquelle und zuhörerNeu blockiert)
8               füge schallquelle bei schallquellenFürGraph hinzu
9
10      else
11          verschiebe virtuelleSchallquelle mit linearer Interpolation nach schallquelle
12          setze Parameter Verdeckung von Steam Audio Plugin auf 1 (nicht verdeckt)
13
14      if (schallquellenFürGraph Liste leer)
15          return
16
17      hole knotenNaheZuhörerNeu Liste mithilfe des Uniform–Grid
18      finde nächsterZuhörerKnoten aus knotenNaheZuhörerNeu mit Unity Job–System
19      berechne modifizierten Dijkstra von nächsterZuhörerKnoten startend mit Unity Job–System
20
21      foreach (schallquelle in schallquellenFürGraph)
22          hole knotenNaheSchallquelle Liste mithilfe des Uniform–Grid
23          finde nächsterSchallquellenKnoten aus knotenNaheSchallquelle mit Unity Job–System
24          berechne virtuelleSchallquelleNeu aus nächsterSchallquellenKnoten
25          verschiebe virtuelleSchallquelle mit linearer Interpolation nach virtuelleSchallquelleNeu
26          berechne gesamtwertVerdeckung
              setze Parameter Verdeckung von Steam Audio Plugin auf gesamtwertVerdeckung

```

Abbildung 15: Pseudo-Code der *FixedUpdate*-Methode innerhalb des *Graph Audio Manager*. Zuständig für das Audio-Rendering außerhalb von FMOD.

den Graphen in diesem Frame verwendet, dann wird dieser nicht neu berechnet (Zeile 13). Im restlichen Teil der Methode werden alle Schallquellen abgearbeitet, die mithilfe des Graphen die Position der virtuellen Schallquelle, sowie die Verdeckungs- Werte berechnen. Dazu muss zunächst der Knoten gefunden werden, der dem Zuhörer am nächsten liegt. Um dafür nicht die Distanz zwischen jedem einzelnen Knoten des Graphen und dem Zuhörer ausrechnen zu müssen, werden nur die Knoten geprüft, die sich räumlich in der Nähe des Zuhörers befinden. Diese Knoten werden mithilfe des Uniform-Grids erlangt (Zeile 16). Zunächst wird die Zell-ID, in der sich der Zuhörer befindet, mithilfe der Formeln 13 und 14 berechnet. Geprüft werden nun alle Knoten, die sich in dieser Zelle befinden, sowie in allen Nachbarzellen in einem Würfel um die Zelle herum, zu sehen in Abbildung 16. Die Nachbarzellen werden auch mitbetrachtet, da es vorkommen, dass Knoten sich an der direkten Grenze zweier Zellen befinden und dann in der Nachbarzelle gespeichert werden. Der tatsächlich am nächsten liegende Knoten wird anschließend mithilfe von Unity DOTS und dem Job-System gefunden (Zeile 17). Dieser Knoten wird dann als Startknoten für den modifizierten Dijkstra-Algorithmus verwendet (Zeile 18). Er berechnet für jeden einzelnen Knoten im Graphen den kürzesten Pfad zum Startknoten des Zuhörers. Hierbei handelt es sich um den rechen-intensivsten Schritt des Audio-Rendering, weshalb auch hier das Job-System verwendet wird. Wie bereits in Kapitel 4.4.1 erläutert, muss durch die Nutzung des modifizierten

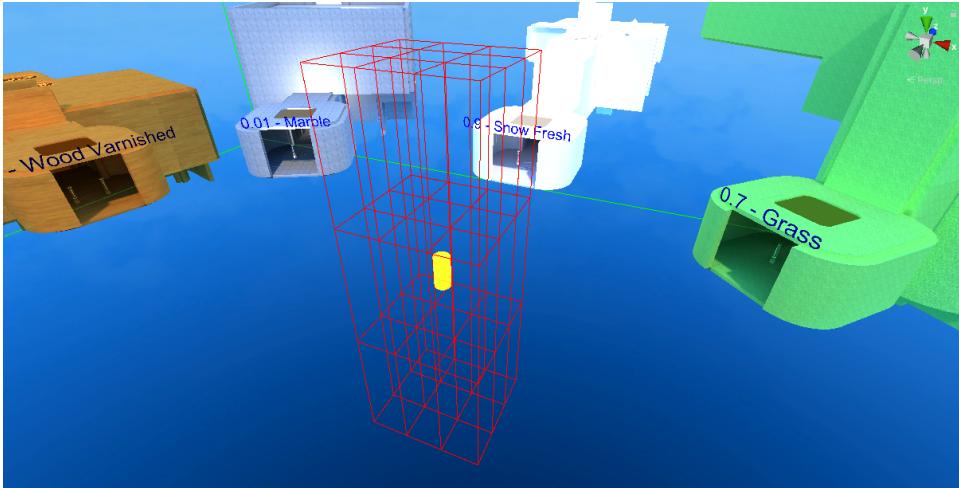


Abbildung 16: Zelle des Zuhörers (gelb) im Uniform-Grid, sowie alle Nachbarzellen in einem Würfel um diese herum.

Dijkstras der Algorithmus nur einmal pro Frame und nicht für jede Schallquelle einzeln durchgeführt werden. Als nächstes werden alle Schallquellen, die den Graphen nutzen, abgearbeitet (Zeile 20). Zunächst wird mit derselben Methode wie für den Zuhörer, der Knoten gefunden, welcher der Schallquelle am nächsten liegt (Zeile 21-22). In diesem Knoten ist dann der kürzeste Pfad zum Startknoten gespeichert. Mithilfe des gespeicherten Index zum Vorgängerknoten kann der Pfad zum direkt mit dem Zuhörer verbundenen Knoten verfolgt werden. In Abbildung 17 ist ein so berechneter Pfad visualisiert. Mithilfe dieser Pfadlänge P und der nach Formel 10 gespeicherten Richtung V_n des direkt verbundenen Knoten wird die neue Position der virtuellen Schallquelle VS_p^{neu} wie folgt berechnet (Zeile 23):

$$VS_p^{neu} = L_p + V_n * P \quad (15)$$

Hierbei ist L_p die aktuelle Position des Zuhörers. Für die tatsächlich genutzte Position der virtuellen Schallquelle wird dann zwischen VS_p^{alt} und VS_p^{neu} linear interpoliert (Zeile 24). Der letzte fehlende Teil ist das Setzen des Verdeckungs-Parameters im *Steam Audio Spatializer*-Plugin. Der benötigte Wert O wird mit der folgenden Formel berechnet (Zeile 25):

$$O = \left(\frac{D}{P} \right)^2 \quad (16)$$

Es handelt sich bei D um die direkte Distanz zwischen Schallquelle und Zuhörer, während P die durch den Graphen berechnete Pfadlänge ist. Diese Formel ist die für das *Steam Audio Spatializer*-Plugin modifizierte Variante von Cowans Formel 11. Dieser Verdeckungs-Wert O muss zum Schluss noch an das Plugin übertragen werden (Zeile 26).

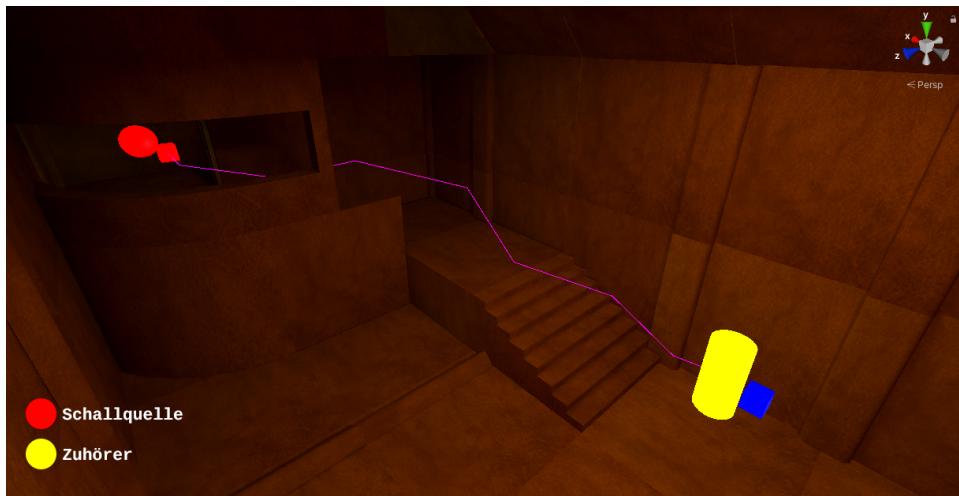


Abbildung 17: Vom Graphen berechneter kürzester Pfad zwischen Zuhörer und Schallquelle.

7 Evaluation von Akustik-Frameworks in Bezug auf Lokalisierung von Schallquellen

In der Einleitung dieser Arbeit ist die Leitfrage „Resultiert eine realistischere Schallausbreitung eines Audio-Framework in einer erhöhten Lokalisierbarkeit von Schallquellen?“ aufgestellt worden. Hergeleitet wurde diese Frage über die Feststellung, dass die großen Shooter-Titel „Counter-Strike:Global Offensive“ und „Valorant“ nur ein simples Panning mit wahlweise HRTF-Filtern zum Audio-Rendering verwenden. Methoden die keine realistische Schallausbreitung bieten. Ist dies von den Entwicklern beabsichtigt oder kann ein Framework mit realistischerer Schallausbreitung eine Verbesserung für die Lokalisierbarkeit von Schallquellen bedeuten? Um dies beantworten zu können, wurde im Rahmen dieser Arbeit eine Evaluation zum Vergleich von vier verschiedenen Audio-Frameworks durchgeführt: Die drei kommerziellen Frameworks *FMOD*, *Steam Audio* und *Project Acoustics* sowie das selbst implementierte *Graph Audio*. Dabei nutzt FMOD ein simples Audio-Rendering, welches sehr ähnlich zu denen der obigen Titeln ist. Die beiden Frameworks Steam Audio und Graph Audio gehören zu denen der geometrischen Akustik und nutzen Raytracing um eine realistische Schallausbreitung zu approximieren und sind somit in der Theorie realistischer als FMOD. Project Acoustics geht noch einen Schritt weiter. Als Verfahren der numerischen Akustik berechnet es die tatsächliche Ausbreitung der physikalischen Schallwellen in der Szene und ist somit am nächsten an der Realität.

Für den Vergleich wurde ein eigens entwickeltes Testprogramms mit dazugehörigem Fragebogen verwendet. Die Zielgruppe sind Personen, die bereits generelle Erfahrung mit Videospielen gemacht haben. Insgesamt wurde der Test mit 17 Probanden durchgeführt, von welchen aber zwei den Test aufgrund von Übelkeit

abbrechen mussten. Diese wurde durch schwere Fälle der Reisekrankheit (engl. *Motion Sickness*) hervorgerufen. Die abgebrochenen Tests sind nicht zu den Ergebnissen gezählt worden, womit 15 Tests für die Ergebnisauswertung verwendet wurden. Alle Probanden verwendeten als Peripheriegeräte Maus, Tastatur sowie Stereo-Kopfhörer für die Audioausgabe.

7.1 Aufbau Testumgebung

Das Ziel der Testumgebung ist es, die vier Frameworks unter gleichen Bedingungen in einem Szenario vergleichen zu können, welches ähnlich zu den zuvor genannten Videospielen ist. Dazu wurde als Testprogramm ein kleines Spiel implementiert, in dem die Probanden einen Charakter in First-Person-Ansicht steuern müssen. Für dieses wird die *Project Acoustics Demo*-Szene verwendet, zu sehen in Abbildung 13 und 11 sowie in allen weiteren Abbildungen aus den Graph Audio Kapiteln. Diese Szene wurde von Microsoft als Demo für Project Acoustics erstellt und bietet viele verschiedene Architekturen, an denen sich der Schall ausbreiten kann. Gleichzeitig ist sie übersichtlich und nicht zu groß, so dass sich die Teilnehmer nicht verlaufen können während des Tests. Die Lautstärke und Distanz über die Schallquellen gehört werden können, also die Schalldämpfung, ist für alle Frameworks gleich eingestellt. Es werden linear quadrierte Abfallkurven mit einer minimalen Distanz von 1 und einer maximalen Distanz von 75 Einheiten verwendet, wie es in Abbildung 4 zu sehen ist. Dies bedeutet auch, dass Schallquellen einen Radius von 1 besitzen. Die Situation die über den Test simuliert werden soll, ist die folgende: Der Zuhörer hört eine Schallquelle ohne sie sehen zu können und versucht nun diese so schnell es geht anhand der akustischen Hinweise zu lokalisieren. Dies wurde versucht in der grundlegenden Mechanik des Tests zu modellieren: Es wird eine unsichtbare Schallquelle in der Szene gespawnt. Die Testperson muss nun versuchen die Position der Schallquelle so schnell es geht anhand der akustischen Hinweise zu erraten. Dazu kann sie eine Kugel als Marker platzieren, welche sie so oft sie will verschieben kann, bevor sie ihre Vermutung endgültig bestätigt. Die Testperson muss nun durch die Szene laufen und versuchen, diesen Marker an die Position zu setzen, an der sie die unsichtbare Schallquelle vermutet. Bestätigt sie die durch den Marker geschätzte Position der Schallquelle, wird automatisch als Ergebnis die benötigte Zeit und die Distanz zwischen Marker und tatsächlicher Schallquellenposition in einer Textdatei gespeichert. Nach dem Bestätigen wird die Schallquellenposition für die Testperson aufgedeckt. In Abbildung 18 ist die Oberfläche des Testprogramms nach Bestätigen zu sehen. Der Marker ist die blaue Kugel und die tatsächliche Schallquelle, welche während des Tests unsichtbar war, die rote Kugel. Im oberen Bereich wird die erreichte Distanz sowie die benötigte Zeit angezeigt. Es gibt zwei verschiedene Audio-Clips die Dauerschleife abgespielt werden. Zum einen Fußstapfen und zum anderen Musik. Während die Fußstapfen näher an der zu simulierenden Ausgangssituation sind, bietet die Musik deutlich mehr verschiedene Frequenzen, wodurch Akustik-Effekte deutlich besser hörbar sind. Außer der zu findenden Schallquelle gibt es keine weiteren Schallquellen,

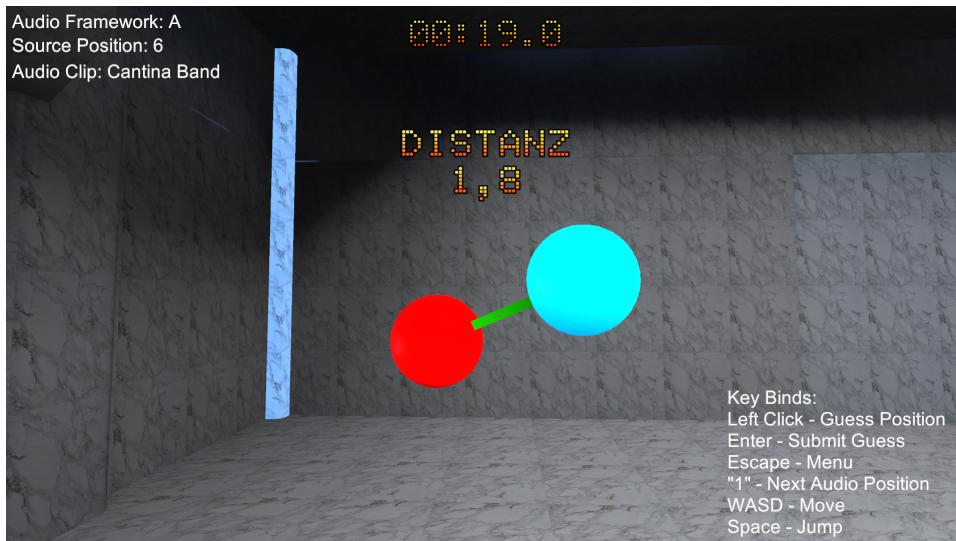


Abbildung 18: Oberfläche und Mechanik des Testprogramms. Die blaue Kugel ist der vom Nutzer platzierte Marker und die rote Kugel ist die tatsächliche Schallquelle.

um die Testperson nicht abzulenken. Das heißt auch die Bewegung des Charakters verursacht keine Fußstapfen. Das zum Audio-Rendering verwendete Framework der vier zur Auswahl stehenden, kann zur Laufzeit frei gewechselt werden.

7.2 Aufbau Fragebogen

Zusätzlich zum Testprogramm gibt es einen Fragebogen, den die Probanden ausfüllen mussten. Der gesamte Fragebogen ist im Anhang 8.3 einsehbar. Im folgenden werden nicht alle Fragen im Detail besprochen, sondern nur bestimmte Fragen, warum sie in dieser Form gestellt worden sind und welche Überlegungen dahinter stecken. Zunächst wird abgefragt, ob die Testperson mit der Charaktersteuerung des Testprogramms (Laufen mit WASD und Maus zur Orientierung) bekannt war. Ist dies nicht der Fall, kann davon ausgegangen werden, dass die erhobenen Zeiten zum Finden einer Schallquelle generell höher ausfallen. Dies ist damit zu begründen, dass Personen dann unter Umständen ihren Charakter nicht sofort dorthin bewegen können wo sie ihn gerne hätten. Zu selbigem Zweck wird auch die Erfahrung mit Videospielen, aufgeteilt in PC, Konsole und Handy, abgefragt. Eine Person die selten auf einem PC spielt, kennt die Steuerung vermutlich ebenfalls schlechter. Es folgt die Frage, wie häufig welche Genres an Videospielen gespielt werden. Die Intention hier ist es, herauszufinden ob Testperson schon Erfahrung mit First- und Third-Person-Shootern haben. Ist dies der Fall, kennen sie zum einen bereits die Charaktersteuerung und zum anderen wird ihnen das System des Lokalisieren von Geräuschen über akustische Hinweise bekannt sein. Dies kann sowohl Einfluss auf die benötigte Zeit, als auch die erreichte Distanz beim finden einer Schallquelle haben. Beide zuvor genannten Fragen besitzen die Wahlmöglichkeit von *Nie* bis hin

zu *Oft* in vier Schritten. Es gib also keine neutrale Antwort, denn diese würden für die Auswertung nicht hilfreich sein.

Als nächstes gibt es vier Fragen, welche sich auf ein Framework aus dem Testprogramm beziehen. Diese werden in exakt derselben Form für jedes einzelne der vier Frameworks gestellt. Sie sollten aus dem eigenen Empfinden der Testperson heraus beantwortet werden ohne dabei auf die automatisch bezogenen Ergebnisse zu achten. Letztere wurden den Testpersonen nicht als Übersicht gezeigt vor dem Beantworten der Fragen. Die vier Fragen sind die folgenden:

1. Wie **schnell** konntest du die Schallquellen lokalisieren?
2. Wie **präzise/genau** konntest du die Schallquellen lokalisieren?
3. Wie **realistisch** empfandest du den Sound?
4. Empfandest du etwas als hilfreich oder störend?

Die zwei ersten Fragen müssen separat für die beiden möglichen Audio-Clips Fußstapfen und Musik beantwortet werden. Sie bieten dazu wieder 4 Antwortabstufungen von *Langsam/Schlecht* bis zu *Schnell/Gut*, so dass es keine neutrale Antwortmöglichkeit gibt. Frage drei hingegen bietet eine fünfteilige Skala von *unrealistisch* bis hin zu *realistisch*. Hier wurde sich für die Möglichkeit einer neutralen Antwort entschieden, da es schwerfallen kann einzuschätzen, was nun realistischer Sound in einem Videospiel überhaupt ist.

Zum Schluss sollen noch die vier Frameworks in einer Rangliste von Platz eins bis vier gegeneinander verglichen werden. Dieser Vergleich ist wieder in drei Teile aufgeteilt und zwar zu denselben Fragestellungen, wie die ersten drei Fragen in obiger Aufzählung: Schnelligkeit bei der Lokalisierung, Präzision/Genauigkeit bei der Lokalisierung und Realismus des Sounds.

7.3 Testablauf

Als erstes muss die Testperson die allgemeinen Fragen des Fragebogens beantworten. Daraufhin startet sie das Testprogramm und bekommt die Möglichkeit die Mechanik des Suchens von Schallquellen ein paar mal mit zufällig gespawnten Schallquellen auszuprobieren. Hierbei kann die Testperson auch den Grundriss der Szene kennenlernen. Fühlt sie sich mit der Steuerung und Mechanik des Testprogramms vertraut kann der eigentliche Test gestartet werden. Dieser ist aufgeteilt in vier Abschnitte, jeweils für jedes Framework einen Abschnitt. In jedem Abschnitt werden nacheinander 10 Schallquellen gespawnt. Zuerst fünf mit Musik als Audio-Clip und anschließend fünf mit Fußstapfen als Audio-Clip. Es ist immer nur eine einzelne Schallquelle gleichzeitig aktiv in der Szene, welche die Testperson nach obiger Mechanik finden muss. Wurde der Marker bestätigt, kann die nächste Schallquelle gespawnt werden, wobei die Testperson immer von der Position der vorherigen Schallquelle aus startet. Letzteres gewährleistet eine bessere Vergleichbarkeit mit den Ergebnissen anderer Probanden. Ist ein Abschnitt abgeschlossen

muss die Testperson die Fragen des Fragebogens zu dem für diesen Abschnitt verwendeten Framework beantworten. Anschließend wiederholt sich dies mit dem nächsten Abschnitt und nächsten Framework, bis alle Frameworks getestet wurden, also insgesamt 40 Schallquellen. Die Namen der Frameworks werden den Testpersonen erst beim Beantworten der Fragen genannt, damit diese nicht beeinflusst werden, falls eine Testperson ein genutztes Framework schon kennt. Im Testprogramm heißen die Frameworks nur *A*, *B*, *C* und *D*. Sind alle vier Abschnitte, also alle Frameworks, getestet, müssen abschließend noch die Fragen zum Vergleich der Frameworks beantwortet werden.

Damit die automatisch erhobenen Ergebnisse so gut es geht miteinander verglichen werden können, soll die Varianz zwischen den Abläufen der einzelnen Tests möglichst gering sein. Aus diesem Grund werden die Schallquellen nicht zufällig in der Szene gespawnt. Stattdessen gibt es vier verschiedene Varianten mit jeweils 10 festen Positionen. Für den Test kann nicht nur eine einzelne Variante an Positionen genutzt werden, da Probanden sich sonst beim Durchlauf mit Framework *A* die Positionen merken können und diese dann für die restlichen Durchläufe direkt wissen würden. Damit nicht jedes Framework immer die gleiche Variante an Positionen nutzt, rotieren die Varianten zwischen den Probanden durch. Das heißt wenn Proband Peter Framework *A* mit Variante 1, *B* mit 2, *C* mit 3 und *D* mit 4 testet, dann testet danach Proband Laura Framework *A* mit Variante 2, *B* mit 3, *C* mit 4 und *D* mit 1. Die Ergebnisse innerhalb einer einzelnen Variante können gut miteinander verglichen werden, denn dort sind dieselben Positionen mit allen vier Frameworks getestet. Somit ist die äußere Beeinflussung für Zeit und Distanz minimal.

7.4 Ergebnisse

Die Ergebnisse der Evaluation werden auf verschiedene Wege ausgewertet. Zunächst werden die Fragebögen und die Daten des Testprogramms separat ausgewertet. Alle erhobenen Ergebnisse sind im Anhang für die Fragebögen 8.3 sowie für die direkt durch das Testprogramm erhobenen Daten 8.3 einzusehen. Im Anschluss werden beide in Verbindung zueinander gestellt, wodurch geprüft werden kann, ob die Daten auch die in den Fragebögen getätigten Einschätzungen der Probanden bestätigen.

Fragebögen. Zunächst kann festgestellt werden, dass die für die Evaluation gewünschte Zielgruppe getroffen wurde. Insgesamt waren 93,33% aller Teilnehmer bereits mit der Charaktersteuerung des Testprogramms vertraut. Dadurch entfällt für erhöhte gemessene Zeiten eine unbekannte Steuerung als Grund. Dies wird dadurch unterstützt, dass alle Teilnehmer, außer einem, regelmäßig an einem PC Videospiele spielen, wie in den Umfrageergebnissen in Abbildung 19 erkennbar ist. Auch spielen etwa 73% der Probanden relativ regelmäßig First- oder Third-Person-Shooter, zu sehen in Abbildung 20. Dies gibt den Probanden den Vorteil, dass sie bereits mit der Mechanik des Lokalisieren von Objekten über Akustische-Hinweise bekannt sind.

Wie häufig spielst du Videospiele am PC?

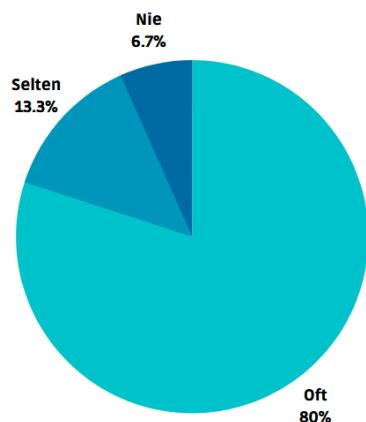
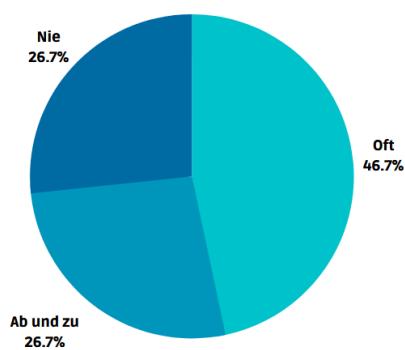


Abbildung 19: Ergebnis der Umfrage zur Häufigkeit von PC-Videospielen aller Probanden zusammen.

Wie häufig spielst du First-Person-Shooter?



Wie häufig spielst du Third-Person-Shooter?

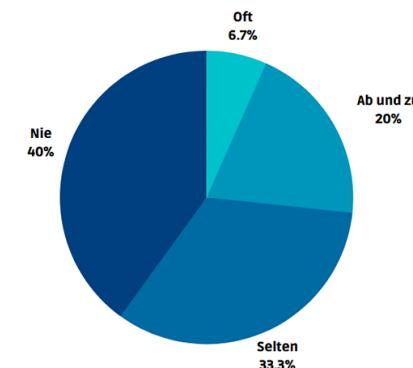


Abbildung 20: Ergebnis der Umfrage zur Häufigkeit von FPS und TPS Videospielen aller Probanden zusammen.

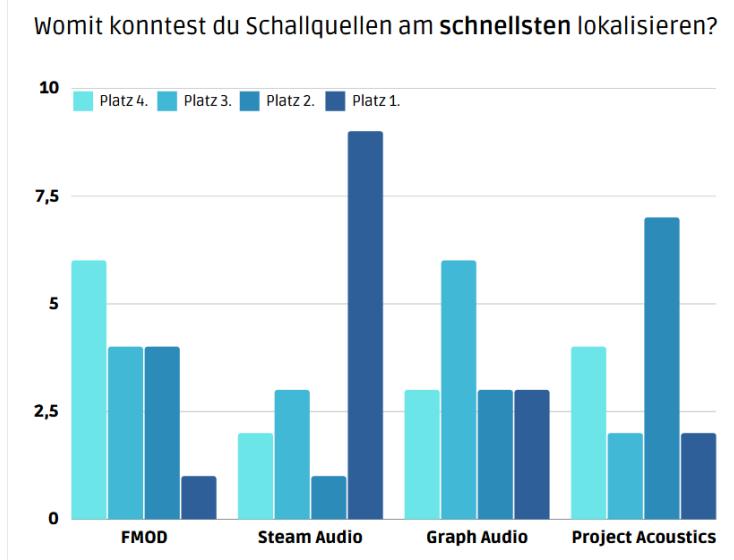


Abbildung 21: Ergebnis der Umfrage zur Schnelligkeit beim Vergleich der vier Frameworks aller Probanden zusammen.

Die wichtigsten Fragen dieses Fragebogens sind die Vergleiche der vier Frameworks untereinander, welche hier in Form einer Rangliste getätigt werden sollten. Teilweise tun sich hier starke Unterschiede zwischen einzelnen Probanden auf, bei welchen beispielsweise Person A FMOD am besten fand und Steam Audio am schlechtesten, während Person B dies genau anders herum sah. In der Gesamtheit aller Probanden zeichnet sich bei den Fragen zur Schnelligkeit der Lokalisierung, sowie zum Realismus, dennoch ein klares Bild, welches in Abbildung 21 und 22 zu sehen ist, ab: Steam Audio belegt den ersten Platz, gefolgt von Project Acoustics. Auf den dritten Rang schafft es Graph Audio, während FMOD für die meisten Probanden am schlechtesten Abschneidet. Dies ist nicht verwunderlich, denn FMOD ist das einzige Framework, welches keine Verfahren implementiert, um erweiterte Schalleffekte zu simulieren. Anders sieht es hingegen bei der Frage zur Präzision bei der Lokalisierung von Schallquellen aus, in Abbildung 23 zu sehen. Hier liegt Project Acoustics auf dem letzten Platz, obwohl es in der Theorie die physikalisch korrekteste Berechnung der Schallausbreitung aufweist. Nach Nachfrage begründeten die Probanden diese Entscheidung damit, dass es in den vielen vorhandenen Räumen der Testszene einen starken Nachhall gibt. Dies fördere zwar den Realismus, sei zur präzisen Lokalisierbarkeit aber nicht gut. Steam Audio auf der anderen Seite belegt den ersten Platz. Bei diesem Framework wird mehr mit Approximation der Schalleffekte gearbeitet und nicht mit totaler physikalischer Korrektheit. So wurde weiterhin auch auf die Nutzbarkeit in Spielen geachtet, bei welcher eine hundertprozentige physikalische Genauigkeit in Simulationen nicht immer zum Gameplay beiträgt.

Wo empfandest du den Sound am **realistischsten**?

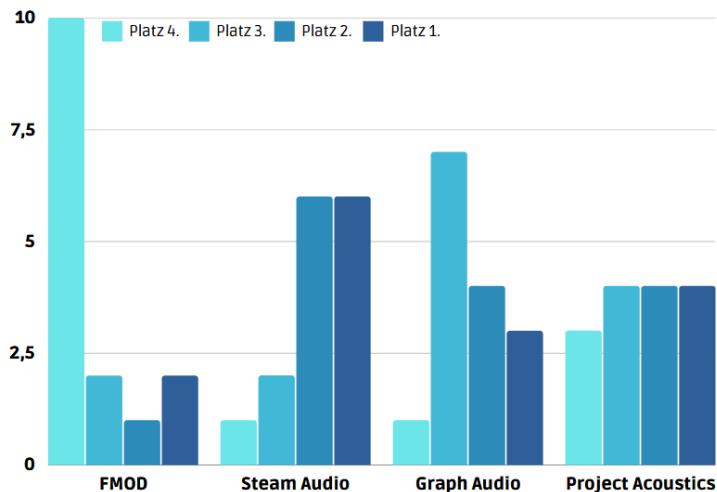


Abbildung 22: Ergebnis der Umfrage zum Realismus beim Vergleich der vier Frameworks aller Probanden zusammen.

Womit konntest du Schallquellen am **präzisesten** lokalisieren?

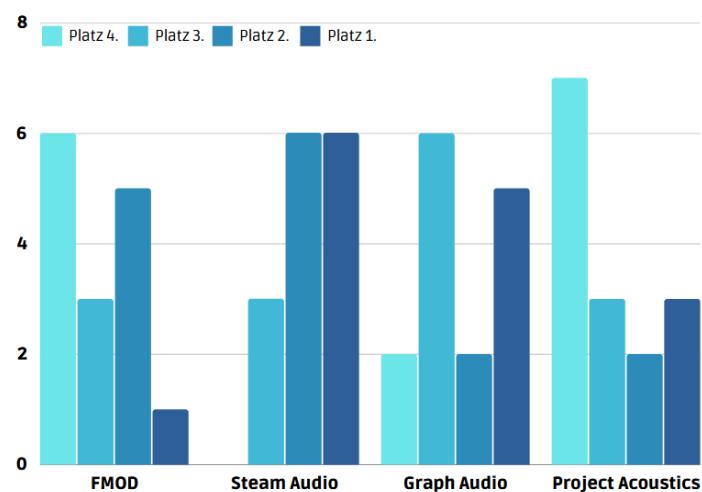


Abbildung 23: Ergebnis der Umfrage zur Präzision beim Vergleich der vier Frameworks aller Probanden zusammen.

	Mittelwerte nach Variation und Framework			
	Variation 0	Variation 1	Variation 2	Variation 3
	Distanz	Distanz	Distanz	Distanz
FMOD	13.25	13.35	10.98	12.68
Steam Audio	13.55	12.83	11.23	12.13
Graph Audio	12.08	12.24	11.68	11.8
Project Acoustic	11.22	11.33	12.89	11.55

Abbildung 24: Ergebnisse der Mittelwerte der Distanzen der einzelnen Frameworks, aufgeteilt auf die Variationen der Positionen. Geringer Distanzen sind besser. Einzelne Distanzwerte von über 20 wurden als Outlier entfernt.

Daten des Testprogramms. Bei den automatisch erhobenen Daten konnte bereits während der Durchführungen der Tests festgestellt werden, dass die aufgenommenen Zeiten nicht verwertet werden können. Dies hat den Grund, dass die Testpersonen zu unterschiedliche Ansätze hatten. Manche Testperson haben versucht besonders schnell fertig zu werden, während andere Testpersonen sich sehr viel Zeit gelassen haben. Sie haben oft den Marker recht schnell platziert, sind aber teilweise noch Minuten um den Marker gelaufen um sicher zu stellen, dass sie die richtige Position haben.

Die Distanzen hingegen können verglichen werden. Dazu wurden die Mittelwerte unter allen Personen der einzelnen Frameworks gebildet. Dies immer innerhalb einer Variation an Positionen, da diese wie zuvor erwähnt mit geringer Varianz gegeneinander verglichen werden können. In Abbildung 24 sind die so erhobenen Mittelwerte zu sehen. Insgesamt liegen die Werte relativ nahe beieinander. Dennoch kann festgestellt werden, dass Project Acoustics bei drei Varianten am besten abschneidet. Dies stellt einen Gegensatz zu der Aussage der Testpersonen her, dass es mit diesem Framework am schwierigsten sei, präzise zu lokalisieren. Auch kann festgestellt werden, dass FMOD außer in Variation 2 immer zu den am schlechtesten abschneidendenden Frameworks gehört. Diese Feststellung wird durch den Fragebogen bestätigt und es kann gesagt werden, dass die Erweiterung des Audio-Rendering durch Simulation von Schalleffekten auch für die Lokalisierung besser ist, wenn auch nicht mit hohen Unterschieden zur simplen Variante mit FMOD.

Probleme und Lösungsvorschläge. Im Bezug auf Graph Audio ist einigen Probanden aufgefallen, dass das durchs Rendering vermittelte Gefühl für die Richtung und Distanz zur Schallquelle nicht konstant ist. Probanden konnten feststellen, dass an bestimmten Stellen Schallquellen stark verdeckt wirkten und dann einen Schritt weiter zu Seite auf einmal viel lauter und direkter erschienen. Dieses Verhalten hängt mit der Auflösung des genutzten Graphen und der berechneten Verdeckungs- Werte der einzelnen Kanten zusammen. Das Verfahren zur automatischen Erstellung der Knoten verteilt diese nicht optimal in der Szene. So gibt es an einigen Stellen nicht genug Knoten, um effiziente Wege für die Schallübertragung berechnen zu können.

Bewegt sich der Zuhörer nun, stehen unter Umständen wieder ganz andere Wege zur Verfügung, die besser geeignet sind. Hier fließen auch die Verdeckungs-Werte der Kanten mit ein, da sie die Pfadlänge und somit auch den kürzesten gewählten Pfad beeinflussen. Ist der Verdeckungs-Wert aus irgendeinem Grund falsch berechnet und viel zu hoch, wird der in der Theorie am besten geeignete Pfad zur Schallübertragung nicht mehr genutzt. Abhilfe schafft hier das manuelle Überprüfen und verbessern des Graphen und dessen Verdeckungs-Werte.

8 Fazit

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und ausgewertet. Dazu wird zunächst das selbst implementierte Audio-Framework Graph Audio erläutert. Anschließend werden die Ergebnisse der zusätzlich im Rahmen dieser Arbeit getätigten Evaluation zum Vergleich der vier Audio-Frameworks FMOD, Steam Audio, Project Acoustics und Graph Audio besprochen. An dieser Stelle wird dann auch die zu Beginn aufgestellte Leitfrage, ob eine realistischere Schallausbreitung auch in einer erhöhten Lokalisierbarkeit von Schallquellen resultiert, beantwortet. Zum Schluss wird ein kurzer Zukunftsausblick zu Graph Audio, sowie zu realistischem Audio-Rendering im allgemeinen getätigt.

8.1 Graph Audio

Mit der Implementation von Graph Audio, welches auf der Grundlage von Cowans „Graph-Based Real-Time Spatial Sound Framework“ [1], basiert, konnte ein Audio-Framework geschaffen werden, welches simples binaurales Audio-Rendering erweitert. Dazu wurden neben den Schalleffekten Nachhall, Verdeckung und Transmission auch stärkere richtungs-basierte Hinweise hinzugefügt. Denn für das binaurale Rendering wird nicht die direkte Position der Schallquelle verwendet, sondern eine virtuelle Schallquelle, dessen Position auf dem kürzesten Pfade basiert, den der Schall in Anbetracht der Szenen-Geometrie nehmen kann. Im Gegensatz zu Cowans Vorlage wurde Graph Audio an einigen Stellen verändert und erweitert. Dazu zählt die Nutzung von Steam Audio zur Berechnung der Verdeckungs-Werte und des Nachhalls. Außerdem wurde, um ein „Springen“ der Audio-Hinweise beim Verschieben der virtuellen Schallquelle zu vermeiden, eine lineare Interpolation der Position hinzugefügt. Ein weiterer Fehler welcher in Cowans Framework gefunden und verbessert wurde ist folgender: Ist die Schallquelle im direkten Sichtfeld des Zuhörers, ist der vom Graph berechnete Pfad aufgrund der Auflösung des Graphen immer länger als der direkte Weg zur Schallquelle ohne Graph. Deshalb wird der Graph nicht genutzt, wenn es eine direkte Verbindung zwischen Zuhörer und Schallquelle gibt.

8.2 Evaluation und Leitfrage

Die im Rahmen dieser Arbeit getätigte Evaluation wurde in Form eines selbst entwickelten Testprogramms in Kombination mit einem Fragebogen durchgeführt. Im Testprogramm steuert die Testperson einen Charakter in First-Person Sicht. Die Situation die simuliert wird ist die, dass der Proband eine Schallquelle hört, diese aber nicht sehen kann. Nun muss durch die Szene gelaufen werden und die vermutete Position der Schallquelle mittels eines Markers erraten werden. Verglichen werden die vier Frameworks FMOD, Steam Audio, Project Acoustics und das selbst implementierte Graph Audio. Die Evaluation liefert insbesondere folgende Ergebnisse. Die Probanden haben größtenteils die Frameworks Graph Audio, Steam Audio und Project Acoustics als die besten Frameworks gewertet, sowohl was Realismus angeht, als auch was die Lokalisierbarkeit von Schallquellen angeht. Die durch das Testprogramm automatisch erhobenen Daten konnten diese Aussagen unterstützen, wenn auch der Unterschied in den gemessenen Werten nicht übermäßig hoch ausfällt. Die zusätzliche Simulation von Schalleffekten bringt also tatsächlich einen Vorteil gegenüber der simplen Variante FMOD, sowohl zum empfundenen Realismus, als auch zur Lokalisierbarkeit von Schallquellen. Damit kann die zu Beginn getätigte Leitfrage „Resultiert eine realistischere Schallausbreitung eines Audio-Framework in einer erhöhten Lokalisierbarkeit von Schallquellen?“ bejaht werden.

8.3 Zukunftsansicht

Bei der Entwicklung von Graph Audio gibt es noch Punkte, die in Zukunft verbessert werden können. Besonderes Augenmerk sollte hier auf die automatische Erstellung der Graphen in einer Szene gelegt werden. Denn nur mit einem gut abgestimmten Graphen kann das Framework realistisches Audio-Rendering bieten. Muss der Graph für jede Szene manuell eingestellt werden, verfehlt das Framework den Sinn der Erleichterung der Arbeit. Denn wenn ein Designer manuell etwas einstellen muss, könnte er auch direkt Verdeckungs- und Nachhall-Werte für die Szene bei einer Nutzung von FMOD einstellen.

Im Hinblick auf realistischen Sound innerhalb von kompetitiven Videospielen zur Verbesserung der Lokalisierung von Schallquellen kann gehofft werden, dass immer mehr Spieleentwickler dies umsetzen. Die hier getesteten kommerziellen Frameworks liefern sehr gute Ergebnisse bei gleichzeitig niedrigen Leistungskosten. Da diese Frameworks noch nicht sehr alt sind, werden sie in Zukunft möglicherweise deutlich an Beliebtheit gewinnen.

Anhang

Hier folgen weiterführende Dokumente und Informationen, welche die in dieser Arbeit getätigten Aussagen unterstützen.

Benchmark Graph Audio

Die folgende Tabelle liefert zwei separat mit der Software FRAPS durchgeführte *Benchmark*-Tests von Graph Audio in der *Project Acoustics Demo*-Szene. Die Testlänge beträgt jeweils eine Minute. Das System auf dem die Tests durchgeführt wurden läuft mit einer *Intel 8700K* CPU, welche auf 5 Ghz getaktet ist, einer *Nvidia Geforce 1080TI* Grafikkarte sowie 32 GB DDR4 Arbeitsspeicher.

FPS Durchschnitt	FPS Minimal	FPS Maximal	Frames	Zeit
558	137	2024	33451	60s
768	124	1866	46080	60s

Tabelle 1: FRAPS Benchmark-Test von Graph Audio auf der *Project Acoustics Demo*-Szene.

Evaluation Fragebogen

Es folgt der gesamte Fragebogen, den die Probanden der Evaluation ausfüllen mussten. Die Antwortmöglichkeiten kommen jeweils nach Einrückungen mit alphabetischer Nummerierung. Ist ein Doppelpunkt hinter einer Antwortmöglichkeit handelt es sich um eine Matrix mit den durch Bindestrichen getrennten Antwortmöglichkeiten für diese Zeile, bezogen auf die Antwort vor dem Doppelpunkt.

1. Allgemeine Fragen
 - 1.1. Wie lautet deine Teilnehmernummer?
 - 1.2. Wie alt bist du?
 - 1.2.a. 30 Jahre oder jünger
 - 1.2.b. Über 30 Jahre
 - 1.3. Bist du mit der Charaktersteuerung vertraut gewesen?
 - 1.3.a. Ja
 - 1.3.b. Nein
2. Erfahrung mit Videospielen
 - 2.1. Wie häufig spielst du Videospiele?

- 2.1.a. *PC* : Nie - Selten - Ab und zu - Oft
- 2.1.b. *Konsole* : Nie - Selten - Ab und zu - Oft
- 2.1.c. *Handy* : Nie - Selten - Ab und zu - Oft
- 2.2. Welche Genres spielst du am häufigsten?
 - 2.2.a. *First-Person-Shooter* : Nie - Selten - Ab und zu - Oft
 - 2.2.b. *Third-Person-Shooter* : Nie - Selten - Ab und zu - Oft
 - 2.2.c. *Role-Play-Games* : Nie - Selten - Ab und zu - Oft
 - 2.2.d. *Andere* : Nie - Selten - Ab und zu - Oft
- 3. Framework A - FMOD Audio
 - 3.1. Wie **schnell** konntest du die Schallquellen lokalisieren?
 - 3.1.a. *Musik* : Langsam - Eher Langsam - Eher Schnell - Schnell
 - 3.1.b. *Schritte* : Langsam - Eher Langsam - Eher Schnell - Schnell
 - 3.2. Wie **präzise/genau** konntest du die Schallquellen lokalisieren?
 - 3.2.a. *Musik* : Schlecht - Eher Schlecht - Eher Gut - Gut
 - 3.2.b. *Schritte* : Schlecht - Eher Schlecht - Eher Gut - Gut
 - 3.3. Wie **realistisch** empfandest du den Sound?
 - 3.3.a. Skala unrealistisch 1 bis realistisch 5
 - 3.4. Empfandest du etwas als hilfreich oder störend?
- 4. Framework B - Steam Audio
 - 4.1. Wie **schnell** konntest du die Schallquellen lokalisieren?
 - 4.1.a. *Musik* : Langsam - Eher Langsam - Eher Schnell - Schnell
 - 4.1.b. *Schritte* : Langsam - Eher Langsam - Eher Schnell - Schnell
 - 4.2. Wie **präzise/genau** konntest du die Schallquellen lokalisieren?
 - 4.2.a. *Musik* : Schlecht - Eher Schlecht - Eher Gut - Gut
 - 4.2.b. *Schritte* : Schlecht - Eher Schlecht - Eher Gut - Gut
 - 4.3. Wie **realistisch** empfandest du den Sound?
 - 4.3.a. Skala unrealistisch 1 bis realistisch 5
 - 4.4. Empfandest du etwas als hilfreich oder störend?
- 5. Framework C - Graph Audio
 - 5.1. Wie **schnell** konntest du die Schallquellen lokalisieren?
 - 5.1.a. *Musik* : Langsam - Eher Langsam - Eher Schnell - Schnell
 - 5.1.b. *Schritte* : Langsam - Eher Langsam - Eher Schnell - Schnell
 - 5.2. Wie **präzise/genau** konntest du die Schallquellen lokalisieren?
 - 5.2.a. *Musik* : Schlecht - Eher Schlecht - Eher Gut - Gut

5.2.b. *Schritte* : Schlecht - Eher Schlecht - Eher Gut - Gut

5.3. Wie **realistisch** empfandest du den Sound?

5.3.a. Skala unrealistisch 1 bis realistisch 5

5.4. Empfandest du etwas als hilfreich oder störend?

6. Framework D - Project Acoustics

6.1. Wie **schnell** konntest du die Schallquellen lokalisieren?

6.1.a. *Musik* : Langsam - Eher Langsam - Eher Schnell - Schnell

6.1.b. *Schritte* : Langsam - Eher Langsam - Eher Schnell - Schnell

6.2. Wie **präzise/genau** konntest du die Schallquellen lokalisieren?

6.2.a. *Musik* : Schlecht - Eher Schlecht - Eher Gut - Gut

6.2.b. *Schritte* : Schlecht - Eher Schlecht - Eher Gut - Gut

6.3. Wie **realistisch** empfandest du den Sound?

6.3.a. Skala unrealistisch 1 bis realistisch 5

6.4. Empfandest du etwas als hilfreich oder störend?

7. Vergleich

7.1. Womit konntest du Schallquellen am **schnellsten** lokalisieren?

7.1.a. *Framework A* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.1.b. *Framework B* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.1.c. *Framework C* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.1.d. *Framework D* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.2. Womit konntest du Schallquellen am **präzisesten/genausten** lokalisieren?

7.2.a. *Framework A* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.2.b. *Framework B* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.2.c. *Framework C* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.2.d. *Framework D* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.3. Bei welchem Framework empfandest du den Sound am **realistischsten**?

7.3.a. *Framework A* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.3.b. *Framework B* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.3.c. *Framework C* : Platz 4 - Platz 3 - Platz 2 - Platz 1

7.3.d. *Framework D* : Platz 4 - Platz 3 - Platz 2 - Platz 1

8. Einwilligungserklärung

- 8.1. Im folgenden willige ich ein, dass die Daten, welche in diesem Test zur Masterarbeit „Evaluation von Akustik-Frameworks in Bezug auf Lokalisierung von Schallquellen“ von Lucas Hilbig an der Uni Koblenz-Landau erhoben wurden, anonymisiert gespeichert und im Rahmen dieser Masterarbeit verwendet werden dürfen.
- 8.1.a. Ja, ich willige ein.

Ergebnisse Fragebogen

Es folgen die Antworten aller 15 Probanden, die zum Ergebnis dazu zählen.

Allgemeine Fragen			Erfahrung mit Videospielen							
Teilnehmernr.	Alter	Bist du mit der Technik vertraut?	Wie häufig spielst du Videospiele?			Welche Genres spielst du am häufigsten?				
			[PC]	[Konsole]	[Handy]	[FPS]	[TPS]	[RPG]	[Andere]	
1	<= 30 Jahre	Ja	Oft	Oft	Selten	Oft	Oft	Selten	Oft	
2	<= 30 Jahre	Ja	Oft	Nie	Nie	Oft	Nie	Oft	Oft	
3	<= 30 Jahre	Ja	Oft	Nie	Selten	Oft	Selten	Oft	Oft	
4	<= 30 Jahre	Ja	Oft	Nie	Nie	Ab und zu	Nie	Oft	Selten	
5	<= 30 Jahre	Ja	Oft	Nie	Nie	Nie	Nie	Oft	Selten	
6	> 30 Jahre	Ja	Oft	Selten	Ab und zu	Ab und zu	Selten	Oft	Ab und zu	
7	> 30 Jahre	Ja	Oft	Nie	Selten	Oft	Selten	Oft	Oft	
8	<= 30 Jahre	Ja	Oft	Nie	Selten	Oft	Ab und zu	Ab und zu	Selten	
9	<= 30 Jahre	Ja	Oft	Nie	Selten	Ab und zu	Ab und zu	Oft	Oft	
10	<= 30 Jahre	Ja	Oft	Nie	Ab und zu	Ab und zu	Nie	Selten	Oft	
11	<= 30 Jahre	Ja	Selten	Ab und zu	Selten	Nie	Nie	Oft	Oft	
12	<= 30 Jahre	Ja	Selten	Selten	Selten	Nie	Ab und zu	Ab und zu	Selten	
13	> 30 Jahre	Nein	Nie	Nie	Selten	Nie	Nie	Nie	Selten	
15	<= 30 Jahre	Ja	Oft	Nie	Nie	Oft	Selten	Nie	Nie	
16	<= 30 Jahre	Ja	Oft	Selten	Nie	Oft	Selten	Ab und zu	Oft	
<= 30 : 80% (12) Ja : 93,33% (14)			Oft	80% (12)	6,67% (1)	0% (0)	46,67% (7)	6,67% (1)	53,33% (8)	53,33% (8)
> 30 : 20% (3) Nein: 6,67% (1)			Ab und zu	0% (0)	6,67% (1)	13,33% (2)	26,67% (4)	20% (3)	20% (3)	6,67% (1)
			Selten	13,33% (2)	20% (3)	53,33% (8)	0% (0)	33,33% (5)	13,33% (2)	33,33% (5)
			Nie	6,67% (1)	66,67% (10)	33,33% (5)	26,67% (4)	40% (6)	13,33% (2)	6,67% (1)

Framework A - FMOD							
Teilnehmernr.		Wie schnell konntest du Schallquellen lokalisieren?		Wie präzise/genau konntest du Schallquellen lokalisieren?		Wie realistisch empfandest du den Sound?	Empfandest du etwas hilfreich oder störend?
		[Musik]	[Schritte]	[Musik]	[Schritte]		
1		Eher Schnell	Eher Langsam	Eher Gut	Eher Schlecht	3	Ich hatte das Gefühl, durch offene Wände/Türen nicht so viel "besser" zu hören, als durch massive Wände, wie man es eigentlich erwartet.
2		Eher Schnell	Schnell	Gut	Eher Schlecht	4	Schritte waren zu laut
3		Schnell	Eher Langsam	Gut	Eher Gut	3	
4		Schnell	Eher Schnell	Gut	Eher Gut	3	Beim Gras war es sehr schwierig.
5		Schnell	Eher Schnell	Gut	Eher Gut	4	Besser als die erste Runde
6		Eher Schnell	Eher Langsam	Gut	Eher Gut	3	When you turn away from the sound, it feels like your ear gets sealed off, makes locating very easy but not a very comfortable feeling.
7		Schnell	Eher Schnell	Gut	Eher Schlecht	2	
8		Schnell	Schnell	Gut	Eher Gut	5	
9		Eher Schnell	Eher Langsam	Gut	Eher Gut	3	
10		Eher Langsam	Langsam	Eher Schlecht	Schlecht	1	Fußstampfen waren extrem schlecht zu finden, da sie über einen weiten Raum (bis zu 3 Häusern) zu hören waren
11		Schnell	Eher Langsam	Gut	Eher Gut	3	In vielen Räumen gleicher/ähnlicher Klang.
12		Schnell	Eher Langsam	Eher Gut	Eher Schlecht	3	Höhenunterschiede schwierig
13		Schnell	Eher Schnell	Gut	Gut	4	Sprache beim Lied irritiert
15		Schnell	Eher Schnell	Gut	Eher Gut	4	
16		Schnell	Eher Schnell	Gut	Schlecht	3	
							Realistisch
Schnell/Gut	66,67% (10)	13,33% (2)	80% (12)	6,67% (1)	[5]: 6,67% (1)		
Eher Schnell/Gut	26,67% (4)	40% (6)	13,33% (2)	53,33% (8)	[4]: 26,67% (4)		
Eher Langsam/Schle	6,67% (1)	40% (6)	6,67% (1)	26,67% (4)	[3]: 53,33% (8)		
Langsam/Schlecht	0% (0)	6,67% (1)	0% (0)	13,33% (2)	[2]: 6,67% (1)		
					[1]: 6,67% (1)		
					Unrealistisch		

Fragebogen cleaned

	Framework B - Steam Audio						
Teilnehmernr.		Wie schnell konntest du Schallquellen lokalisieren?		Wie präzise/genau konntest du Schallquellen lokalisieren?		Wie realistisch empfandest du den Sound?	Empfandest du etwas hilfreich oder störend?
		[Musik]	[Schritte]	[Musik]	[Schritte]		
1		Schnell	Schnell	Gut	Gut	4	Die Wände haben ein wenig zu viel Sound "geschluckt".
2		Eher Schnell	Eher Schnell	Gut	Gut	5	
3		Schnell	Eher Schnell	Gut	Eher Gut	4	
4		Schnell	Schnell	Gut	Eher Gut	2	
5		Eher Schnell	Eher Schnell	Eher Gut	Gut	4	Drapery hinten drin war hart
6		Eher Schnell	Eher Langsam	Eher Gut	Eher Gut	4	Dont do this with bad motion sickness.
7		Schnell	Schnell	Gut	Gut	4	
8		Schnell	Schnell	Gut	Gut	4	
9		Schnell	Schnell	Gut	Gut	5	
10		Eher Langsam	Langsam	Eher Gut	Gut	3	"Größerer, lauterer" Stampfer bei den Schritten war hilfreich
11		Schnell	Eher Schnell	Gut	Gut	4	
12		Eher Langsam	Langsam	Eher Gut	Schlecht	5	
13		Schnell	Schnell	Gut	Gut	5	Sprache beim song stört
15		Schnell	Schnell	Gut	Gut	4	Bhop war gut
16		Schnell	Eher Langsam	Gut	Eher Schlecht	4	
							Realistisch
Schnell/Gut		66,67% (10)	46,67% (7)	73,33% (11)	66,67% (10)	[5]: 26,67% (4)	
Eher Schnell/Gut		20% (3)	26,67% (4)	26,67% (4)	20% (3)	[4]: 60% (9)	
Eher Langsam/Schlecht		13,33% (2)	13,33% (2)	0% (0)	6,67% (1)	[3]: 6,67% (1)	
Langsam/Schlecht		0% (0)	13,33% (2)	0% (0)	6,67% (1)	[2]: 6,67% (1)	
						[1]: 0% (0)	Unrealistisch

Fragebogen cleaned

Fragebogen cleaned

Framework D - Project Acoustics							
Teilnehmernr.		Wie schnell konntest du Schallquellen lokalisieren?		Wie präzise/genau konntest du Schallquellen lokalisieren?		Wie realistisch empfandest du den Sound?	Empfandest du etwas hilfreich oder störend?
		[Musik]	[Schritte]	[Musik]	[Schritte]		
1		Eher Schnell	Eher Schnell	Eher Schlecht	Schlecht	4	Es gab zu viel Hall. Die Diffraktion war gut.
2		Schnell	Eher Schnell	Gut	Gut	5	
3		Schnell	Eher Schnell	Gut	Eher Gut	4	
4		Schnell	Schnell	Gut	Gut	4	
5		Eher Langsam	Eher Schnell	Eher Gut	Eher Gut	4	Wenn man zu beginn schon was gehört hat war es deutlich einfacher, außerdem konnte man den ball nicht in die luft plazieren
6		Eher Langsam	Langsam	Eher Schlecht	Schlecht	4	Sounds were all over the place
7		Schnell	Schnell	Gut	Eher Gut	3	
8		Schnell	Eher Schnell	Eher Gut	Eher Gut	3	
9		Eher Schnell	Eher Langsam	Eher Gut	Eher Gut	5	
10		Eher Langsam	Schnell	Eher Schlecht	Gut	3	Musik war sehr realistisch (5), aber dadurch auch schwieriger zu finden, Fußstampfen waren sehr unrealistisch (1), aber dadurch extrem leicht zu finden
11		Schnell	Eher Langsam	Gut	Eher Gut	4	Wenn der Raum groß war, war es schwerer die Geräuschquelle zu lokalisieren.
12		Eher Schnell	Langsam	Gut	Schlecht	4	
13		Schnell	Schnell	Gut	Gut	5	Sprache stört
15		Eher Schnell	Eher Langsam	Gut	Eher Schlecht	4	
16		Schnell	Eher Schnell	Eher Gut	Eher Schlecht	4	
							Realistisch
Schnell/Gut		53,33% (8)	26,67% (4)	53,33% (8)	26,67% (4)	[5]: 20% (3)	
Eher Schnell/Gut		26,67% (4)	40% (6)	26,67% (4)	40% (6)	[4]: 60% (9)	
Eher Langsam/Schlecht		20% (3)	20% (3)	20% (3)	13,33% (2)	[3]: 20% (3)	
Langsam/Schlecht		0% (0)	13,33% (2)	0% (0)	20% (3)	[2]: 0% (0)	
							Unrealistisch

Fragebogen cleaned

Ergebnisse Testprogramm

Es folgen die automatisch durch das Testprogramm erhobenen Daten zur Distanz und benötigten Zeit, die die Testpersonen benötigt haben, um die Schallquellen zu finden.

	Mittelwerte nach Variation und Framework				
	Variation 0		Variation 1	Variation 2	Variation 3
	Distanz	Distanz	Distanz	Distanz	
FMOD	13.25		13.35	10.98	12.68
Steam Audio	13.55		12.83	11.23	12.13
Graph Audio	12.08		12.24	11.68	11.8
Project Acoustic	11.22		11.33	12.89	11.55

Abbildung 25: Ergebnisse der Mittelwerte der Distanzen der einzelnen Frameworks, aufgeteilt auf die Variationen der Positionen. Geringer Distanzen sind besser. Einzelne Distanzwerte von über 20 wurden als Outlier entfernt.

Test.Nr.	Mittelwerte (ohne Outlier mit Distanz > 20)							
	Variation 0		Variation 1		Variation 2		Variation 3	
	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit
1	15	00:21	11.1	00:34.1	12.1	00:42.8	11.4	00:44.1
2	14	00:21.7	15	00:20.7	13.3	00:25.2	14.2	00:24.4
3	9.2	00:17.5	14.25	00:20.1	14.75	00:42.8	13.7	00:18.1
4	9	00:15.1	14	00:14.4	11	00:21.1	14.3	00:16.3
5	12.1	00:32.5	15.1	00:36.9	9.2	00:51.6	13.3	00:52.2
6	16	00:53.2	11.5	00:37.8	16	00:58.3	13	00:34.7
7	15.2	01:01.3	12.4	00:36.7	9	00:35.1	12.8	00:33.1
8	9	01:28.2	8.2	01:03.3	7.8	00:54.6	7.6	00:58.7
9	15.25	00:50.2	11.25	00:37.1	12	00:43.5	9.1	00:49.8
10	11.8	00:59.1	10.5	01:11.2	10	01:17.1	14	01:28.7
11	10.9	01:13.8	8.4	02:02.9	12.7	01:01.1	12.2	01:41.9
12	6.3	1:45.2	13.7	01:36.2	12.5	02:14.9	13	01:13.8
13	15	04:39.7	17	04:13.2	13.5	02:34.4	13	01:53.9
15	17.2	00:15.1	12.7	00:16.5	10.8	00:47.3	10	00:12.6
16	13.25	00:17.4	10.5	00:22.6	11.2	00:31.6	9.2	00:40.2

Abbildung 26: Ergebnisse der Mittelwerte der Distanzen, aufgeteilt auf die Variationen der Positionen. Geringer Distanzen sind besser. Einzelne Distanzwerte von über 20 wurden als Outlier entfernt.

Schallquellenposition	Teilnehmernummer	Zeit																16	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		
		Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	Zeit	Distanz	
		2.6 [FMC 00:34.4 17.8]	[Fm 0:18.1 0.7]	[Stee 00:21.5 2.3]	[Gra 00:28.8 6.2]	[Proj 01:09.5 1.8]	[Ste 00:30.5 2.8]	[Gra 01:19.5 0.8]	[FMC 01:05.4 3.3]	[Proj 00:27.9 2.4]	[Ste 00:52.3 3.0]	[Gra 00:36.5 4.1]	[FMC 04:54.7 1.3]	[Stee 00:13.2 2.7]	[Gra 00:25.7				
1	2.2	00:29.4 4.4	0:19.7 0.7	00:22.5 3.6	00:15.3 4.0	00:27.9 3.5	00:25.3 2.8	00:30.9 1.1	00:48.0 2.0	00:28.9 1.0	00:28.4 1.6	01:01.7 13.7	01:14.5 4.4	01:40.5 3.6	00:18.0 0.8	00:16.8			
2	3.1	00:20.4 4.3	0:05.1 1.4	00:12.2 3.5	00:13.3 2.1	00:15.6 4.1	00:20.8 1.6	00:15.1 3.9	00:59.9 53.6	00:28.3 1.4	00:41.4 0.3	00:20.9 2.5	02:41.7 3.0	01:27.7 1.9	00:09.4 1.0	00:14.8			
3	4.3	00:30.6 5.6	0:20.1 0.5	00:19.7 89.6	00:53.4 5.3	00:29.9 3.6	00:26.9 1.2	01:06.6 1.9	01:10.8 3.7	00:51.4 2.9	00:58.4 0.8	01:11.7 4.0	01:19.1 1.8	04:24.7 2.3	00:17.9 5.4	01:10.8			
Variation 0	5	1.9	00:21.5 3.7	0:14.2 1.3	00:11.7 0.9	00:15.1 3.6	00:46.8 2.9	00:28.8 1.7	00:34.7 0.3	01:12.4 2.0	00:19.9 1.8	00:46.2 1.1	00:28.7 2.2	00:31.5 3.1	01:36.4 2.0	00:21.0 1.6	00:15.5		
	6	4.1	01:05.5 43.3	0:09.4 23.6	00:15.5 3.5	00:20.3 4.1	00:30.7 10.0	00:50.5 2.6	00:40.1 0.9	02:35.3 7.4	01:04.6 2.3	01:04.6 2.5	00:46.2 8.6	02:47.2 11.6	05:07.6 1.7	00:17.0 10.6	00:27.1		
7	8.6	00:39.8 9.1	0:10.9 7.9	00:09.1 66.8	00:43.5 3.4	00:18.3 4.7	00:37.6 1.3	02:17.7 0.3	01:35.3 2.1	00:44.7 0.8	00:59.7 5.3	00:42.9 3.4	00:47.7 5.7	00:26.0 2.1	00:17.7 70.8	02:27.9			
8	2.6	00:47.1 2.9	0:18.4 7.6	00:19.2 2.9	00:25.8 6.5	00:39.3 1.4	01:15.9 1.8	00:52.5 1.1	01:08.6 1.5	01:49.7 0.8	01:06.8 1.2	01:50.9 4.9	02:57.9 11.0	03:27.3 1.7	00:15.0 4.4	00:18.2			
9	4.3	01:13.8 2.5	0:17.4 2.7	00:19.4 3.0	00:19.4 3.4	00:40.1 4.3	00:17.0 2.1	00:47.0 0.8	01:21.0 0.6	00:42.4 1.3	01:51.8 1.3	01:38.4 13.4	03:03.0 7.5	03:32.5 4.1	00:13.8 1.9	00:22.5			
10	3.4	01:10.6 3.3	0:22.9 5.3	00:36.5 4.7	00:20.3 3.6	00:38.3 3.3	00:46.2 3.0	01:03.3 2.3	01:32.9 6.4	01:46.0 2.9	02:47.6 1.3	02:04.0 5.8	04:46.4 6.8	09:13.2 2.9	00:30.8 2.7	00:43.3			
11	1.4	[Stee 00:38.7 3.3]	[Stee 00:12.1 1.3]	[Gra 00:15.2 1.6]	[Proj 00:14.8 1.2]	[FMC 00:35.6 2.2]	[Gra 00:21.9 3.0]	[Proj 00:11.9 1.7]	[FMC 00:11.9 1.7]	[Stee 00:36.9 2.5]	[Gra 00:07.9 0.5]	[Proj 00:49.6 3.1]	[FMC 01:19.5 8.1]	[Stee 00:56.8 1.9]	[Gra 00:08.6 0.5]	[Proj 00:44.1			
12	2.6	00:33.8 1.5	0:20.7 2.5	01:00.7 3.0	00:20.4 1.0	00:36.8 1.1	00:48.4 1.8	01:25.6 0.5	00:55.6 0.9	00:50.8 1.2	04:49.6 1.0	02:04.4 1.3	01:15.0 1.6	04:51.5 0.3	00:24.0 2.9	00:26.0			
13	0.9	00:27.3 2.4	0:41.6 2.0	00:42.6 2.5	00:16.9 2.3	00:50.2 4.3	00:21.8 2.3	00:32.5 0.7	01:16.5 1.4	00:37.4 2.2	01:51.4 0.8	00:49.4 2.5	01:35.6 6.0	01:31.3 1.5	00:24.8 2.3	00:51.7			
14	0.4	00:24.8 5.2	0:10.1 1.2	00:12.7 2.3	00:15.4 1.9	00:23.1 7.2	00:13.9 0.6	00:18.9 0.8	00:26.0 1.1	00:33.4 1.1	00:32.1 2.9	01:34.7 8.7	02:14.6 0.9	00:09.7 1.5	00:18.8				
15	2.4	00:27.0 2.1	0:25.1 5.8	00:11.6 2.1	00:20.2 5.4	00:22.9 7.1	00:14.3 2.3	00:32.5 1.7	00:58.3 1.0	00:30.6 1.1	00:43.0 4.1	01:14.2 6.8	02:34.2 2.5	01:22.0 4.6	00:09.2 2.9	00:15.0			
16	2.0	00:38.4 3.4	0:12.3 32.4	00:47.0 2.2	00:13.6 28.2	00:34.5 7.0	00:25.0 1.8	00:20.4 0.8	01:10.8 1.8	00:42.4 1.3	01:31.8 53.3	03:29.1 0.8	02:16.9 4.4	01:19.4 3.3	00:22.0 1.3	00:15.3			
17	1.2	00:34.2 4.2	0:19.0 2.7	00:25.4 2.7	00:32.5 1.6	00:51.3 3.7	00:33.7 0.3	00:53.4 1.1	01:15.7 0.4	00:51.2 1.1	01:21.4 2.8	01:21.4 3.6	01:18.9 5.5	02:30.1 4.1	00:21.2 2.4	00:24.8			
18	2.5	00:20.6 2.5	0:22.6 4.2	00:21.9 7.8	00:12.0 1.8	00:35.6 1.2	00:27.3 1.7	00:51.3 0.2	00:30.4 2.8	00:52.2 0.9	01:42.8 1.2	03:39.2 2.0	01:16.8 4.0	01:51.4 1.2	00:21.8 56.0	00:59.7			
19	0.9	00:45.3 6.2	0:15.2 4.5	00:31.9 1.2	00:14.1 1.6	00:39.3 3.4	00:28.3 8.3	00:55.5 0.6	01:37.0 8.3	01:02.1 0.7	02:11.7 3.2	01:18.9 2.2	05:07.7 5.8	03:26.3 1.8	00:10.5 5.8	00:33.2			
20	1.0	00:29.7 2.9	0:48.0 1.2	00:09.8 2.1	00:08.9 2.3	00:12.4 4.2	00:14.6 1.0	00:13.8 1.2	01:10.9 0.7	00:14.1 1.0	00:28.0 0.7	02:56.3 22.6	00:48.6 1.8	03:35.0 2.2	00:07.7 0.9	00:12.4			
21	2.9	[Gra 00:33.8 3.2]	[Gra 00:22.7 8.7]	[Proj 00:10.7 1.6]	[FMC 00:12.1 3.3]	[Stee 00:36.3 2.6]	[Proj 00:40.7 3.3]	[FMC 00:15.7 0.7]	[Stee 00:43.2 1.4]	[Gra 00:43.2 1.4]	[Proj 00:43.0 1.9]	[FMC 00:47.1 2.0]	[Stee 01:03.8 3.56]	[Gra 01:03.8 1.8]	[Proj 00:25.0 2.8]	[FMC 00:34.6			
22	0.8	00:44.8 1.3	0:10.3 3.0	00:13.8 3.6	00:11.8 1.1	00:57.2 8.0	00:54.1 0.5	00:42.1 0.1	01:19.1 1.9	00:24.1 2.4	01:52.9 0.5	01:10.1 5.1	03:47.7 2.6	01:55.3 2.2	00:10.9 2.6	00:13.4			
23	2.0	00:27.4 4.0	0:11.2 4.2	00:12.3 2.3	00:11.8 1.1	00:29.1 11.8	00:40.4 1.0	00:45.8 0.6	00:50.7 1.3	00:29.7 1.2	01:22.4 0.8	00:32.3 3.9	00:27.3 6.5	00:41.7 2.6	00:15.7 2.6	00:12.1			
24	1.5	00:28.8 4.0	0:11.7 2.7	00:11.1 2.1	00:09.8 1.5	00:26.4 4.2	00:12.0 2.2	00:23.0 0.7	00:37.2 0.7	00:19.6 9.4	02:55.7 1.7	00:34.4 2.8	01:10.9 3.7	00:22.6 3.4	00:13.0 10.1	00:13.6			
25	2.3	00:34.9 4.5	0:12.8 1.6	00:25.1 1.5	00:27.9 2.3	01:04.9 1.6	00:58.3 2.3	00:42.8 0.2	00:59.8 1.3	00:49.1 2.6	01:10.0 2.0	01:06.3 1.4	03:33.2 4.6	00:51.0 0.9	00:24.4 5.8	00:16.3			
Variation 2	26	1.0	01:35.0 21.7	0:21.3 2.0	01:09.5 2.2	00:21.0 0.5	02:44.3 100.2	00:38.0 1.5	00:32.2 0.4	01:12.5 0.2	00:35.1 0.6	02:31.2 25.3	00:34.1 3.2	03:51.3 2.5	02:23.7 0.9	02:07.5 56.4	00:20.9		
	27	0.9	00:28.1 1.5	0:22.4 1.2	00:18.3 0.2	00:23.2 3.0	00:46.3 12.9	01:04.2 0.6	00:20.5 1.9	00:44.2 0.5	01:37.4 1.5	00:33.0 3.7	02:22.7 3.7	02:51.6 8.9	00:55.6 3.6	00:17.7 36.9	00:11.1		
28	1.4	00:34.6 1.2	0:42.9 1.1	00:58.6 26.5	00:21.2 0.8	00:35.2 60.3	00:52.5 2.3	00:25.7 1.5	01:10.4 1.1	01:04.1 0.9	00:34.5 2.9	00:37.5 2.4	00:39.3 0.9	01:41.7 4.0	00:27.4 3.3	00:09.6			
29	2.5	00:16.6 3.1	0:32.1 5.0	00:25.2 1.0	00:17.5 35.9	00:40.2 6.3	00:34.3 0.4	00:54.1 2.0	00:25.5 0.3	00:32.7 0.6	01:56.3 3.4	00:29.2 8.1	02:12.1 0.7	00:44.7 19.9	00:12.7				
30	0.9	00:44.8 3.9	0:11.9 2.4	00:20.0 4.7	00:14.8 0.3	00:32.1 17.0	00:11.2 2.5	00:33.8 1.8	00:35.0 1.6	00:47.7 0.6	01:03.4 4.2	01:24.4 1.1	00:56.7 4.8	01:40.8 1.5	00:17.5				
31	3.6	[Proj 00:34.7 7.0]	[Proj 00:15.7 1.9]	[FMC 00:07.9 1.3]	[Stee 00:11.4 3.4]	[Gra 00:19.9 1.9]	[Stee 00:11.7 0.6]	[Proj 00:11.7 0.6]	[FMC 00:19.9 1.4]	[Gra 00:18.4 2.9]	[FMC 00:26.4 5.6]	[Proj 00:21.4 1.5]	[Gra 00:21.4 1.5]	[FMC 00:10.2 3.5]	[Stee 00:09.9 0.7]	[FMC 00:10.2 3.5]	[Stee 00:09.9 0.7]		
32	0.9	00:34.5 1.2	0:20.1 9.9	00:21.3 1.5	00:14.3 1.5	00:23.6 2.2	00:22.3 0.3	00:35.4 1.5	00:22.7 7.8	00:43.4 1.6	00:44.5 7.9	00:31.0 8.4	00:26.3 2.8	00:26.2 2.8	00:12.4				
33	2.7	00:29.9 2.4	0:29.7 1.9	00:22.7 2.7	00:08.9 0.7	00:25.0 2.3	00:57.2 1.0	00:25.1 1.4	00:34.2 0.8	00:18.9 5.0	00:45.4 3.1	01:49.3 7.7	00:43.0 12.6	00:25.3 5.8	00:13.4 0.9	00:16.5			
34	2.1	00:41.0 2.9	0:15.9 14.9	01:09.9 2.2	00:19.4 1.7	00:33.2 2.6	00:44.8 3.7	00:19.4 0.6	01:20.0 0.5	00:37.9 3.5	01:21.0 0.5	01:59.4 4.6	00:50.2 13.6	00:49.7 2.7	00:40.0 0.7	00:28.5			
35	2.7	00:51.2 4.1	0:26.1 1.3	00:23.8 6.1	00:21.3 1.5	00:42.7 1.5	00:26.3 1.7	00:30.9 0.9	00:37.0 0.6	00:35.5 1.8	00:38.5 1.3	02:26.6 7.3	00:42.1 10.7	00:24.1 2.3	00:09.0 2.0	00:24.5			
36	4.1	00:51.0 1.5	0:37.5 21.7	00:23.0 1.0	00:39.3 29.4	00:43.3 1.6	00:29.0 1.2	01:14.4 0.5	01:15.9 1.1	00:41.4 0.5	03:05.5 0.8	01:05.3 3.2	01:16.3 25.7	00:54.1 22.1	00:14.3 3.4	01:02.9			
37	1.5	00:41.9 1.9	0:09.5 12.6	00:12.3 2.7	00:09.4 1.2	00:42.4 4.0	00:18.5 1.0	00:26.4 4.4	01:31.3 0.9	01:19.4 2.7	01:53.2 1.5	02:05.2 2.9	00:44.7 2.8	01:08.8 2.1	00:11.6 3.7	00:15.1			
38	1.1	00:43.4 1.5	0:13.8 3.1	00:17.1 2.8	00:12.0 0.4	00:32.1 1.7	00:11.2 2.5	00:33.8 1.8	00:35.0 1.6	00:47.7 0.6	01:03.4 4.2	01:24.4 1.1	00:56.7 4.8	01:40.8 1.5	00:17.5				
39	0.7	00:42.8 1.0	0:41.1 2.1	00:12.6 6.3	00:34.5 2.0	00:55.1 1.3	00:34.7 1.4	00:34.2 0.6	02:14.0 0.6	00:31.3 0.8	00:24.5 1.7	02:28.8 1.6	01:05.3 8.0	00:10.6 0.5	00:12.5 0.6	02:19.9			
40	1.5	00:57.8 2.2	0:51.6 11.8	00:24.2 3.5	00:24.8 1.6	01:37.9 11.5	00:28.2 2.4	00:38.5 0.5	00:37.0 0.5	02:33.9 1.9	01:46.2 1.3	02:00.8 0.5	01:36.0 2.9	01:20.2 12.4	00:20.9 0.6	00:22.0			

Literatur

- [1] B. B. Cowan, „A graph-based real-time spatial sound framework,“ Diss., 2020.
- [2] F. Jacobsen, T. Poulsen, J. H. Rindel, A. C. Gade und M. Ohlrich, *Fundamentals of acoustics and noise control*, 2011.
- [3] Microsoft. (2019). „Immersive sound propagation for games and mixed reality,“ Adresse: <https://www.microsoft.com/en-us/research/project/project-triton/> (besucht am 03.11.2021).
- [4] K. Attenborough, „Sound Propagation in the Atmosphere,“ in Jan. 2014, S. 117–155. DOI: 10.1007/978-1-4939-0755-7_4.
- [5] N. Raghuvanshi, J. Snyder, R. Mehra, M. Lin und N. Govindaraju, „Pre-computed Wave Simulation for Real-Time Sound Propagation of Dynamic Sources in Complex Scenes,“ *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2010*, Jg. 29, Juli 2010. Adresse: <https://www.microsoft.com/en-us/research/publication/precomputed-wave-simulation-real-time-sound-propagation-dynamic-sources-complex-scenes/>.
- [6] F. Rumsey, *Spatial Audio*. Taylor & Francis, 2012, ISBN: 9781136119903. Adresse: <https://books.google.de/books?id=b3kqBgAAQBAJ>.
- [7] V. Pulkki, „Virtual sound source positioning using vector base amplitude panning,“ *Journal of the audio engineering society*, Jg. 45, Nr. 6, S. 456–466, 1997.
- [8] V. Pulkki und M. Karjalainen, „Localization of amplitude-panned virtual sources I: stereophonic panning,“ *Journal of the Audio Engineering Society*, Jg. 49, Nr. 9, S. 739–752, 2001.
- [9] S. Laine, S. Siltanen, T. Lokki und L. Savioja, „Accelerated beam tracing algorithm,“ *Applied Acoustics*, Jg. 70, Nr. 1, S. 172–181, 2009. DOI: <https://doi.org/10.1016/j.apacoust.2007.11.011>. Adresse: <http://www.sciencedirect.com/science/article/pii/S0003682X07001910>.
- [10] C. Schissler und D. Manocha, „Gsound: Interactive sound propagation for games,“ in *Audio Engineering Society Conference: 41st International Conference: Audio for Games*, Audio Engineering Society, 2011.
- [11] R. Mehra, L. Antani, S. Kim und D. Manocha, „Source and Listener Directivity for Interactive Wave-Based Sound Propagation,“ *IEEE Transactions on Visualization and Computer Graphics*, Jg. 20, Nr. 4, S. 495–503, 2014. DOI: 10.1109/TVCG.2014.38.

- [12] M. L. Ibáñez, N. Álvarez und F. Peinado, „A study on an efficient spatialisation technique for near-field sound in video games,“ in *Proceedings of the 4th Congreso de la Sociedad Espanola para las Ciencias del Videojuego,(Barcelona)*, 2017.
- [13] F. Technologies. (2021). „FMOD,“ Adresse: <https://www.fmod.com/> (besucht am 03. 11. 2021).
- [14] V. Corporation. (2017). „Jetzt neu: Steam Audio,“ Adresse: <https://steamcommunity.com/games/596420/announcements/detail/521693426582988261> (besucht am 03. 11. 2021).
- [15] Microsoft. (2019). „What is Project Acoustics?“ Adresse: <https://docs.microsoft.com/en-us/gaming/acoustics/what-is-acoustics> (besucht am 03. 11. 2021).
- [16] F. Technologies. (2021). „FMOD Games,“ Adresse: <https://www.fmod.com/games> (besucht am 05. 11. 2021).
- [17] F. Technologies. (2021). „FMOD documentation 3D sound and spatialization,“ Adresse: <https://www.fmod.com/resources/documentation-api?version=2.02&page=core-guide.html#3d-sound-and-spatialization> (besucht am 05. 11. 2021).
- [18] F. Technologies. (2021). „FMOD documentation Spatializer Effect,“ Adresse: <https://www.fmod.com/resources/documentation-studio?version=2.02&page=glossary.html#spatializer-effect> (besucht am 05. 11. 2021).
- [19] V. Corporation. (2020). „Steam Audio Github,“ Adresse: <https://valvesoftware.github.io/steam-audio/> (besucht am 03. 11. 2021).
- [20] B. Games. (2020). „Escape From Tarkov Patchnotes 0.12.6,“ Adresse: <https://forum.escapefromtarkov.com/topic/129049-patch-01267456/> (besucht am 08. 11. 2021).
- [21] V. Corporation. (2020). „Steam Audio Documentation C-API HRTF,“ Adresse: <https://valvesoftware.github.io/steam-audio/doc/capi/guide.html#hrtf> (besucht am 10. 11. 2021).
- [22] V. Corporation. (2020). „Steam Audio Documentation FMOD Spatializer,“ Adresse: <https://valvesoftware.github.io/steam-audio/doc/fmod/spatializer.html> (besucht am 10. 11. 2021).
- [23] N. Raghuvanshi und J. Snyder, „Parametric Wave Field Coding for Pre-computed Sound Propagation,“ *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2014*, Jg. 33, Juli 2014. Adresse: <https://www.microsoft.com/en-us/research/publication/parametric-wave-field-coding-precomputed-sound-propagation/>.

- [24] N. Raghuvanshi und J. Snyder, „Parametric Directional Coding for Precomputed Sound Propagation,“ *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2018*, Jg. 37, Nr. 4, Sep. 2018. Adresse: <https://www.microsoft.com/en-us/research/publication/spatial-audio-for-immersive-sound-propagation/>.
- [25] Microsoft, *Project Acoustics Game Developers Conference 2019*, 2019. Adresse: <https://www.youtube.com/watch?v=uY4G-GUAQIE>.
- [26] Microsoft. (2021). „Project Acoustics Unreal Beta Features,“ Adresse: <https://docs.microsoft.com/en-us/gaming/acoustics/unreal-beta> (besucht am 12.11.2021).
- [27] B. Cowan und B. Kapralos, „GPU-based real-time acoustical occlusion modeling,“ *Virtual reality*, Jg. 14, Nr. 3, S. 183–196, 2010.
- [28] B. Cowan und B. Kapralos, „A GPU-based method to approximate acoustical reflectivity,“ *Journal of Graphics, GPU, and Game Tools*, Jg. 15, Nr. 4, S. 210–215, 2011.
- [29] B. Cowan, B. Kapralos und K. Collins, „Does Improved Sound Rendering Increase Player Performance? A Graph-Based Spatial Sound Framework,“ *IEEE Transactions on Games*, S. 1–1, 2020. DOI: 10.1109/TG.2020.3000261.
- [30] Unity. (2021). „Unitys neuer Multithreading-Data-Oriented Technology Stack,“ Adresse: <https://unity.com/de/dots> (besucht am 17.11.2021).
- [31] Unity. (2021). „Unity C Sharp Job System,“ Adresse: <https://docs.unity3d.com/Manual/JobSystem.html> (besucht am 17.11.2021).
- [32] V. Corporation. (2020). „Steam Audio Documentation Unity Steam Audio Source,“ Adresse: <https://valvesoftware.github.io/steam-audio/doc/unity/source.html> (besucht am 17.11.2021).
- [33] S. Green, „Particle simulation using cuda,“ *NVIDIA whitepaper*, Jg. 6, S. 121–128, 2010.
- [34] Unity. (2021). „Unity Dokumentation FixedUpdate,“ Adresse: <https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html> (besucht am 17.11.2021).