Sound und Sound Design in First Person Shooter: Theorie und praktische Umsetzung mit Hilfe der CryENGINE 3

Bachelorarbeit

im Studiengang
Audiovisuelle Medien

vorgelegt von

Tobias Schiedhelm
Matr.-Nr.: 20855

am 13. Februar 2013
an der Hochschule der Medien Stuttgart

Erstprüfer: Prof. Oliver Curdt
Zweitprüfer: Prof. Dipl.-Ing. Uwe Schulz

Eidesstattliche Versicherung

Name: Schiedhelm Vorname: Tobias

Matrikel-Nr.:20855 Studiengang: Audiovisuelle Medien

Hiermit versichere ich, Tobias Schiedhelm, an Eides statt, dass ich die vorliegende Bachelorarbeit mit dem Titel „Sound und Sound Design in First Person Shooter: Theorie und praktische Umsetzung mit Hilfe der CryENGINE 3“ selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der eidesstattlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester) der HdM) sowie die strafrechtlichen Folgen (gem. § 156 StGB) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.

Heilbronn, den 13.03.2013 _____

Ort, Datum Unterschrift

Kurzfassung / Abstract

Computerspiele stellen ein enormes Unterhaltungsmedium dar. Sogenannte „Gamer“ findet man inzwischen in allen Altersgruppen, über alle sozialen Schichten und alle Bildungsniveaus hinweg. Viele First-Person-Shooter (FPS) zählen dabei zu den meistverkauften Spielen. Allerdings stand im Rahmen der Spieleproduktion lange Zeit lediglich das Visuelle im Vordergrund. Auch in der gängigen Fachliteratur existieren mehrheitlich Studien zum visuellen Design, wohingegen sich nur wenige Autoren mit dem Sound Design in Computerspielen beschäftigen. Bei der Spieleproduktion hingegen wurde inzwischen das Potenzial des Akustischen erkannt, da ein optimales Sound Design zur Authentizität der „virtuellen Realität“ beiträgt, dem Spieler Feedback gibt und Emotionen transportieren kann. Das Ziel der Bachelorarbeit besteht darin, das Potenzial und die Besonderheiten des Sound Designs in FPS anhand von praktischen Sound-Beispielen mit Hilfe der CryENGINGE 3 aufzuzeigen. Die Besonderheiten liegen dabei hauptsächlich in der Interaktivität und Nonlinearität des Mediums und der daraus resultierenden dynamischen Tonarbeit.

Computer games depict an enormous entertainment medium. So-called “gamers” can be found in all age groups regardless of socioeconomic and educational backgrounds. Many first person shooters also known as “FPS” are amongst the top-selling games worldwide. In the beginning of the computer game phenomenon, the primary focus of game development mostly concentrated on visuals. There are multiple studies on visual design in relevant specialist literature, whereas sound design seemed to be less emphasized among authors. Meanwhile, game developers have not only recognized the potential behind acoustics but also acknowledged that combining ideal sound design can create more of an authentic “virtual reality” therefore giving the player communicative feedback such as demonstrating lifelike emotions. The purpose of this bachelor thesis is to demonstrate the potential and characteristics of sound design within FPS game types especially when used in combination with practical sound-examples such as CryENGINE 3. Overall, the peculiarities lie primarily in the interactivity and nonlinearity of the medium therefore resulting in dynamic sound work.

Inhaltsverzeichnis

1 Einleitung

Die aktuellen Zahlen des Bundesverbands Interaktive Unterhaltungssoftware e.V. besagen, dass die deutsche Game-Industrie im Jahr 2012 insgesamt 73,7 Millionen Computer- und Videospiele verkauft hat. Im Vergleich zum Vorjahr bedeutet dies eine Steigerung um vier Prozent. Insgesamt betrug 2012 der Umsatz mit Computer- und Videospielsoftware 1,85 Milliarden Euro. In Deutschland spielen knapp 25 Millionen Personen regelmäßig Computer- und Videospiele[1].

Die Computerspielbranche ist ein ansteigender Markt, wobei bei der Spieleproduktion lange Zeit das Visuelle im Vordergrund stand. Erst aufgrund von technischen Entwicklungen und der Erkenntnis, dass beim Spieler alle Sinnesreize angesprochen werden müssen, um ein realistisches Spielerlebnis zu erreichen, wurde das Potenzial des Akustischen erkannt (Leenders, 2012).

Inzwischen schenken die Entwickler neben dem visuellen Design auch dem Sound Design immer größere Aufmerksamkeit. Trotzdem existieren - vor allem in deutschsprachigen Raum - nur wenige wissenschaftliche Veröffentlichungen zu Ton und Musik in Computerspielen. Im Vergleich zur Vielzahl der Publikationen zur Tongestaltung in Filmen, existiert nur eine geringe Anzahl an Autoren, die sich diesem Thema auch in Bezug auf Computerspiele widmet (ebd.).

Dabei kommen dem Sound Design in Computerspielen mehrere Bedeutungen zu. Erst ein authentischer Sound ermöglicht im Zusammenspiel mit einem stimmigen visuellen Design die perfekte Darstellung einer virtuellen Realität, in die der Spieler eintauchen kann. Zum anderen dient der Sound dem Spieler auch als Feedbackmöglichkeit und kann zudem Emotionen transportieren. Diese Aspekte zusammen genommen sind entscheidend für den großen Unterhaltungsfaktor und den enormen Boom der Computerspiele.

Neben den Adventures- und Strategiespielen, beherrschen vor allem Shooter den Markt der meist verkauften Spiele[2]. Deshalb widmet sich die vorliegende Arbeit dem Thema Sound und Sound Design im Genre der First-Person-Shooter (FPS). Dabei soll die Um-

[1] Vgl. *www.biu-online.de/de/fakten/marktzahlen.html* / *www.biu-online.de/de/fakten/gamer-statistiken.html*

[2] Die Zeitschrift PC Games veröffentlichte 2011 die fünfzehn meist verkauften Spiele – darunter sind auch einige Shooter zu finden (*www.pcgames.de/Panorama-Thema-233992/Specials/Die-15-meistverkauften-PC-Spiele-aller-Zeiten-und-ihre-Verkaufszahlen-PCG-Top-Artikel-Mai-2010-747800/)*

setzung anhand der CryENGINGE 3 mit Hilfe von Beispielvideos aufgezeigt werden, da mit dieser Engine einer der meist verkauften Shooter umgesetzt wurde und diese zudem frei zugänglich ist. Die Beispielvideos dienen der Veranschaulichung und können direkt über QR-Codes mit Hilfe eines internetfähigen Smartphones oder von der beiliegenden DVD abgespielt werden. Alternativ können die Beispiele über die Links im Anhang aufgerufen werden. Die Beispiele sind jeweils numerisch aufgelistet.

Die Arbeit gliedert sich dabei sowohl in einen theoretischen als auch praktischen Teil. Zu Beginn erfolgt in Kapitel 2 eine kurze Einführung in das Thema „Computerspiel". Dabei wird zuerst auf dessen Unterhaltungsfaktor und Nutzung im Allgemeinen eingegangen, um anschließend die Besonderheiten des Genres der FPS aufzuzeigen. Aufgrund der immer wiederkehrenden öffentlichen Diskussion um die Auswirkungen solcher Spiele, wird das Thema Gewalt kritisch diskutiert3.

In Kapitel 3 erfolgt die theoretische Fundierung der Arbeit, indem die Begriffe Sound und Sound Design, sowie das Berufsfeld des Sound Designers vorgestellt werden. Anschließend werden die Wirkung von Sound und die geschichtliche Entwicklung von Sound speziell in Computerspielen systematisch aufgezeigt. Das Kapitel schließt dabei mit den Besonderheiten des Sounds in Computerspielen, indem insbesondere auf die Aspekte der Interaktivität und Nonlinearität eingegangen wird. Diese Besonderheiten spielen bei der Umsetzung des Sound Designs eine entscheidende Rolle und werden im Hauptteil der Arbeit nochmals anhand der Praxisbeispiele aufgegriffen.

In Kapitel 4 wird die in der Arbeit verwendete CryENGINGE vorgestellt. Dabei wird kurz die Firmengeschichte angerissen und es werden die wichtigsten Features der Engine präsentiert, die für den weiteren Verlauf der Arbeit wichtig sind und dem Verständnis der Praxisbeispiele dienen.

Im Anschluss daran erfolgt in Kapitel 5 die praktische Umsetzung des Sound Designs mit Hilfe der CryENGINE 3. Der Fokus der Sound-Beispiele liegt auf den für FPS typischen Sounds, und zwar auf den Waffensounds, den materialabhängigen Sounds und den Fahrzeugsounds, sowie der Vertonung von Atmos.

---

[3] Jedoch liegt der Fokus der Arbeit nicht auf der Beantwortung der zentralen Fragen der Medienwirkungsforschung, das heißt, die Arbeit untersucht nicht den Zusammenhang zwischen realer Gewalt und dem Spielen von Shootern, sondern stellt lediglich die Besonderheiten des Sounds in diesem Genre heraus und legt den Fokus auf dessen praktische Umsetzung.

Abschließend erfolgt das Fazit, das nochmals die Besonderheiten des Sounds und Sound Designs in FPS anhand der theoretischen Fundierung und der praktischen Umsetzung zusammenfasst.

Das Ziel der Arbeit besteht insgesamt darin, die Besonderheiten beim Sound Design von Computerspielen mit Hilfe von Praxisbeispielen speziell für das Genre der FPS aufzuzeigen. Die Besonderheiten liegen dabei hauptsächlich in der Interaktivität und Nonlinearität des Mediums und der daraus resultierenden dynamischen Tonarbeit, die bei der Umsetzung der Beispiele berücksichtigt wurde.

2 Das Computerspiel-Genre „First Person Shooter“

Dieser Abschnitt geht zu Beginn auf das Medium Computerspiel im Allgemeinen ein, um anschließend dessen Bedeutung speziell als Unterhaltungsmedium und dessen Nutzung aufzuzeigen. Dabei werden die Besonderheiten des Computerspielens hervorgehoben. Diese allgemeine Einführung in die Thematik ist für den Verlauf der Arbeit unerlässlich, da Computerspiele ein großes Unterhaltungspotenzial besitzen und ein ansteigender Markt vorherrscht. Auch auf die Bedeutung der fortschreitenden technische Entwicklung wird kurz eingegangen. Dies führt auch dazu, dass neben visuellen Aspekten bei der Computerspielentwicklung auch das Thema Sound bzw. Sound Design stark an Bedeutung in der Spieleindustrie gewinnt (vgl. *2.2*). Im Anschluss daran erfolgt eine kurze Einführung in die Genreeinteilung von Computerspielen, wobei der Fokus auf das Genre der First-Person-Shooter (FPS) liegt. Abschließend wird kurz das Thema Gewalt im FPS-Genre angerissen, da dies häufig in der öffentlichen Kritik und Diskussion steht.

2.1 Das Medium Computerspiel

Mit dem im Jahr 2001 vom Filmwissenschaftler Mark Wolf veröffentlichten Buch *„The Medium of the Video Game“* wird bereits anhand des Titels deutlich, dass es sich beim Computerspiel um ein eigenständiges Medium handelt (Günzel, 2012).

Dabei ist der Begriff „Medium“ im Zusammenhang mit dem Computerspiel häufig zu finden. In der Computerspielforschung wird es unter anderem als „narratives Medium“ bezeichnet, das eine Geschichte erzählt – ähnlich wie in einem Film oder Roman. Darüber hinaus wird es auch als „interaktives Medium“ verstanden, da sich eine Form von Kommunikation zwischen spielenden Menschen ergibt und der Spieler[4] sich aktiv in das Geschehen einbringen kann (ebd.). Insgesamt taucht jedoch häufig der Begriff „Unterhaltungsmedium“ auf, welches, im Gegensatz zum klassischen Unterhaltungsmedium

[4] Aus Gründen der besseren Lesbarkeit wurde in der Regel die männliche Schreibweise verwendet. Es soll an dieser Stelle darauf hingewiesen werden, dass sowohl die männliche als auch die weibliche Schreibweise gemeint ist.

Fernsehen, spezifische Besonderheiten[5] aufgrund des oben genannten interaktiven Charakters[6] aufweist (Wünsch & Jenderek, 2009).

Günzel (2012) bezeichnet das Medium Computerspiel schlicht als „[…] Erscheinungen auf einem Bildschirm […]“ und konstatiert dessen Besonderheit nicht „[…] allein in der Interaktivität, sondern […] in der Manipulationsmöglichkeit des interaktiven Bildes selbst […]“ (Günzel, 2012: 17). Computerspiele können demnach schlicht als „erspielte Bilder“ bezeichnet werden (ebd.).

2.2 Computerspielen als Unterhaltung

Heutzutage fallen im Alltag häufig die Begriffe „Spaßgesellschaft“ und „Unterhaltungszeitalter“ (Früh, 2002). Als sogenanntes Unterhaltungsmedium kommt dem Computerspiel eine besondere Bedeutung zu, wobei das Wort „Unterhaltung“ nach Früh (2002) ein „Allerweltsbegriff“ ist, den es genauer zu definieren gilt.

Früh (2002) definiert Unterhaltung als eine „[…] angenehm erlebte Makroemotion[7] […] unter der Bedingung, dass der Rezipient […] die Gewissheit hat, die Situation souverän zu kontrollieren“ (Früh, 2002: 240). Unterhaltung wird somit als angenehm empfunden und ist mit Souveränität bzw. Kontrolle verbunden. Dies bedeutet, dass der Rezipient alleinige Dispositions- und Entscheidungsfreiheit besitzt, wie zum Beispiel durch die Freiheit welches Computerspiel er spielen möchte oder welchen Film er ansehen möchte (Wünsch & Jenderek, 2009). Demnach gelten nach Früh (2002) zwei Prämissen: Zum einen ist Unterhaltung ein positives Erleben und zum anderen ist Unterhaltung selbstbestimmt. Dies bedeutet, dass sie nicht gefordert oder erzwungen werden kann.

Die Unterhaltung spielt im Zusammenhang mit dem Spiel-Erleben eine zentrale Rolle. Ein Unterhaltungsmerkmal ist das sogenannte *Flusserleben* oder aber auch *Flow-*

---

[5] Weitere Besonderheiten entstehen aufgrund der Technik, die im nachfolgenden Abschnitt nochmals aufgegriffen werden.

[6] Zur Interaktivität vgl. Abschnitt 3.5.

[7] Das Modell des Unterhaltungserlebens nach Früh und Wünsch (2007) unterscheidet mediale Emotionen auf der Mikro-und Makroebene: „Sie nehmen an, dass die einzelnen, durch Medieninhalte ausgelösten Emotionen eine Mikroebene bilden, die durch weitere Verarbeitungsprozesse auf der Makroebene zu einem ganzheitlichen Unterhaltungserleben integriert wird. Während auf der Mikroebene eine Vielzahl unterschiedlicher, auch negativer Emotionen auftreten kann, ist die ›Makroemotion Unterhaltung‹ durch eine positive Valenz gekennzeichnet. Ausschlaggebend für die positive Valenz auf der Makroebene ist das Erleben von Souveränität und Kontrolle im Umgang mit Emotionen auf der Mikroebene. Früh und Wünsch beschreiben Unterhaltung somit als ein emotionales Phänomen, das nicht primär durch Medieninhalte, sondern durch die Weiterverarbeitung der erlebten Emotionen auf einer Makroebene hervorgerufen wird“ (Bartsch, Eder & Fahlenbrach, 2007: 23).

Erleben, das während des Spielens eintreten kann. Der Spieler „[…] verliert dabei das Bewusstsein, sich in einem anderen Realitätsrahmen zu bewegen“ (Wünsch & Jenderek, 2009: 46). Darüber hinaus erfolgt der von Früh (2002) bezeichnete *kontrollierte Kontrollverlust*, der eine spezifische Form des Unterhaltungserlebens ist. Der kontrollierte Kontrollverlust kann folgendermaßen beschrieben werden:

> „Das Phänomen baut auf dem Vorhandensein von zwei verschiedenen Realitätsebenen auf, mit denen ein Spieler konfrontiert wird: Zum einen die Ebene der realen Umwelt, in welcher das Spielbrett oder die Spielkonsole aufgebaut wird, zum anderen die Ebene der Spielrealität mit ihren eigenen Regeln. Dies ermöglicht, dass sich eine Person dem Spiel ausliefert, indem sie sich dessen Realität unterwirft, also ‚mitspielt‘. Dadurch ist der Spieler nicht mehr souveräner Herr über seine Handlungen und deren Folgen. Er muss – im Rahmen des Spieles- auf das Spielgeschehen reagieren und kann dabei auch die Kontrolle über die eigenen Handlungen oder das Spielgeschehen insgesamt abgeben. […] Durch Beenden des Spieles kann er stets die Kontrolle über seine Handlungen zurückerhalten – mit anderen Worten: Sein Kontrollverlust ist kontrolliert“ (Wünsch & Jenderek, 2009: 46).

Diese Unterhaltungsmerkmale weisen sowohl Spiele im Allgemeinen als auch Computerspiele im Besonderen auf (ebd.)

Speziell in Computerspielen weist das Spiel-Erleben neben dem oben beschriebenen *Flow-Effekt* und dem *kontrollierten Kontrollverlust* eine weitere Besonderheit auf bzw. steht vor einer besonderen Herausforderung, und zwar ist das Ziel das sogenannte *Präsenzerleben*. Beim Präsenzerleben geht es um die Frage, wie Personen in einer virtuellen Realität ein Gefühl der tatsächlichen Anwesenheit bzw. einem „being there“ entwickeln können. Dieses Gefühl der Präsenz steht dabei im engen Zusammenhang mit den technischen Besonderheiten des Computerspielens (ebd.).

Zum einen spielen der „Umfang und die Glaubwürdigkeit der sensorischen Informationen […]“ (ebd.) eine besondere Rolle, um dem Spieler ein Präsenzerleben zu ermöglichen:

> „Je mehr Sinneskanäle beim Nutzer konsistent in einer naturgetreuen Form angesprochen werden, desto besser entsteht ein Präsenzerleben. Bei Computerspielen wurden bisher vor allem die visuelle Komponente (Grafik) und die akustischen Reize im Rahmen des technisch Möglichen eingesetzt, um Präsenzerleben zu unterstützen. Neuere Entwicklungen sprechen auch dem hapitschen Kanal an (Vibrieren des Pads im Rhythmus des Herzschlags, wenn der Avatar Angst hat) oder verlangen vom Spieler die Nachahmung realistischer Bewegungen statt der Verwendung von Tastenkombinationen (Wii) […]“ (Wünsch & Jenderek, 2009: 52).

Zum anderen ist die „Kopplung der sensorischen und visuellen Informationen […]“ (ebd.) für das Präsenzerleben des Spielers von Bedeutung. Dies bedeutet, dass je präzi-

ser die Reaktion des Spiels auf die Steuerung des Spielers ausfällt, desto eher kann ein Gefühl der Präsenz entstehen. Technische Mängel, wie beispielsweise „Ruckeln“ im Bildaufbau oder ein „Hängen“ des Rechners verhindern jedoch dieses Präsenzerleben und haben zudem negative Auswirkungen auf das *Flow*-Erleben (ebd.).

Darüber hinaus wird dem Spieler bei technischen Mängeln bewusst, dass es sich nicht um die „Realität“ handelt und es kommt somit auch nicht zu einem kontrollierten Kontrollverlust, der - wie zuvor beschrieben - ebenso für das Unterhaltungserleben von Bedeutung ist (ebd.).

Insgesamt muss festgehalten werden, dass die technischen Bedingungen des Computerspiels entscheidend auf das Spiel-Erleben und insbesondere auf das Gefühl der Präsenz wirken – insbesondere die visuellen und akustischen Reize sind hier von enormer Wichtigkeit. Die Leistung der Computersysteme und die Qualität der Spieleprogrammierung prägen das Unterhaltungsmedium. Aufgrund von technischem Fortschritt in Form von Weiterentwicklung der Rechner- und Programmiertechnik wird das Spiel-Erleben immer realistischer. Dadurch erhält der Spieler das Gefühl sich direkt „in“ der Spielwelt zu befinden und es ermöglicht ihm mit den virtuellen Figuren innerhalb der Spielwelt zu interagieren. Die Figuren werden als „reale“ Personen wahrgenommen, wodurch „[…] das Spiel-Erleben als Äquivalent zum Erleben realer Sozialkontakte“ (Wünsch & Jenderek, 2009: 54) angesehen werden kann. Darüber hinaus identifiziert sich der Spieler mit der Rolle seiner Spielfigur (ebd.).

Nach einer näheren Betrachtung des Phänomens „Unterhaltung“ kann alles in allem festgehalten werden, dass sich Computerspiele besonders gut zur Unterhaltung eignen und zu Recht als Unterhaltungsmedium bezeichnet werden können. Insbesondere das Flow- und Präsenzerlebens, aber auch der kontrollierte Kontrollverlust prägen das Unterhaltungsgefühl im Computerspiel. Durch das Vorhandensein einer künstlichen Umgebung wird dem Spieler ermöglicht, sich in einer geschützten „Realität“ zu bewegen. Dabei liegt die Besonderheit in der stark aktiven Form der Unterhaltung, da der Spieler selbst das eigene Handeln gestalten kann. Dies kann vor allem dadurch erreicht werden, indem beim Spieler (fast) alle Sinnesreize angesprochen werden. Dies setzt eine entsprechende technische Umsetzung im Rahmen der Spieleentwicklung voraus und ist auch für das Sound Design von enormer Bedeutung (ebd.).

2.3 Die Nutzung von Computerspielen

Trotz der großen Relevanz des Computerspiels als Unterhaltungsmedium und dessen Aufstieg zu einem „Schlüsselmedium“ existieren nur unzureichende differenzierte Forschungen zur Spielenutzung. Klimmt (2009) betrachtet die Computerspiel-Forschung aus einer interdisziplinären und internationalen Perspektive und stellt dabei fest:

> „Der Mangel an wissenschaftlichen Erkenntnissen im Forschungsbereich Computerspiele beginnt bereits bei der einfachen Quantifizierung der Computerspielnutzung. Weltweit gibt es überraschend wenige, zumeist disparate und wenig kontinuierlich betriebene Forschungsbemühungen, die Verbreitung […] und den zeitlichen Nutzungsumfang […] von Computerspielen zu bemessen […]. Die Datenlage zur Nutzung von Computerspielen ist im Vergleich zur Nutzung konventioneller Massenmedien ausgesprochen dürftig“ (Klimmt, 2009: 58).

Hierfür werden von Klimmt (2009) zwei Gründe aufgeführt: Zum einen der Mangel an persönlichen Erfahrungen und Alltagswissen der Forscher über Computerspiele. Zum anderen die mangelnde kommerzielle Mediaforschung in diesem Bereich, da Computerspiele als Werbeträger bisher weitgehend unbeachtet blieben. Letzteres scheint sich dem Medienwissenschaftler zufolge jedoch zukünftig zu ändern.

Eine Problematik bei der Untersuchung von Spielenutzung ist das vorherrschende einseitige empirische Datenmaterial. In der bisherigen Computerspiel-Forschung dominieren Befragungsstudien zur Nutzung von Computerspielen, die auf eigenen Aussagen der Nutzer basieren. Hier stellt sich die Frage nach der Validität, das heißt, der Gültigkeit der Aussagen. Aufgrund von „sozialer Erwünschtheit“ (vgl. hierzu Esser, 1986) kommt es in diesem Fall häufig zu Verzerrungen.

Aus den bestehenden Forschungen lassen sich trotz dieser Einschränkungen einige allgemeine Aussagen über die Entwicklungen der Spielenutzung der wichtigsten Spielemärkte, wie Japan, USA, Großbritannien und Deutschland, ableiten (Klimmt, 2009):

1. Erstens kann eine massenhafte Verbreitung von Computerspielen als Unterhaltungsmedium insbesondere bei jungen Altersgruppen festgestellt werden.
2. Zweitens wird ein langfristiger Anstieg des Spielerkreises beobachtet werden, das heißt, von Personen, die bisher selten Computerspiele nutzten.

Allerdings muss beachtet werden, dass bei diesen Pauschalaussagen eine Messungs- und Vergleichsproblematik besteht, da die Mediennutzungsform von Computerspielen inzwischen stark ausdifferenziert und unterschiedlich ist, wie zum Beispiel in Form von Online-Spielen, mobiles Spielen oder LAN-Partys (ebd.). Auch müssten die unter-

schiedlichen Genres der Computerspiele einzeln betrachtet werden, um konkretere Aussagen zu tätigen.

Insgesamt kann jedoch festgehalten werden, dass die Nutzung von Computer,- Konsolen- und Onlinespiele immer weiter ansteigt – insbesondere bei jüngeren Generationen. Jedoch steckt die empirische Medienforschung bezüglich der Verbreitung und des Nutzungsumfangs von Computerspielen noch in den Kinderschuhen (ebd.).

2.4 Das Genre der First-Person-Shooter (FPS)

Dieser oben angeführte Forschungsbedarf trifft auch auf das Genre der First-Person-Shooter (FPS) zu. Zwar rückt aufgrund der in die Kritik geratenen „Killerspiele“ die Wirkungsfrage von solchen Spielen in den Mittelpunkt wissenschaftlicher Betrachtung, jedoch bleibt die Frage nach der Spielenutzung von FPS weitgehend aus (Lehmann et al., 2009). Um jedoch zu untersuchen, wie und warum sich die Nutzer intensiv mit FPS beschäftigen, muss eine differenziertere Betrachtung auf das Genre selbst erfolgen.

Der Medien- und Kommunikationswissenschaftler Christoph Klimmt (2001) beschreibt den Forschungsbedarf zu Computer-und Videospiele im Allgemeinen wie folgt:

> „Computer- und Videospiele stellen die Medien- und Kommunikationswissenschaft vor neue Herausforderungen: Angesichts ihrer rasanten Verbreitung und ihrer herausragenden Stellung unter den medialen Unterhaltungsangeboten besteht erheblicher Forschungsbedarf. Bisher liegen jedoch nur wenige Abhandlungen und noch weniger empirische Studien zum Umgang mit Computer- und Videospielen vor. Selbst eine einheitliche Terminologie und Genreeinteilung für diesen neuen Forschungsgegenstand existiert (noch) nicht“ (Klimmt, 2001: 480).

2.4.1 Genreaufteilung

Betrachtet man die Praxis der Computerspiele-Zeitschriften, zeigt sich eine Fülle von Bezeichnungen und Mischformen (Klimmt, 2001). Beispielsweise klassifiziert die Zeitschrift „Game Star“ fünf verschiedene Genres, und zwar Actionspiele[8], Strategiespiele, Sportspiele, Simulationen und Adventure-Spiele (ebd.). Nach Klimmt (2001) besteht jedoch folgendes Problem bei dieser Einteilung:

> „[…] die Übergänge zwischen „Simulation“ und „Actionspiel“ sind genauso fließend wie die von „Strategiespielen“ und „Sportspielen“. Wiederum besteht das Problem in der Vermischung von inhaltlichen Beschreibungen und Spielanforderungen“ (Klimmt, 2001: 483).

[8] Hierzu zählt die Zeitschrift 3D Shooter und Prügelspiele.

Im Gegensatz dazu beschränkt sich die Zeitschrift „PC Games“ auf vier verschiedene Genres, nämlich Strategie-, Action[9]-, Abenteuer-, und Sport-Spiele. Ebenso besteht hier die Problematik einer eindeutigen Klassifikation (ebd.).

Insgesamt weisen die verschiedenen Klassifikationen im Rahmen der Genreaufteilung einige Schwächen auf, da häufig Überlappungen unvermeidlich sind. Diese Problematik besteht jedoch nicht nur bei der Genreaufteilung in der Praxis von Computerspiele-Zeitschriften, sondern auch bei den verschiedenen Klassifizierungen in der Medienpädagogik, der Kommunikationswissenschaft und in der (Medien-) Psychologie[10]. Aus diesem Grund entwickelte Klimmt (2001) ein eigenes Kategorisierungsschema. Dabei werden die Computerspiele anhand dieser Merkmale kategorisiert:

1. **Der narrative Kontext:** Unter „narrativer Kontext“ versteht man, dass ein Spiel in eine Geschichte eingebettet ist. Dabei gehören zwei Arten von Informationen dem narrativen Kontext an. Einerseits die „Hinweise über die Beschaffenheit der Spielwelt […]“ und die „[…] Beschreibungen der Rolle, welche die Spieler/innen in der Spielwelt einnehmen“ (Klimmt, 2001: 490).
2. **Die Aufgaben der Spieler:** Wie in den Abschnitten 2.1 und 3.5 beschrieben, ist das zentrale Element des Computerspielens der interaktive Aspekt der Unterhaltung. Das Lösen von Aufgaben und Problemstellungen sind dabei die wesentlichen Aktionen beim Spielen: „Computer- und Videospiele setzen ihre Nutzer/innen aber auch unter Druck, diese Handlungsmöglichkeiten zu nutzen: Antagonisten, Konkurrenten und widrige Umstände schaffen immer neue Probleme und Bedrohungen, auf die die Spieler/innen reagieren müssen (Klimmt, 2001: 491).
3. **Die mediale Präsentation:** Mit „medialer Präsentation“ ist die Präsentationsform des Spiels gemeint, die wiederum eng mit dem narrativen Kontext zusammenhängt. Der narrative Kontext wird in Form von einer Rahmenhandlung mit Hilfe von Bildern und Klängen erzählt. Dabei ist die Präsentationsform dieser Rahmenhandlung abhängig von der technischen Umsetzung des Spiels: „Neue Grafik- und Sound-Chips ermöglichen immer detailliertere und reichhaltigere Eindrücke von der Spielwelt“ (Klimmt, 2001: 492). Zur medialen Präsentation zählt Klimmt (2001) unter anderem die Aspekte „Raum“ und „Zeit“.

[9] Hierzu zählt die Zeitschrift auch First Person Shooter.

[10] Auf die Genreeinteilung in den verschiedenen Wissenschaftsdisziplinen soll hier nicht näher eingegangen werden, da der Fokus auf das Genre FPS liegen soll. Vgl. hierzu Klimmt (2001).

2.4.2 FPS nach dem Kategorisierungsschema nach Klimmt (2001)

Auch innerhalb des Genres der FPS selbst existiert eine Vielzahl von unterschiedlichen Spielen, die sowohl Gemeinsamkeiten als auch Unterschiede aufweisen (Lehmann et al., 2009). Unter Berücksichtigung des oben beschriebenen Kategorisierungsschemas nach Klimmt (2001) weisen FPS folgende Gemeinsamkeiten auf:

1. **Der narrative Kontext:** Beim narrativen Kontext geht es um die „Rolle“ des Spielers in der Spielewelt. Beim Genre des FPS nimmt der Spieler stets die Rolle des „Kämpfers“ ein, der „[…] sich allein oder im Team seinen virtuellen Aufgaben stellt“ (Lehmann et. al. 2009: 242).
2. **Die Aufgaben:** Die Gemeinsamkeiten bei den Aufgaben in FPS beziehen sich dabei auf den Aspekt der Geschwindigkeit. „Ein Großteil der zu bewältigenden (Kampf-) Aufgaben kann nur durch hohe (Reaktions-) Schnelligkeit gemeistert werden, welche eine zentrale Spieleigenschaft bei FPS ist“ (Lehmann et. al. 2009: 242).
3. **Die mediale Präsentation:** Für die mediale Präsentation spielen die beiden Komponenten „Zeit“ und „Raum“ eine wesentliche Rolle. Im Genre des FPS werden diese beiden Komponenten immer auf die gleiche Art und Weise umgesetzt, und zwar erfolgt die räumliche Darstellung der Spielwelt immer in der Ego-Perspektive und die Handlung selbst erfolgt in Echtzeit (ebd.).

Neben den vorgestellten Gemeinsamkeiten der Spiele im Genre des FPS existieren auch Unterschiede im Genre selbst. Beispielsweise in Bezug auf das „Setting“, das dem narrativen Kontext zuzuordnen ist (vgl. hierzu das Kategorisierungsschema nach Klimmt, 2001: 490ff.). Ein wesentlicher Unterschied zwischen den verschiedenen Spielen der Kategorie FPS besteht in Hinsicht auf den „Realitätsbezug“. Zum einen weisen einige Spiele einen geschichtlichen Hintergrund und historische Schauplätze als Setting auf, wohingegen andere Spiele wiederum „futuristische Handlungsumgebungen“ zeigen (Lehmann, et al. 2009: 242). Des Weiteren existieren Unterschiede innerhalb des Genres in Bezug auf den Komplexitätsgrad eines Spiels (vgl. hierzu die Ausführungen von Klimmt, 2001) und auf „ […] das Ausmaß der im Spiel dargestellten Gewaltintensität“ (Lehmann et. al. 2009: 243).

Insgesamt ist festzuhalten, dass die Spiele innerhalb des Genres des FPS sowohl Gemeinsamkeiten als auch Unterschiede aufweisen (ebd.).

2.4.3 Kritik am Genre unter Berücksichtigung des Gewaltbegriffs

Wie zuvor beschrieben, variiert die Gewaltintensität in den verschiedenen Spielen des FPS. Zwar sind Gewalt und Kämpfe zentral für das Genre, jedoch unterscheidet sich deren Darstellungsform in den einzelnen Spielen. Beispielsweise wird der Tod eines Gegners in den verschiedenen Spielen auch unterschiedlich inszeniert, wobei einige Spiele diesen sehr blutig darstellen und im Gegensatz dazu andere Spiele wiederum auf eine grausame Darstellung verzichten (Lehmann et. al. 2009).

Das Thema „Gewalt“ in Computerspielen wird häufig diskutiert. Insbesondere nach einem Amoklauf wie in Winnenden im Jahr 2009 wird die öffentliche Diskussion um sogenannte „Killerspiele“ entfacht (Bareither, 2012). Dies stellt auch Thimm (2010) in den Ausführungen zu „Spiel als Thema des öffentlichen Diskurses“ fest und schreibt dazu folgendes:

> „Der Zusammenhang zwischen Spiel und Medien umfasst ein Spannungsfeld, das in der massenmedialen Berichterstattung zumeist dann aktualisiert wird, wenn es im Kontext problematischer und konfliktärer gesellschaftlicher Ereignisse auf der Medienagenda steht. Die zerstörerischen Gewaltausbrüche jugendlicher Amokläufer führen regelmäßig zur Hinterfragung der gesetzlichen Rahmenbedingungen von Computerspielen und sind Anlass für ein sich wiederholendes Muster in der öffentlichen Debatte: Starke Schuldzuweisungen aus Politik und Teilen besorgter Elternschaft und nicht minder heftige Verteidigung des Rechts auf Spiel durch Tausende nicht-gewalttätige Spielefans“ (Thimm 2010: 7).

Aus diesem Grund wird - im Gegensatz zur allgemeinen Mediennutzung des Computerspiels - die Medienwirkung von FPS in Bezug auf das Thema Gewalt intensiv erforscht:

> „Während die Popularität interaktiver Unterhaltungsangebote unumstritten ist, wird über ihre Konsequenzen heftig debattiert. Im Mittelpunkt steht dabei die Beobachtung, dass ein Großteil der verfügbaren Computer- und Videospiele Gewalthandlungen beinhaltet […]. Deshalb wird hauptsächlich über mögliche aggressionsfördernde Spielwirkungen diskutiert […] und geforscht […]“ (Klimmt, 2001: 481).

Allerdings besteht bei der Erforschung des Themas Gewalt in FPS die Problematik, dass dieser Aspekt nur einseitig aus Sicht der (Medien-) Psychologie oder Pädagogik betrachtet und nicht im Kontext des eigentlichen Spieleprozesses untersucht wird (Bareither, 2012). Bareither (2012) kritisiert zudem eine nicht ausreichende Definition des Gewaltbegriffs in FPS und stellt dabei fest:

> „Oftmals wird schlicht vorausgesetzt, dass in Spielen, in denen Repräsentationen physischer Gewalt vorkommen auch eine Art von Gewalt stattfinde. Der entsprechende Gewaltbegriff bleibt dabei diffus oder wird nicht weiter thematisiert. Meine Kritik an solchen Ungenauigkeiten zielt nicht darauf ab, die Befunde dieser Studien in Bezug auf je-

> ne Wirkungen infrage zu stellen, sondern soll vielmehr auf die Probleme verweisen, die ein undifferenzierter Gewaltbegriff mit sich bringt“ (Bareither, 2012: 94).

Dabei kritisiert er vor allem den virtuellen Gewaltbegriff nach Fritz und Fehr (2003). Diese beschreiben „virtuelle Gewalt“ als:

> „[…] Ähnlichkeitsbeziehung zwischen der realen Gewalt im Kampfsport und dem Geschehen auf dem Bildschirm, in das der Spieler handelnd einbezogen ist. Die Ähnlichkeit ist hergestellt; sie ist eine Konstruktionsleistung des menschlichen Gehirns, das in der Lage ist, zu ergänzen und hinzuzudenken, was der virtuellen Welt im Vergleich zur realen fehlt, um so in der Virtualität ein emotionales Erleben zu erreichen, das der Spieler wünscht: Das Gefühl von Macht durch virtuelle Entfaltung aggressiver Impulse“ (Fritz & Fehr, 2003: 49).

Der virtuelle Gewaltbegriff scheint nach Bareither (2012) ungeeignet zu sein, da die nicht-physische, „virtuelle Gewalt“ lediglich auf der emotionalen Ebene stattfindet, das heißt, dem Spieler ein Gefühl von Macht verleiht. Das Gefühl von „Macht“ und die „virtuelle Entfaltung aggressiver Impulse“ (Fritz & Fehr, 2003) können Bareither (2012) zufolge jedoch nicht mit realer Gewalt gleichgesetzt werden.

In der Medienwirkungsforschung wird Gewalt auch häufig synonym zum Begriff der Aggression verwendet. Mehrheitlich werden mit Gewalt extreme Formen der Aggression beschrieben, wie zum Beispiel die physische Gewalt (Möller, 2006). Da jedoch im FPS keine Gewalt in Form von physischer Verletzung stattfinden kann, wird deutlich, dass eine differenziertere Betrachtung des Gewaltbegriffs notwendig ist, um Kritik am Genre ausüben zu können (Bareither, 2012).

Die zentralen Fragen der Medienwirkungsforschung, ob gewalthaltige Computerspiele aggressiv machen und welchen Einfluss sie auf das Verhalten der Spieler haben, können Bareither (2012) zufolge jedoch nur beantwortet werden, sofern der Gewaltbegriff im Kontext des Spieleprozesses betrachtet und eine differenziertere Definition des Begriffs erfolgt. Auch der Medienpsychologe Malte Elson (2011) weist darauf hin, dass bei dem Vergleich von Spielen im Rahmen der Medienwirkungsforschung der Einfluss von Gewalt auf das Verhalten der Nutzer nur schwer zu messen ist, da die Auswirkungen von Spielen nicht allein auf diesen einen Medieninhalt zurückzuführen sind. Neben dem Ausmaß an Gewalt spielen auch weitere Faktoren eine Rolle, wie zum Beispiel die Besonderheit des FPS Genres, die in der Ego-Perspektive liegt. Fraglich ist dem Medienpsychologe zufolge, ob allein diese Besonderheit, das heißt, diese Ego- Perspektive bereits irgendwelche Auswirkungen auf das Verhalten von Spielern hat, da es hierzu noch

keine systematischen Untersuchungen gibt. Ein weiterer Einflussfaktor ist Elson (2011) zufolge auch die Spielgeschwindigkeit[11], die je nach Shooter etwas variieren kann.

Insgesamt wird deutlich, dass eine berechtigte Kritik am Genre nur erfolgen kann, sofern der Gewaltbegriff in FPS differenziert betrachtet wird und eine genaue Begriffsbestimmung erfolgt (Bareither, 2012). Des Weiteren gilt es nicht alleine den Aspekt der Gewalt zu untersuchen, denn um die Fragen der Medienwirkungsforschung sinnvoll beantworten zu können reicht die Betrachtung auf einen Medieninhalt nicht aus (ebd.). Inwiefern ein Zusammenhang zwischen realer Gewalt und dem Spielen von FPS besteht, kann demnach nur beantwortet werden, sofern die oben genannten Aspekte berücksichtigt werden. Hierzu herrscht jedoch noch Forschungsbedarf vor (Bareither, 2012).

[11] In einem Experiment wurde dies untersucht, indem die Probandengruppen einmal eine schnelle und eine langsame Version eines Shooters spielten. Das Experiment zeigte tatsächlich starke Auswirkungen auf die Spieler (vgl. hierzu die Studie von Elson, 2011).

3 Sound Design in First-Person-Shooter (FPS)

In diesem Abschnitt geht es um das Thema Sound und Sound Design in FPS. Zu Beginn erfolgt eine kurze Einführung und Begriffsbestimmungen zu Sound und Sound Design im Allgemeinen sowie in das Arbeitsfeld des Sound Designers. Im Anschluss daran wird die Wirkung von Sound diskutiert, um darauf aufbauend die Besonderheiten des Sounds in Computerspielen bzw. speziell im FPS-Genre anhand von verschiedenen Beispielen aufzuzeigen. Dabei wird der Fokus auf die Aspekte „Interaktivität“ und „Nonlinearität“ gelegt.

3.1 Sound, Sound Design und Sound Designer

Bisher existieren in der medienwissenschaftlichen Literatur mehrheitlich Definitionen und Studien zum visuellen Design, wohingegen nur wenige Publikationen und Begriffserläuterungen zu Sound Design zu finden sind. Zudem liegt der Fokus in der gängigen Fachliteratur auf Filmvertonungen und weniger auf der Vertonung von Computer- bzw. Videospielen. Insgesamt existieren nur wenige Definitionen zu Sound Design und in den meisten Lexika ist der Begriff überhaupt nicht zu finden (Lensing, 2009). Nach Schätzlein (2005) besteht folgende Problematik:

> „Die meisten medienwissenschaftlichen Publikationen zum Akustischen in den Medien vermeiden eine explizite Definition des Begriffs. Oft werden innerhalb eines Textes unterschiedliche Begriffsbestimmungen nebeneinander verwendet – ohne die Definitionsproblematik und das vorhandene Bedeutungsspektrum zu thematisieren. Hier lassen sich zum einen historisch unterschiedliche Ansätze beschreiben, zum anderen haben sich in den einzelnen Fachbereichen auch unterschiedliche Perspektiven auf das Thema Sound entwickelt“ (Schätzlein, 2005: 24).

Aus diesem Grund sollen in den folgenden Abschnitten auch historische Aspekte im Rahmen der Begriffserläuterungen zu Sound und Sound Design sowohl in der Alltagssprache als auch in den verschiedenen wissenschaftlichen Disziplinen berücksichtigt werden. Zudem wird auf die Entwicklung eines neuen Arbeitsfeldes - nämlich die des Sound Designers - kurz eingegangen. Die Arbeit des Sound Designers wird mit Hilfe der praktischen Umsetzung von Beispielvideos mit Hilfe der CryENGINE schließlich in Kapitel 5 demonstriert.

3.1.1 Definition Sound

Im deutschsprachigen Raum liegt der Ursprung des Begriffs *Sound* in der Fachsprache des Jazz, womit der typische „Klang" einer Band oder eines Solisten bezeichnet wurde. Das musikalische Ziel damals war es, einen individuellen Sound bzw. Klang zu erzeugen. Eine frühe Quelle zur Verwendung des Begriffs ist die von Joachim Berendt 1953 erschienene Publikation „Das Jazzbuch", wobei das Thema Sound in seinem Glossar „Die Sprache des Jazz" erstmalig aufgegriffen wurde (Schätzlein, 2005).

Später in den sechziger Jahren spielte der Begriff „Sound" sowohl in der Rockmusik als auch in der elektronischen Popmusik eine wichtige Rolle, wobei die Tontechnik aufgrund von neuen Geräten und Verfahren in Studios zunehmend an Bedeutung gewann. Nach Schätzlein (2005) rückte der „[…] Stellenwert von Interpretation, Komposition und Notation […]" (Schätzlein, 2005. 25) gegenüber der Technik in den Hintergrund. Der Fokus liegt seither vor allem auf dem erzeugten Klang der Tonstudiotechnik und weniger auf dem Sound eines Musiker bzw. einer Band. Der charakteristische Klang eines Produzenten bzw. Tonstudios steht nun im Mittelpunkt und wird zum Markenzeichen (ebd.).

Der Begriff Sound veränderte sich nun dahingehend, dass nicht nur der Klang oder die Klangfarbe im akustischen Sinn gemeint war, sondern der Begriff umfasst inzwischen den Gesamteindruck der Musik (ebd.). Im Duden findet sich im Jahr 1973 erstmalig der Begriff „Sound" und wird schlicht als „Klang(wirkung, -richtung)" definiert (Duden, 1973: 639). Auch in weiteren Lexika und Enzyklopädien beschränken sich die Definitionen auch heute noch auf die Synonyme wie „Klangfarbe", „Klangqualität" oder „charakteristischer Klang" (Schätzlein, 2005: 26).

3.1.2 Sound in verschiedenen Fachdisziplinen

Darüber hinaus haben sich in der Alltags- und Fachsprache noch weitere begriffliche Assoziationen zum Thema Sound entwickelt, wie zum Beispiel die Tonebenen in der elektronischen Musik, Geräuscheffekte (Sound Effects bzw. sogenannte SFX) oder aber auch der charakteristische Klang einer Ware bzw. Marke eines Unternehmens im Rahmen von akustischem Produktdesign bzw. Corporate Sound (Schätzlein, 2005: 26).

Neben den verschiedenen Begriffen für Sound in der Alltagssprache existiert auch eine Vielzahl von Definitionen und Beschreibungsansätze in den unterschiedlichen wissenschaftlichen Fachdisziplinen. Beispielsweise wird der Begriff „Sound" in musikwissen-

schaftlichen Publikationen als Klang bzw. auch als Klangfarbe definiert. Darüber hinaus kann Sound auch als „[…] Synonym für einen musikalischen Stil oder Stilelemente und als spezifisch gestalteter Klang eines Musikers, einer Band, eines Labels, einen tontechnischen Geräts […]"(Schätzlein, 2005: 27) verstanden werden.

Auch in der Filmwissenschaft, die sich vorrangig mit dem Visuellen beschäftigt, gewinnt das Thema Sound an Bedeutung. Der Sound umfasst hierbei den gesamten Ton und die Musik eines Films, sowie einzelne Klänge und Geräusche im Zusammenhang mit den sogenannten Sound-Effekten (ebd.).

Neben den Film- und Medienwissenschaften spielt das Thema Sound auch in den Ingenieurswissenschaften eine entscheidende Rolle. Hierbei steht der Sound im Zusammenhang mit dem Klang eines bestimmten Produkts und soll zu dessen Kauf anregen. Des Weiteren beschäftigen sich Ingenieurswissenschaftler auch mit dem sogenannten Akustik-Design, wobei die akustischen Merkmale eines Raums verbessert werden sollen, um beispielsweise das personelle Wohlbefinden in Büroräumlichkeiten zu steigern (ebd.).

Dagegen beschäftigt sich der Fachbereich Psychologie mit der akustischen Wahrnehmung spezifischer Klänge oder aber auch mit Lärm und dessen Auswirkungen auf das personelle Wohlbefinden. Das Thema Sound findet demnach in der Psychologie vor allem in der Psychoakustik und der Umweltpsychologie Beachtung (ebd.).

Insgesamt spielt das Thema Sound in vielen Fachbereichen eine Rolle und die hier angeführten Beispiele zeigen nur einen kurzen Ausschnitt von aktuellen Forschungsbemühungen. Der Fokus der Arbeit liegt jedoch auf der medienwissenschaftlichen Forschung und praktischen Arbeit zu Sound in Computerspielen und insbesondere zum Sound Design im Genre der FPS.

3.2 Definition Sound Design

Der Begriff „Sound Design" tauchte in etwa ab Mitte der siebziger Jahre in der Medienlandschaft auf und ist auf Innovationen und Entwicklungen in der Ton- und Aufnahmetechnik zurückzuführen (Börsing, 2009; Lensing, 2009). Börsing (2009) beschreibt den Begriff „Sound Design" schlicht als „[…] die Arbeit an einem Soundtrack in seiner Gesamtheit" (Börsing, 2009: 2).

Lensing (2009) stellt in seinem Buchabschnitt „Sounddesign, was ist das?" jedoch folgendes fest:

> „Immer wieder taucht der Begriff Sounddesign in unterschiedlichen Zusammenhängen auf und ist doch nirgends genau definiert. Schlägt man eines der gängigen Lexika auf, fehlt dieser Begriff oft völlig. Den Regal füllenden Buchreihen über visuelles Design stehen nur sehr wenige Publikationen zu Sounddesign gegenüber“ (Lensing, 2006: 26).

Nach Lensing (2009) steht dabei der englische Begriff „Sound“ für „Ton“ oder „Klang“, wohingegen der Begriff „Design“ dessen Ausgestaltung meint. Demnach kann Sound-Design folgendermaßen definiert werden:

> „Sound-Design ist die Kunst, den richtigen Sound am richtigen Platz in der richtigen Zeit einzusetzen […]. Das Sound-Design umfasst genauso stilistische wie handwerkliche Fähigkeiten“ (Lensing, 2009: 38).

Im deutschsprachigen Raum wird anstatt Sound Design häufig der Begriff „Tongestaltung“ verwendet und beschreibt sowohl die kreative als auch technische Arbeit mit Klängen und Geräuschen (Leenders, 2012; Lensing, 2009).

3.2.1 Sound Designer: Die Entwicklung eines neuen Arbeitsfelds

Erstmalig tauchte die Bezeichnung Sound Design bzw. die Berufsbezeichnung des Sound Designers 1979 im Abspann des Films „*Apocalypse Now*“ von Francis Ford Coppolas Film auf. Inzwischen hat sich der Anwendungsbereich des Sound Designs auch auf weitere Medien erweitert, und zwar arbeiten heute sogenannte Sound Designer nicht nur bei Filmen, sondern auch im Rundfunk, im Rahmen von Multi-Media Anwendungen und im industriellen Kontext (Schätzlein, 2005).

Die wachsende Bedeutung von Sound Design und die Entwicklung des Berufsfeldes des Sound Designers sind - wie bereits erwähnt - auf den Fortschritt in der Audiotechnik im Laufe der letzten Jahre im Rahmen von Tonproduktionen zurückzuführen (ebd.).

Seit Anfang der neunziger Jahre hat sich das Arbeitsfeld Sound Design und die Berufsbezeichnung des Sound Designers auch in Deutschland etabliert. Flückiger (2001) beschreibt das Tätigkeitsfeld des Sound Designers nicht nur als eine rein technische, sondern auch als kreativ-ästhetische Arbeit mit Geräuschen und Sprache:

> „Die Tätigkeit des Sound Designers umfasst die Erarbeitung eines tonästhetischen Gesamtkonzepts für die Bereiche Sprache und Geräusch, die Kommunikation mit dem Komponisten, die Kreation von einzelnen Klängen und ihre Montage sowie die Koordination von Arbeitsprozessen und –zielen der verschiedenen Abteilungen inklusive Geräuschemacher und Nachsynchronisation. Anders als der Leiter des Tondepartments im Studiosystem ist der Sound Designer nicht nur Manager, sondern auch zentrale kreative Instanz […]“ (Flückiger, 2001: 18).

3.3 Psychoakustik: Die Wirkung von Sound in Computerspielen

Das Ziel bei der Arbeit des Sound Designers besteht darin, den Ton möglichst authentisch zu gestalten - insbesondere beim Sound Design von Computerspielen. Neben dieser realistischen Wiedergabe von Klängen, geht es im Sound Design zusätzlich darum, beim Spieler bestimmte Emotionen zu erzeugen. Wie zuvor beschrieben, müssen beim Rezipient möglichst alle Sinnesreize angesprochen werden, damit ein Gefühl der Präsenz und Authentizität der erspielten Realität erzeugt werden kann. Hierzu zählt auch die auditive Wahrnehmung, die bestimmte Stimmungen erzeugen kann. Raffaseder (2010) beschreibt dies folgendermaßen:

> „Akustische Ereignisse eignen sich bestens um Emotionen zu kommunizieren und Stimmungen zu regulieren [...]. Oft reichen schon kleine akustische Gesten aus, um eine Vielzahl von Gefühlsregungen zu transportieren“ (Raffaseder, 2010: 21).

Bei der auditiven Wahrnehmung werden unterbewusst Emotionen beim Rezipient ausgelöst. Menschliche Empfindungen und Wahrnehmungen werden von der Wissenschaftsdisziplin „Psychologie“ untersucht. Akustische Ereignisse, die Empfindungen auslösen, zählen wiederum zum Teilbereich der Psychoakustik. Als wissenschaftliche Disziplin beschäftigt sich die Psychoakustik mit dem Zusammenhang zwischen messbaren, physikalischen Schallereignissen und den subjektiven, psychischen Höreindrücken (Herrmann, 2009).

Der amerikanische Sound-Designer Ben Burtt spricht in diesem Zusammenhang von einem „emotionalen Wörterbuch“, auf das ein Sound-Designer zurückgreifen kann. Durch bestimmte Klänge können Schlüsselreize aktiviert werden, die spezifische Emotionen beim Rezipient auslösen. Bei der Aktivierung von Schlüsselreizen sind Erinnerungen entscheidend, das heißt, erlebte Ereignisse aus der Vergangenheit werden auch mit bestimmten Klängen verknüpft. Beispielsweise kann der Klang eines Meeresrauschens unterbewusst bestimmte Emotionen auslösen, die an einen zurückliegenden Urlaub erinnern (Lensing, 2009).

Emotionen spielen beim Konsum von Computerspielen eine entscheidende Rolle. Dabei sind die erlebten Gefühle während des Spiels von Erfolg und Misserfolg des Spielers abhängig. Zudem ist auch der Inhalt der Spiele für teilweise sehr intensive Gefühlszustände ausschlaggebend. Hinzu kommt die Wirkung von Sound- und visuellen Effekten, die ebenso Emotionen auslösen bzw. verstärken können. Beispielsweise erlebt der Spie-

ler beim Erfolg ein Hochgefühl, wie durch das Erreichen von Punkten oder einem höheren Level. Dies wird gewöhnlich durch bekräftigende Soundeffekte und grafische Signale untermauert (Möller, 2007). Der Sound hat somit auch eine Feedbackfunktion für den Spieler (Grunwald, 2008).

Hauptsächlich die Lautstärke hat einen großen Einfluss auf die Empfindungen des Spielers. Dabei kann die Lautstärke als ein expressives Mittel verwendet werden, das heißt, eine große Lautstärke beeinflusst auch die menschlichen Emotionen stark (Flückiger, 2010). Hierbei können sowohl emotionale als auch körperliche Reaktionen hervorgerufen werden. Beispielsweise beobachtete Rudolph (1993) ab einer bestimmten Lautstärke (60dB) „[…] unwillkürliche Aktivierungserscheinungen, die sich in Blutdruckerhöhung, Herzfrequenzsteigerung, Verengung der Kapillaren, erhöhtem Muskeltonus, Pupillenerweiterung etc. zeigen“ (Rudolph, 1993: 239). Darüber hinaus vermittelt eine hohe Laustärke Emotionen wie Angst, Bedrohung oder Aggression (Flückiger, 2010). Rudolph (1993) erklärt die Wirkung von extremer Lautstärke wie folgt:

> „Es gibt keine mentale Strategie, um der Wirkung exaltierter Lautstärke auszuweichen. Sie übt einen unmittelbaren und unwillkürlichen Einfluss auf die psychischen und vegetativen Funktionen aus, und zwar unabhängig von der Beschaffenheit und Bewertung des Reizes“ (Rudolph, 1993:33).

Insgesamt kann das Sound Design im Computerspiel folgende Wirkungen besitzen: Zum einen kann es den Kontext beschreiben, das heißt, es erfolgt eine möglichst authentische Wiedergabe von Klängen. Beispielsweise vermittelt ein Schussgeräusch dem Spieler, dass er gerade eine Waffe abgefeuert hat. Marks (2001) bezeichnet dies auch als „*aural feedback*“:

> „Sound effects in games serve many functions. Primarily, they are a method of aural feedback that simulates what you would hear if you were actually in the game. If I shoot my weapon and it’s not empty, it goes “bang” instead of “click.” If I hit something with it, it goes “thwack-oomph-thud.” If I depress a button and it opens a door, it goes “whoosh.” Every direct action has a corresponding sound reaction“ (Marks, 2001: 291).

Diese authentischen Sounds sind für das in Abschnitt 2.2. beschriebene Flow- bzw. Präsenserleben wichtig:

> „Sound effects can lend believability to an otherwise unbelievable place. Ambiance and general Foley sounds bring a scene to life and lend an air of realism. For example, a good driving game will use sounds of city life, wind, scenery ambience, and even animals — all of which we have heard ourselves in our travels. We expect to hear them and are satisfied when we do, even on a subconscious level. These sounds don’t have to be “in your face” to do the job, they justneed to be there somewhere in the background“ (Marks, 2001: 291).

Zum anderen kann der Sound im Computerspiel auch Emotionen erzeugen bzw. verstärken. Beispielsweise können spezifische Soundeffekte Erfolge oder Misserfolge im Spiel kennzeichnen und diese akustisch untermauern. Die Lautstärke kann hier als wirkungsvolles Gestaltungsmittel eingesetzt werden. Dabei hat der Sound neben der Feedback- auch eine Unterhaltungsfunktion:

> „Guttural satisfaction cannot be discounted either. Sound effects are also designed to entertain and wow the player. Hollywood goes with bigger than life sounds, and games can too. After spending five minutes in a battle with the big bad guy, you want to hear something to make the effort worthwhile. This little sonic reward can bring a smile to the player's face and keep themcoming back for more" (Marks, 2001: 291).

3.4 Sound in Computerspielen: Ein kurzer historischer Abriss

Wie in Abschnitt 2.2 erläutert, dient das Medium Computerspiel zur Unterhaltung. Das Ziel des Computerspielens besteht darin, dem Spieler ein möglichst realistisches Präsenzerleben zu bieten, das heißt, den Spieler in eine virtuelle Welt eintauchen zu lassen und dies auf eine interaktive Art und Weise (Wünsch & Jenderek, 2009). Wie zuvor beschrieben, spielt der Sound hierbei eine wesentliche Rolle, da der Spieler nur dann in die virtuelle Welt eintauchen kann, wenn auch seine akustischen Sinnesreize angesprochen werden (ebd.). Hierfür muss der Sound eines Spiels möglichst authentisch wirken, wie zum Beispiel durch passende Hintergrundgeräusche. Darüber hinaus ist der Sound im Computerspiel dafür zuständig, den Inhalt bzw. die Dramaturgie der Handlung zu unterstützen und Emotionen zu vermitteln (Raffaseder, 2010; Lensing, 2009; Möller, 2007). Daneben dient er als Feedbackmöglichkeit für den Spieler (Grunwald, 2008) und kann somit dessen Spielerleben intensivieren (vgl. *3.3*).

Schon zu Beginn der Spieleentwicklung spielte die Gestaltung des Tons eine wesentliche Rolle. Bereits das im Jahr 1971 erschienene Spiel „*Computer Space"* verwendete „[…] klangerzeugende Schaltungen, um die verschiedenen auf dem Bildschirm dargestellten Handlungen (Raketenantrieb, Schussgeräusche, Explosionen) für den Spieler akustisch erfahrbar zu machen" (Leenders, 2012: 23). Eines der prägnantesten Sounds der frühen Spielentwicklung enthielt das ein Jahr später erschienene und populäre Spiel *„Pong"* von Atari (ebd.). Leenders (2012) beschreibt die Rolle des Sounds in der frühen Spieleentwicklung folgendermaßen:

> „Diese Art des «Sounddesigns» ist in der Frühzeit der Videospiele eher Regel als Ausnahme, da der kleine Stab der Programmierer sowohl für Konzept und Programmierung als auch für die technische Umsetzung verantwortlich ist. Der Ton der Spiele entsteht zu dieser Zeit häufig direkt durch die Kombination von Bauteilen auf der Platine oder durch Programmierung der Welle über Binärcode. Der sehr begrenzte ROM-Speicherplatz im einstelligen Kilobyte-Bereich führt zusätzlich zu Einschränkungen bei den Möglichkeiten der Tongestaltung. Trotzdem sind auch in dieser frühen Zeit die Soundkapazitäten der Videospielgeräte ein zentrales Verkaufsargument“ (Leenders, 2012: 23ff).

Aufgrund von technischem Fortschritt kam es auch zur Weiterentwicklung des Sound Designs in den achtziger Jahren. Die Hersteller verwendeten in ihren Geräten erstmalig eigene Chips speziell für den Spiele-Sound (sogenannte „programmable sound generators“ oder „PSGs“). Die technischen Veränderungen beschreibt Leenders (2012) wie folgt:

> „Die populärsten Chips dieser Zeit verwendetet […] 3 Frequenzgeneratoren, die Rechteck-, Sinus- oder Sägezahnspannungen erzeugen, und einen Rauschgenerator für Weißes Rauschen, das häufig für perkussive Musikelemente eingesetzt wird. Simultan beginnt man, mit Samples zu arbeiten, die aufgrund der damals geringen Größe des ROM-Speichers und der Datenbusse zunächst nur eine sehr niedrige Bit-Rate von maximal 8 Bit aufweisen und stets von stark begrenzter Länge sind. *Donkey Kong* (1981) und *Missile Command* (1980) sind zwei berühmte Vertreter dieser Umbruchszeit“ (Leenders, 2012: 24).

Die neuen technischen Möglichkeiten veränderten die Branche und führten zu neuen Möglichkeiten in der kreativ-ästhetischen Sound-Entwicklung. Darüber hinaus führte auch die Einführung von Videospielkonsolen zu Veränderungen des Sound Designs und zu einem „[…] Umdenken in design- und klangästhetischer Hinsicht“ (ebd.). Zuvor wurden Computerspiele hauptsächlich für Spielhallen produziert, wodurch der Sound auf eine kurzfristige Spielzeit und einen sich wiederholenden Spielablauf ausgelegt war. Auch aufgrund der starken Lautstärke in Spielhallen wurde der Ton so hergestellt, dass er sich durch ein sehr lautes und aggressives Sound Design auszeichnete. Durch die Einführung von Spielkonsolen änderte sich auch der Ton und die neuen Herausforderungen lagen nun in der dynamischen Tongestaltung[12]. Die neuen Spiele, wie beispielsweise *„The Legend of Zelda (1986)“* und *„Final Fantasy (1987)“*, zeichneten sich, im Gegensatz zu den für die Spielhallen konzipierten Spiele, durch eine deutlich längere Spielzeit und einen kontinuierlichen Spielablauf aus (Leenders, 2012: 25). „Längere

[12] Vgl. dynamischer Ton in Abschnitt 3.5.3.

Musik-Cues[13], dynamische Musikgestaltung […], geringerer Themenbezug in der Musik und abwechslungsreicheres Sounddesign“ (ebd.) waren die Folge.

Die Veränderungen des Markts brachten Mitte der achtziger Jahre noch weitere Herausforderungen für die Tongestaltung mit sich. Der PC wurde als Massenware eingeführt und auch der Konsolenmarkt wuchs weiter an. Die Spiele sollten nun sowohl für die Spielhallen als auch für Konsolen und den PC produziert werden. Leenders (2012) beschreibt die dadurch entstandene Problematik wie folgt:

> „Die verschiedenen Systeme haben allerdings zum Teil vollkommen unterschiedliche Soundchips, die teilweise auch eine unterschiedliche Anzahl an verwendbaren Stimmen (Kanälen) aufweisen. So gibt es häufig Versionen eines Spiels, die auf wichtige Teile der Musik oder eine ganze Soundebene verzichten müssen. Diese Problematik wird noch verstärkt, als Mitte der achtziger Jahre die Entwicklung von Soundkarten den Ton auf dem PC revolutioniert. Soundkarten eröffnen damals durch die Implementierung von FM –Synthese und MIDI völlig neue Gestaltungsmöglichkeiten für den Videospielton. Gleichzeitig tauchen jedoch neue Probleme auf. Dadurch, dass sich nun jeder Besitzer eines PCs eine andere Soundkarte einbauen kann, ist es für den Tongestalter nur noch sehr schwer möglich, vorherzusagen, wie das fertige Spiel auf den unterschiedlichen PCs klingen wird“ (Leenders, 2012: 26).

In den neunziger Jahren kam es Leenders (2012) zufolge erneut zu einem technischen Meilenstein. Durch die Einführung der allgemeinen Standards wie General MIDI und durch das Aufkommen der 16 Bit-Konsolen kam es zu einer erweiterten Datenkapazität, die sich auch auf die Ton-und Musikgestaltung auswirkte. Auch die Einführung der Audio CD im Jahr 1990 beeinflusste den Sound und das Sound Design von Computerspielen, da diese die Hörgewohnheiten der Konsumenten änderte und neue Standards für einen qualitativ höherwertigen Klang im Heimbereich schuf (ebd.). Darüber hinaus veränderten auch neue Kompressionsverfahren und Datenträger die Soundentwicklung:

> „Im PC-Markt, der aufgrund der Wandlungsfähigkeit seiner Hardwarestruktur immer schneller auf aktuelle Entwicklungen reagieren kann, nimmt in der Folge die Geschwindigkeit der technischen Neuerungen rapide zu, und immer mehr Grenzen für die Tongestaltung bei Videospielen fallen weg. Die Einführung von Kompressionsverfahren wie MP3 verringert die engen Beschränkungen des verfügbaren Speicherplatzes ebenso wie die Einführung der DVD-ROM als Datenträger. Die Möglichkeit der Installation auf der Festplatte des Rechners ermöglicht zusätzlich wieder dynamischere Toneinsätze und eine große Zahl simultaner Tonebenen“ (Leenders, 2012: 28).

Insgesamt wirkt sich der technische Fortschritt in Form von Surround Sound im Heimbereich, das Pro Logic II-Format sowie immer größere Datenträger und Einsparungen im Speicherplatz auch auf die Entwicklung des Sounds und das gesamte Sound Design

[13] Werden in der CryENGINE Music Mood genannt vgl. 5.9.1.

in Computerspielen aus. Leenders (2012) zufolge ist es aufgrund der heutigen technischen Errungenschaften nun möglich, „[…] ohne qualitative Einschränkungen dynamische Tonarbeit[14] zu leisten“ (Leenders, 2012: 29).

Betrachtet man den geschichtlichen Abriss des Tons in Computerspielen von Leenders (2012), so stellt man fest, dass zu Beginn der Ton aufgrund von technischen Beschränkungen hauptsächlich darauf ausgelegt war, dem Spieler eine simple Feedbackmöglichkeit anzubieten (vgl. hierzu das Pong-Geräusch). Erst später kam es zu einem differenzierterem Sound Design aufgrund von technischem Fortschritt. Auch das Arbeitsfeld des Sound Designers hat sich erst hierdurch etabliert (vgl. *3.2.1*) und das Sound Design ist inzwischen zu einer komplexen, technisch-kreativen Aufgabe im Rahmen der Spieleproduktion geworden.

3.5 Besonderheiten des Sounds in Computerspielen: Interaktivität und Nonlinearität

Die Besonderheit der Interaktivität des Mediums Computerspiel wurde bereits in den Teilabschnitten 2.1 und 2.2 kurz aufgegriffen. Interaktivität bedeutet, dass im Gegensatz zum Film der Rezipient selbst aktiv agieren und verschiedene Aufgaben lösen bzw. bestimmte Entscheidungen fällen muss (Günzel, 2012). Leenders (2012) beschreibt dies so:

> „Der Spieler entscheidet nicht nur über die Handlungen seiner Spielfigur in einer vorgegebenen Spielumgebung, sondern bestimmt selbst über die Zeitgestaltung des Spieles und damit auch über die Art und Weise, wie das Spiel letztlich rezipiert wird; er gestaltet sein eigenes Erleben mit. Mit seinen Eingaben bestimmt er, wie schnell das Spiel abläuft, wann eine Spielumgebung auf die andere folgt, wie oft Unterbrechungen auftreten, wie viel Zeit mit welcher Aktion verbracht wird, und nimmt damit direkten Einfluss auf die audiovisuelle Gestaltung“ (Leenders, 2012: 47).

Diese Interaktivität muss auch bei der Umsetzung des Sound Designs berücksichtigt werden und stellt den Sound Designer vor großen Herausforderungen, denn die Aktionen des Spielers sind nicht vorhersehbar und können nie so genau geplant werden, wie dies beispielsweise beim Sound Design im Film möglich ist. Bei der Vertonung von Computerspielen muss somit nicht nur eine bekannte Handlung, sondern viele Handlungsvariablen berücksichtigt werden (Clark, 2007). Die Vielzahl akustischer Entschei-

[14] Dynamischer Ton dient als Oberbegriff, um die interaktive und adaptive Tonarbeit im Bereich der Videospiele zusammenzufassen. Vgl. Abschnitt 3.5.3.

dungsmöglichkeiten, die der Sound Designer vorhersehen muss, bedarf dabei eines durchdachten Managements[15] (Collin, 2008; Leenders, 2012).

Collins (2008) beschreibt diesen Aspekt als sogenannte „Nonlinearität“ und erklärt dies in ihrem Buch „*Game Sound*“ so:

> „The most significant problem facing game composers is the nonlinear basis of games in general. Put simply, games are largely unpredictable in terms of the directions the player may take, and the timings involved. Many game narratives progress in a ‘‘branching’’ manner, similar in shape to the branches of a tree, in which there are many possible paths and endings“ (Collins, 2008: 142).

Durch die Interaktivität des Mediums kommt es zu einer nicht-linearen Struktur des Spiels. Leenders (2012) erläutert dies in Anlehnung an Collins (2008) Ausführungen:

> „Nonlinearität ist neben der Interaktivität das wichtigste Merkmal, das Videospiele von Film und Fernsehen und besonders Animationsfilmen unterscheidet. Die Spielerfahrung eines Videospiels ist schon vom Prinzip her nicht linear, da der Eingriff des Konsumenten in die Gestaltung des medialen Inhalts schon die Linearität verletzt. Selbst der einfachste denkbare Eingriff des Spielers, die Entscheidung zwischen zwei definierten Möglichkeiten, durchbricht die Linearität, da es nun schon zwei mögliche Varianten gibt, wie sich das Geschehen auf dem Bildschirm entwickelt“ (Leenders, 2012: 39).

3.5.1 Wiederkehrende Sounds

Dabei wirkt sich die Nonlinearität auf verschiedene Bereiche der Musik– und Tongestaltung aus, wie zum Beispiel auf die wiederkehrenden Sounds. Im Computerspiel lässt es sich nur schwer vermeiden, dass ein Spieler einzelne Sounds oder musikalische Abschnitte häufig und möglicherweise auch direkt nacheinander hören wird. Dazu zählen unter anderem die Sounds, die für die Menüauswahl verwendet werden oder aber auch beispielsweise die Schrittgeräusche der Spielfigur. Auch die Hintergrundmusik in den einzelnen Umgebungen des Spiels wiederholt sich. Hierbei entsteht die Problematik, dass es unter Umständen zu einer „Abnutzung des Sounds“ (Leenders, 2012: 40) kommt. Die sich wiederholende Musik kann dazu führen, dass sich der Spieler an den Sounds stört und sogar das Spiel abbricht (ebd.). Anhand eines Beispiels bringt Leenders (2012) diese Problematik auf den Punkt:

> „In einem Spiel der *Final Fantasy*-Reihe hört man den Sound für die Bewegung des Cursors im Menü sowie den Bestätigungssound beim Auswählen eines Menüeintrages

[15] Auf das Managementsystem und der sogenannten „Branching tree-Struktur“ der Handlungsorte soll hier nicht näher eingegangen werden. Für nähere Informationen vgl. hierzu Kapitel 8 in Collins (2008) und Abschnitt 3.5.1 in Leenders (2012).

> zusammen circa 800 Mal innerhalb einer Spielstunde. Die Schrittsounds der Spielfigur kann man […] noch deutlich öfter hören, da circa 2 pro Sekunde erzeugt werden, wenn man sich kontinuierlich fortbewegt. Je nachdem, wie lange man sich in ein und demselben Gebiet aufhält, kann man sich ausrechnen, wie oft man den zum Beispiel 45-sekündigen Loop der Hintergrundmusik hört. Dass es bei einer so großen Zahl der Sound- und Musikwiederholungen zu Abnutzungserscheinungen kommen kann, ist ersichtlich“ (Leenders, 2012: 40ff.)

Je nach Spielkontext ergeben sich unterschiedliche Lösungsstrategien für den Sound Designer. Durch den technischen Fortschritt und den erweiterten Speicherplatz (vgl. *3.4*) ist es inzwischen möglich, die sich abnutzenden Sounds zu umzugehen, und zwar durch die Schaffung von Varianz. Dies kann am Beispiel der Schrittgeräusche verdeutlicht werden: Mittlerweile können Schrittgeräusche so an die Realität angepasst und variiert werden, dass sich hierfür nicht nur ein Sound ständig wiederholen muss. Beispielsweise werden Schrittgeräusche an die Laufgeschwindigkeit des Spielers und an den Untergrund angepasst, wodurch sich eine Vielzahl von verschiedenen Kombinationsmöglichkeiten ergibt und der Sound sich nicht so schnell abnutzt (ebd.). Die praktische Umsetzung hierfür wird in Abschnitt 5.7.1 gezeigt.

Neben der Schaffung von Varianz ist eine weitere Lösungsstrategie des Sound Designers die Erzeugung von möglichst stimmigen und authentischen Sounds. Leenders (2012) stellt dabei fest, dass insbesondere wiederholenden Sounds möglichst „unauffällig“ und „passend“ gestaltet werden müssen, da es sonst schneller zur Abnutzung kommen kann. Beispielsweise können Schussgeräuschen dem Sound Designer Charles Maynes zufolge bei einer Übertreibung des Sounds, wie es häufig in Hollywoodfilmen der Fall ist, schnell ermüdend auf den Spieler wirken (ebd.).

Zusammenfassend beschreibt Leenders (2012) die Herausforderung bei sich wiederholenden Sounds für die Gestaltung des Tons in Computerspielen folgendermaßen:

> „Um einen Sound wiederholbar zu produzieren bedarf es viel Erfahrung und dauerhafter Tests, da es keine wirklichen Rezepte gibt, nach denen man vorgehen kann. Es gibt bestimmte Spektren und Frequenzbilder, die für das menschliche Ohr störender oder auffälliger sind. Diese zu beachten, ist sicherlich vorteilhaft bei der Produktion von Sounds, die oft wiederholt werden müssen“ (Leenders, 2012: 42)

3.5.2 Loops

Ein weiteres Beispiel sind sogenannte Loops (vgl. hierzu Collins, 2008), die dort verwendet werden, wo nicht vorhersehbar ist, wie lange sich ein Spieler in einem bestimmten Spielbereich aufhalten wird. Sie stehen im Zusammenhang mit dem Aspekt der

Nonlinearität und entstanden Mitte der achtziger Jahre, um trotz der Speicherplatzbegrenzungen eine „[…] kontinuierliche Musikuntermalung zu gewährleisten […]“ (Leenders, 2012: 42).

In Computerspielen werden Loops hauptsächlich zur akustischen Gestaltung von Hintergrundmusik, Menüs und Atmosphären eingesetzt. Mit Loops können Spielabschnitte in beliebiger Länge akustisch untermalt werden. Dies geschieht folgendermaßen:

> „Hierzu müssen der Komponist und der Sounddesigner im simpelsten Fall beim Schreiben und bei der Produktion der Musik und der Atmo Ein- und Ausstiegspunkte definieren, zwischen denen die Musik oder Atmo in dauerhafter Wiederholung laufen kann“ (ebd.).

Dadurch kann es jedoch wieder zur Abnutzung des Sounds kommen. Auch für diese Problematik existieren verschiedene Techniken[16] des Sound Designs, damit der Konsument die Loops nicht erkennt und sich nicht an einer akustischen Wiederholung stört (ebd.).

Eine einfache Möglichkeit wäre beispielsweise, ein sich wiederkehrendes Musikstück zu verlängern, so dass die gesamte Anzahl der nötigen Wiederholungen innerhalb eines Spielabschnittes vermindert werden kann. Dabei sollte die Umsetzung so erfolgen:

> „Um dies zu gewährleisten, bietet es sich zusätzlich an, Musik, die unter Umständen über einen sehr langen Zeitraum geloopt werden muss, nicht mit stark wiedererkennbaren Elementen wie simplen Melodien oder eindeutigen Einsätzen von Instrumenten und Sounds zu versehen, da auch so die Gefahr der Abnutzung verringert werden kann. Gleiches gilt für die Verwendung eindeutig wiedererkennbarer Sounds wie Schreien von Tieren in der At mo“ (Leenders, 2012: 43).

Darüber hinaus kann die Gestaltung einer Atmo so erfolgen, dass diese in verschiedene Anteile aufgegliedert wird. Dabei wird ein Grundsound der Atmo möglichst ohne wiedererkennbare Elemente per Loop eingefügt und mit weiteren Einzelsounds (One-Shots), die zeitlich und räumlich per Zufall innerhalb des Bereichs der Atmo platziert werden, ergänzt (ebd.). Ein Beispiel zur praktischen Umsetzung ist in Abschnitt 5.2 zu finden.

Insgesamt ist der Umgang mit Loops ein umfangreicher und wichtiger Teil bei der Arbeit des Sound Designers (ebd.).

---

[16] Weiterführende Informationen zu den verschiedenen Techniken liefert das Kapitel 4.3 in Leenders (2012).

3.5.3 Dynamischer Ton

Der Begriff „dynamischer Ton“ fällt häufig im Zusammenhang mit den Besonderheiten des Sound Designs von Computerspielen - insbesondere in Verbindung mit der Interaktivität und Nonlinearität des Sounds (Leenders, 2012). Synonym tauchen für den dynamischen Ton auch häufig die Begriffe „interaktiver Ton“, „adaptiver Ton“ oder „nichtlinearer Ton“ in der gängigen Fachliteratur auf (Marks & Novak, 2009).

In dieser Arbeit wird der Begriff „dynamischer Ton“ gebraucht und dabei in Anlehnung an Collins (2008) und Leenders (2012) als Oberbegriff verstanden, der die *interaktive und adaptive* Tonarbeit in Computerspiele beschreibt. Der Ton ist einerseits deshalb als interaktiv zu bezeichnen, da er auf die direkten Eingaben des Spielers reagiert. Ein Beispiel hierfür wären die Schussgeräusche einer Waffe, die der Spieler betätigt. Der adaptive Ton hingegen reagiert auf unterschiedliche Zustände und Parameter innerhalb einer Spielumgebung. Ein Beispiel hierfür sind Änderungen in der Hintergrundmusik oder die Beschleunigung der Musik bei der Annäherung an einen Gegner (Leenders, 2012). Da es sich beim Computerspiel um ein nicht-lineares und interaktives Medium handelt, muss der Ton sich generell immer auf die Eingaben des Spielers und an die Variablen der Umgebung anpassen (ebd.). Zur praktischen Umsetzung vgl. Abschnitt 5.9.

Aus diesem Grund könnte jeder Computerspielton als dynamisch bezeichnet werden. Allerdings wird der Begriff „dynamischer Ton“ im Sound Design inzwischen lediglich als Abgrenzung verwendet, „[…] um diejenigen Spiele hervorzuheben, in denen der Ton die dynamische Struktur des jeweiligen Spieles und des Mediums aufnimmt und stark in seine Konzipierung und Umsetzung einbindet. Ein in sich linearer orchestraler Score, dessen verschiedene Musik-Cues allein nach Gebieten ausgelöst werden, in denen sich die Spielfigur befindet, wäre unter diesem Gesichtspunkt sehr statisch, eine Hintergrundmusik, die je nach Gesundheitszustand und Anzahl der Gegner in der Nähe der Spielfigur durch Änderungen ihres Arrangements und der Struktur den Grad der Spannung, die auf den Spieler einwirkt, anpasst, wäre dynamisch“ (Leenders, 2012: 55).

Insgesamt ist dabei festzuhalten, dass mit einer Weiterentwicklung von technischen Möglichkeiten es auch zu einer ständigen Weiterentwicklung und Verbesserung der Ton- und Musikgestaltung kommt. Alles in allem müssen bei der Umsetzung eines dynamischen Tons sowohl die technischen Voraussetzung als auch die genrespezifischen Besonderheiten beachtet werden:

„Um ein dynamisches Tonkonzept umzusetzen, müssen Sounddesigner und Komponisten neben einiger Erfahrung mit Videospielen vor allem auch Beweglichkeit im Denken und einen hohen Grad an Kreativität in Bezug auf die Umsetzung und Einbindung der eigenen Ton- und Musikarbeit in das Spiel mitbringen. Denn jedes Spiel erfordert durch seine einzigartige Struktur, aber auch bezogen auf sein Genre […] und die technischen Gegebenheiten, wie zum Beispiel die verwendete Audioengine, eine ganz eigene Herangehensweise an dynamischen Ton“ (Leenders, 2012: 57).

4 Die CryENGINE 3

Die CryENGINE 3 ist die dritte Version der CryENGINE entwickelt von der Firma Crytek und ist die Basis mehrerer AAA[17] Spiele, wie Crysis 2, Crysis 3 oder Sniper Ghost Warrior 2. Das CryENGINE 3 SDK[18] kann für eine nicht-kommerzielle Nutzung kostenlos heruntergeladen werden.

Das vorliegende Kapitel beschäftigt sich zunächst mit dem CryENGINE-Entwickler Crytek und dessen Firmengeschichte. Anschließend werden die Audiofeatures der Engine und Audio-Engine FMOD vorgestellt, die zum Verständnis des 5. Kapitels und somit für die Umsetzung des Sound Designs mit Hilfe der CryENGINE 3 zwingend erforderlich sind.

4.1 Die Geschichte von Crytek

Im Jahr 1997 gründeten die drei Brüder Faruk, Avni und Cevat Yerli die Firma Crytek in Coburg. Bereits drei Jahre nach der Gründung sorgten sie mit ihrer Tech-Demo *X-Isle* auf der ECTS[19] erstmals für Aufsehen. Vier Jahre später veröffentlichten sie ihr erstes Spiel mit dem Titel *Far Cry*, das mit Hilfe der ersten Version der selbst entwickelten Game-Engine namens *CryENGINE* umgesetzt wurde. Das Erstlingswerk *Far Cry* wurde ein Welterfolg und über 2,6 Millionen Mal verkauft. Nach dem Erfolg begann Crytek mit der Entwicklung des FPS *Crysis* und arbeitete an der zweiten Version der CryENGINE. Im Jahr 2006 zog das Unternehmen von Coburg nach Frankfurt am Main und gründete zusätzlich in Kiew ein Tochterunternehmen. Ein Jahr später folgte ein weiteres Studio in Budapest. Schließlich wurde noch im gleichen Jahr *Crysis* auf Basis der „CryENGINE 2“ veröffentlicht.

In den folgenden Jahren expandierte Crytek weiter und übernahm 2008 den bulgarischen Entwickler *Black Sea Studios*. Zudem gründeten sie in diesem Jahr ein weiteres

[17] „Als AAA-Spiel oder Triple A-Spiel bezeichnet man Spiele für PC und Konsole dann, wenn der Aufwand für die Herstellung ungewöhnlich hoch ist und auch die Verkaufszahlen im Vergleich zu vielen anderen Spielen überragend ausfallen. Zudem ist die Qualität der Spiele ausgezeichnet und solche Spiele fallen auch durch eine riesige Fangemeinde auf.“ (*www.wissenswertes.at/index.php?id=spiele-aaa*)

[18] Software Development Kit.

[19] Die ECTS (European Computer Trade Show) war eine Computerspiel-Fachmesse in London.

Studio in Seoul und übernahmen bereits ein Jahr später die britische Spieleschmiede *Free Radical Design*.

Basierend auf der aktuellen CryENGINGE 3 wurde im Jahr 2011 *Crysis 2* veröffentlicht, das ein Jahr später mit dem Deutschen Computerspielpreis als bestes deutsches Spiel ausgezeichnet wurde:

> „Mit Cryis 2 haben erstmalig Entwickler aus Deutschland technologisch, qualitativ und ökonomisch weltweit Publikum und Fachwelt überzeugt und begeistert. Eine eigene Technologie, die weltweit auch im Bereich Serious Games eingesetzt wird, eine mehr als Hollywood-Reife Präsentation, eine grafische, akustische und spielerische Qualität auf höchstem Niveau überzeugten die anwesenden Jurymitglieder“ (*www.deutscher-computerspielpreis.de/3.0.html*).

Im Januar 2013 wurde ein weiteres Studio in den USA gegründet. Inzwischen beschäftigt Crytek insgesamt 775 Mitarbeiter weltweit, wobei in etwa die Hälfte der Beschäftigten in Deutschland tätig sind (vgl. hierzu *www.crytek.com/company*).

4.2 Das Sound Event System der CryENGINE 3

Das Sound System unterstützt 7.1 Surround Sound und gewährleistet durch die Integration der *FMOD Ex*® Sound Bibliothek auch die Kompatibilität mit Spielekonsolen, wie Xbox 360 und PlayStation3.

Das Sound System der CryENGINE 3 kann aus zwei Perspektiven betrachtet werden: Zum einen aus der Sicht des Sound Designers; dieser erstellt und organisiert Sound Events mit dem FMOD-Designer. Zum anderen aus der Sicht des Programmierers, der die Sounds mittels Sandbox Editor oder per Code in das Spiel einpflegt.

4.2.1 FMOD-Designer

Mit dem FMOD Designer 2010, der beim CryENGINE 3 SDK mitgeliefert wird, ist es möglich komplexe Audio Events zu erstellen, die mit der Engine wiedergegeben werden können. Der FMOD-Designer ermöglicht dem Sound Designer Sound Events ohne Programmierkenntnisse zu erstellen und dessen Eigenschaften bzw. Verhalten von der Engine aus zu steuern. Die Sound Events können aus mehreren Ebenen bestehen, bestimmte Parameter besitzen, überblendet, sowie mit DSP Effekten versehen werden und ein zufälliges Verhalten besitzen (vgl. hierzu FMOD Designer 2010 User Manual 2.0, S.11).

Im Folgenden werden kurz einige wichtige Elemente des „FMOD-Designer 2010“ erklärt.

4.2.1.1 FMOD-Designer Projekt

Das Projekt ist eine Sammlung von Sound Events (vgl. *4.2.1.2*). Obwohl es möglich ist alle Sound Events eines Spiels in einem Projekt zu erstellen, ist es empfehlenswert mehrere kleine Projekte zu verwenden. Nicht nur für die Übersichtlichkeit ist es sinnvoll für jede Waffe, jedes Fahrzeug etc. ein eigenes Projekt zu erstellen, sondern es ermöglicht auch das parallele Arbeiten mehrerer Sound Designer an unterschiedlichen Projekten.

4.2.1.2 Sound Event

Ein Sound Event besteht aus einem oder mehreren Sound Definitionen (vgl. *4.2.1.3*) und Ebenen. Dem Sound Event können Eigenschaften zugeteilt werden, wie zum Beispiel ob er als Loop oder als One-Shot[20], sowie im 2D oder 3D Modus abgespielt wird, wie der Sound über die Entfernung gedämpft wird und viele Weitere. In einem Sound Event kann auch bestimmt werden, wie dieser auf die von der Engine übergebenen Parameter reagiert. Beispielsweise übermittelt der Parameter *daylight* die Tageszeit des Spiels an das Sound Event. Jede Ebene eines Sound Events kann durch übermittelte Werte der Parameter manipuliert werden. Ein Beispiel hierfür wäre, dass mit zunehmender Tageszeit auch ein bestimmter Sound ausgeblendet wird (vgl. *5.2*).

4.2.1.3 Sound Definition (kurz: Sound Def)

Eine Sound Definition ist ein Kontainer, der einen oder mehrere Sound Dateien enthält. Bei jedem Abspielen des Sound Def wird dabei eine andere enthaltene Sound-Datei per Zufall abgespielt. Jede enthaltene Sound-Datei kann eine eigene Wahrscheinlichkeit für das Abspielen besitzen.

4.2.1.4 Event Gruppen (event group)

Ähnliche Sound Events können in sogenannte Event Gruppen gesammelt werden. Beispielsweise sollten alle Sounds einer Waffe in einer Event Gruppe zusammengefasst werden. Gruppen können vom Programmierer bei Bedarf vorgeladen und, falls sie nicht mehr gebraucht werden, entladen werden.

[20] Einmaliges Abspielen.

4.2.1.5 Kategorien

Jedes Sound Event kann in genau eine Kategorie eingeteilt werden. Dabei sollten Sounds gleicher Art, wie zum Beispiel alle Atmo-Sounds, in derselben Kategorie (z.B. *environment*) abgelegt werden. Dadurch ist es möglich, alle Sounds einer Kategorie, auch wenn sie in unterschiedlichen Projekten erstellt wurden, gemeinsam über die *Sound Mood* Funktion zu steuern. Eine sinnvolle Kategorie-Einteilung der Sound Events ist für den späteren Audio-Mix unerlässlich. In Abschnitt *5.3* wird nochmals näher darauf eingegangen.

4.2.2 Integrierte Audio Spezial-Effekte

Um den Realitätsgrad der auditiven Wahrnehmung eines Spielers zu erhöhen, können mehrere Audio Spezial Effekte innerhalb der Engine verwendet werden. Dazu zählen **Obstruction**, **Pitch**, der **Doppler-Effekt** und **HDR-Audio**, die nachstehend kurz erklärt werden:

4.2.2.1 Obstruction (auf Deutsch: Hindernis)

Mit der *Obstruction*-Funktion wird jedem Sound, der aufgrund eines Hindernisses – beispielsweise aufgrund einer Wand - nicht direkt auf den Spieler trifft, ein Tief-Pass Filter hinzugefügt. Für weniger Leistungsstarke PCs kann der Effekt deaktiviert bzw. auf eine Absenkung der Lautstärke reduziert werden.

4.2.2.2 Pitch

Pitchen bedeutet im Deutschen eine Tonhöhenänderung. Sounds können in Echtzeit in ihrer Tonhöhe nach oben oder unten gepitcht werden. Die jeweilige Stärke des Pitch kann über Sound Moods definiert werden (vgl. *5.3*).

4.2.2.3 Doppler-Effekt

Bewegt sich der Spieler auf eine Audioquelle zu oder von ihr weg, wird eine Frequenzänderung des hörbaren Audiosignals hervorgerufen. Dieser sogenannte Doppler Effekt wird von der Engine automatisch für jeden Sound berechnet. Die Stärke des Effekts kann im FMOD Designer für jeden Event bestimmt werden.

4.2.2.4 HDR Audio

Mit Hilfe des HDR-Audio passt die Engine leisere Soundeffekte so an das Spielgeschehen an, dass sehr laute Sounds, wie beispielsweise Schüsse oder plötzlich auftretende Explosionen, besser zur Geltung kommen.

4.3 Der Sandbox Editor

Der Sandbox Editor bietet die Möglichkeit in Echtzeit Änderungen am Spiel vorzunehmen und diese direkt im Editor zu testen. Crytek nennt dieses System *"What You See Is What You Play"* (WYSIWYP).

Über die Menüleiste kann auf eine Vielzahl von Funktionen und Tools zugegriffen werden, wie beispielsweise die Database Ansicht oder der Charakter Editor. Mit Hilfe des Viewport können Objekte (*Entities*) in Echtzeit an die gewünschte Position innerhalb des Levels gesetzt werden. Die *RollupBar* bietet unter anderem Zugriff auf die *Entities* und Tools zum Bearbeiten des Terrains.

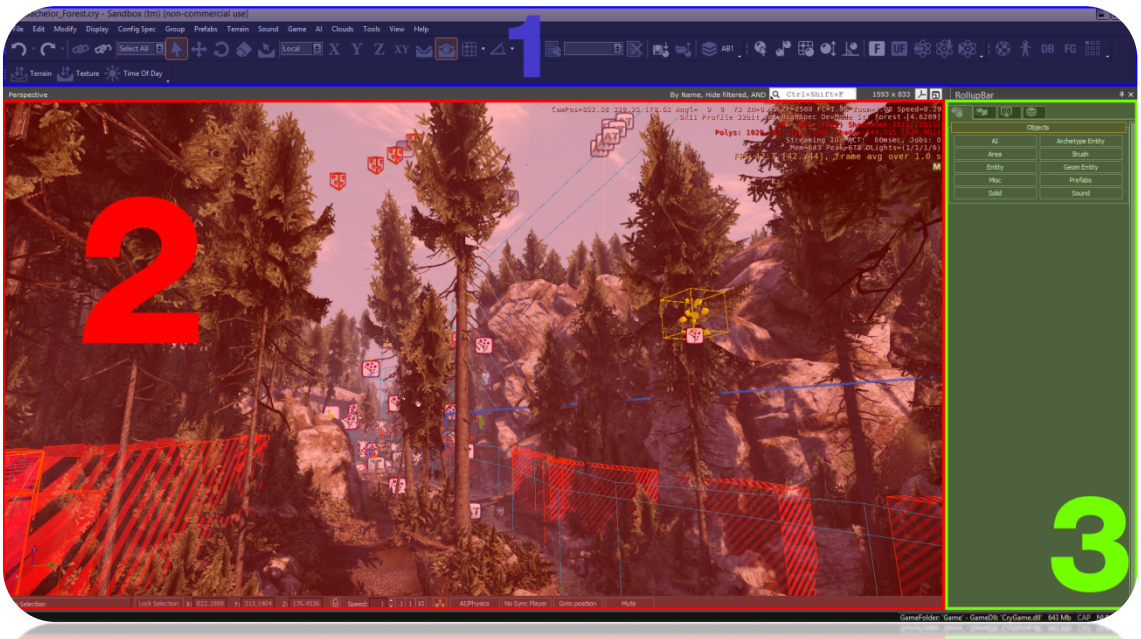


Abbildung 1: Menüleiste(1), Viewport(2) und RollupBar(3) des Sandbox Editors

4.4 Area Objects

Mit sogenannten *Area Objects* können im Level dreidimensionale Zonen erstellt werden, mit denen Ereignisse ausgelöst (getriggert) oder *Sound Entities* (vgl. *4.5*) verknüpft werden können. Es gibt verschiedene *Area Objects* Varianten. Dazu zählen: *AreaBox*

(quaderförmig), *AreaSphere* (kugelförmig) und *AreaShape* (eigene Form), die sich nur in ihrer Form unterscheiden. Es existieren noch weitere *Area Objects,* wie etwa für Beleuchtung und Nebel Effekte, die in dieser Arbeit jedoch keine Rolle spielen. Auf die *Area Objects* kann über die *RollupBar* zugegriffen werden.

4.5 Entities

Entities sind Objekte, mit denen der Spieler interagieren kann. Diese Arbeit beschäftigt sich ausschließlich mit *Entities*, die für den Sound Bereich verwendet werden. Die *Sound Entities* können sowohl im Level platziert als auch per *Flow Graph* integriert werden. Im Folgenden werden die *Sound Entities* der CryENGINE 3 erläutert.

4.5.1 AmbientVolume

Ein *AmbientVolume Entity* muss mit einer *Area* verknüpft werden. Dadurch entsteht innerhalb dieser *Area* ein Sound mit volumetrischen Verhalten. Die *Area* wird quasi mit dem Sound ausgefüllt. Das heißt, wenn der Spieler sich innerhalb dieser *Area* befindet ist er vom Sound umgeben. Bewegt er sich dagegen aus der *Area* heraus oder in die *Area* hinein wird der Sound aus- bzw. eingeblendet und kommt aus der Richtung der *Area*. Somit eignen sich *AmbientVolume Entities* für das Einfügen von Atmo Sounds. Parameter, wie die Tageszeit (*daylight*) oder Kampfaktivität (*battle*), die an das Sound Event weitergeleitet werden, wodurch es auf die entsprechenden Spielaktionen reagieren kann (vgl. *5.2*).

4.5.2 MusicEndTheme

Das *MusicEndTheme* bestimmt, wie das aktuell laufende Music Thema beendet wird. Es wird innerhalb eines *Flow Graph* ausgelöst.

4.5.3 MusicLogicTrigger

Wird das *MusicLogicTrigger Entity* mit einer *Area* verknüpft, kann *MusicLogic* (vgl. *5.8.2*) beim Betreten dieser *Area* aktiviert und wiederum beim Verlassen deaktiviert werden.

4.5.4 MusicMoodSelector

Sofern einem Musik Thema ein neues *Mood*, das heißt, eine Stimmung innerhalb eines Musik Themas, zugwiesen werden soll, kann das *MusicMoodSelector Entity* mit einer *Area* verknüpft oder in einen *Flow Graph* integriert werden.

4.5.5 MusicPlayPattern

Mit diesem *Entity* kann ein Musik Pattern entweder zusätzlich zu einem derzeit abgespielten Pattern oder als ein eigenständiges Stück abgespielt werden. Das *MusicPlayPattern Entity* kann entweder über eine *Area* oder mittels *Flow Graph* ausgelöst werden.

4.5.6 MusicStinger

Das *MusicStinger Entity* spielt beim Auslösen eine kurze musikalische Phrase (*Stinger*) ab, die sich besonders zur Unterstützung eines Übergangs zweier Musik *Moods* eignet.

4.5.7 MusicThemeSelector

Mittels der *MusicThemeSelector Entity* kann ein Music Thema ausgewählt werden, indem das *Entity* mit einer *Area* verknüpft oder per *Flow Graph* getriggert wird.

4.5.8 RandomSoundVolume

Das *RandomSoundVolume Entity* muss mit einer *Area* verknüpft werden. Mit dieser *Entity* wird ein *One-Shot* Sound Event festgelegt, der an zufälliger Position in zufälligen Abständen innerhalb der *Area* abgespielt wird. Die *RandomSoundVolume* Sound Events können, wie die der *AmbientVolume*, auf Parameter reagieren.

4.5.9 ReverbVolume

Ist das *ReverbVolume Entity* mit einer *Area* verknüpft, werden alle Sounds innerhalb dieser *Area* mit einem in der *Enity* ausgewählten Reverb Effekt versehen. (vgl. *5.1*).

4.5.10 SoundEventSpot

Die *SoundEventSpot Entity* spielt einen Sound Event an der Stelle innerhalb eines Levels ab, an der der *SoundEventSpot* platziert ist.

4.5.11 SoundMoodVolume

Ist die *SoundMoodVolume Entity* mit einer *Area* verknüpft, wird beim Betreten ein *Sound Mood* aktiviert. Beim Verlassen wird er wiederum deaktiviert. Die *SoundMood-Volume Entity* kann auch über *Flow Graph* ausgelöst werden. (vgl. *5.3*)

4.5.12 SoundSpot

Die *SoundSpot Entity* hat die gleiche Funktion wie *die SoundEventSpot Entity*, mit einem Unterschied und zwar, dass die *SoundSpot Entity* neben Sound Events auch Wave Dateien abspielen kann. Die *SoundSpot Entity* eignet sich daher zum Testen innerhalb der Produktion. Aus Performancegründen sollten sie jedoch nicht im fertigen Spiel verwendet werden.

4.6 Flow Graph

Flow Graph ist ein visuelles Skript System, das es ermöglicht, Skripte ohne Programmiererfahrung visuell zu erstellen. Eine große Auswahl an *Nodes*[21] ermöglicht die Kontrolle über *Entities* und *KI*[22] innerhalb des Levels.

4.7 LUA Skripte und C++

Die CryENGINE verwendet LUA als Skriptsprache. Das KI System enthält beispielsweise Verhaltensweisen, die per Skript definiert sind. Wie zum Beispiel der Schaden, der eine Waffe anrichten kann oder die Geschwindigkeit, mit der ein Auto fährt, können ebenfalls per LUA Skript bestimmt werden. Außerdem ist es möglich mittels LUA, C++ Code aufzurufen.

[21] Nodes repräsentieren Entities oder auch mathematische Operationen innerhalb des Flow Graph und können miteinander interagieren.

[22] Künstliche Intelligenz.

5 Sound Design: Umsetzung am Beispiel der CryENGINE 3

Da die CryENGINE 3 sehr komplex ist, sollen die Praxisbeispiele so einfach wie möglich gestaltet werden. Oftmals existieren mehrere Möglichkeiten, die gezeigten Beispiele umzusetzen. Der folgende Abschnitt bietet daher lediglich eine Übersicht zur Implementierung der wichtigsten Soundelemente und Effekte. Das Ziel der Beispiele besteht jedoch nicht darin ein perfektes Sound Design zu präsentieren, vielmehr sollen die technische Umsetzung und die Verwendung der Engine im Vordergrund stehen. Das vorliegende Kapitel wurde mit Hilfe des Online-Handbuchs der Engine und anhand praktischer Übungen basierend auf Sean Tracys und Paul Reindells (2012) „CryENGINE 3 Game Development“ *Beginner's Guide* erstellt.

5.1 Raumklang

Wie in Abschnitt *2.2* gezeigt wurde, ist es für ein unterhaltsames Spiel-Erleben entscheidend, beim Spieler möglichst alle Sinnesreize anzusprechen. Aus diesem Grund muss der Klang der Spielgeräusche möglichst stimmig sein, um ein authentisches Spielgefühl zu schaffen. Daher sollte der gesamte Klang auch entsprechend an die Umgebung angepasst werden. Je nach Raumgröße und dessen Absorptionsoberflächen klingt ein Raum unterschiedlich. Beispielsweise klingen Geräusche in einem großen Wohnzimmer mit Teppichboden anders als in einem gefliesten Badezimmer.

Um mit der CryENGINE passende Räume zu simulieren, gibt es die *ReverbVolume Entity*, die *Areas* einen Halleffekt zuteilt. Dadurch werden alle Soundeffekte, die innerhalb dieser Bereiche abgespielt werden, mit dem entsprechenden Hall versehen. Somit muss jedes Soundfile nur einmal trocken[23] vorliegen. Der Hallanteil wird von der Engine automatisch berechnet. Die *ReverbVolume Entity* bietet dabei folgende Einstellungsmöglichkeiten:

[23] Trocken bedeutet in diesem Fall ohne Hallanteil.

Enabled	Aktiviert den Reverb Effekt.
Environment	Bestimmt, ob es sich um einen Außenbereich (0 – 1.4) oder Innenbereich (1.5 – 3) handelt. Dieser Parameter muss im FMOD Sound Event gesetzt sein. Zum Beispiel für den Nachhall eines Schusses im Freien.
FullEffectWhenInside	Deaktiviert das Überblenden angrenzender ReverbVolumes sobald der/die Spieler/in sich innerhalb der Area befindet.
OuterRadius	Definiert den Radius ab wann der Reverb Effekt außerhalb der Area eingeblendet wird.
ReverbPreset	Hier muss das gewünschte „Reverb Preset“ eingetragen werden.

Im Beispielvideo ***06*** wird der Übergang des Raumklangs beim Betreten einer Holzhütte gezeigt. Dazu wurde die Holzhütte im Level platziert und mit einer *Area Shape* versehen. Darüber hinaus wurde ein *ReverbVolumes Entity* erstellt, der mit der *Area Shape* verbunden und mit den Einstellungen wie in Abbildung 2 versehen wurde.

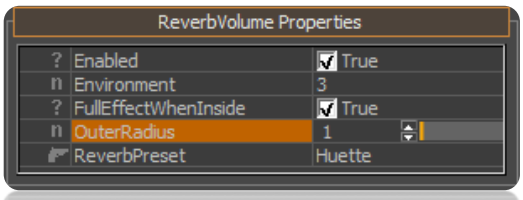


Abbildung 2:Einstellungen des ReverbVolume Entity

5.1.1 Erstellen eines Reverb Presets

Für den Fall, dass kein geeignetes Preset[24] vorhanden ist, können Eigene erstellt werden. Eine Übersicht der vorhandenen Presets findet sich unter dem Reiter *Reverb Presets*, in der *DataBase* Ansicht. Dort können auch neue, eigene Presets erstellt werden.

Nachfolgend werden die einzelnen Parameter der Reverb Preset erklärt, da es von großer Bedeutung ist, die einzelnen Parameter und deren Auswirkungen auf den Raumklang zu verstehen. Nur so kann ein realistischer Raumklang für eine Umgebung erzeugt werden.

1. Der ***Master-Level*** steuert die Lautstärke des Reverb Effekts.
2. ***Decay-Time*** bestimmt die Nachhallzeit in Millisekunden Diese kann mittels der Formel zur Berechnung der Nachhallzeit von Wallace C. Sabine (1868 - 1919) berechnet werden:
   „Die Nachhallzeit (RT) ist ein Maß für die Menge an Nachhall in einem Raum und gleich der Zeit für den Pegel eines stationären Tons der um 60 dB abfällt, nachdem der Ton abbricht. Die Abklinggeschwindigkeit hängt von der Größe der Schallabsorption in einem Raum, der Raumgeometrie und der Frequenz des Tons ab. RT wird in Sekunden angegeben“ (www.sengpielaudio.com/Rechner-RT60.htm).

$$T = \frac{k \cdot V}{A} \qquad A = \sum_{i=1}^{N} S_i \alpha_i + 4mV$$

[24] Voreinstellung.

Eine gute Hilfe hierbei bietet der Nachhallzeitrechner von www.sengpielaudio.com[25]. In den Nachhallzeitrechner werden die Werte für Raumgröße, Beschaffenheit der Wände, der Decke und des Bodens und die Größe von Störungsmaterial, wie beispielsweise Fenster oder Türen, eingetragen und anschließend erhält man die resultierende Nachhallzeit ***(Decay-Time)***.

3. ***HF-Decay-Ratio*** gibt das Verhältnis der Abklingzeit hoher Frequenzen in Relation zur Abklingzeit (***Decay Time***) an. Bei einem Wert von 1,0 ist die Abklingzeit frequenzübergreifend gleich. Übersteigt der ***HF-Decay-Ratio*** Wert 1,0 erhöht sich die Abklingzeit hoher Frequenzen und ist somit länger als die Abklingzeit niedriger Frequenzen. Sinkt der ***HF-Decay-Ratio*** Wert unter 1,0, dann sinkt auch gleichzeitig die Abklingzeit hoher Frequenzen und ist damit kürzer als die der niedrigen Frequenzen. Ist der ***HF-Decay-Ratio*** Wert kleiner 1,0 wirkt der Nachhall natürlicher. Bei einem Wert größer 1,0 wirkt er brillanter (*wiki.thedarkmod.com/index.php?title=Setting_Reverb_Data_of_Rooms_%28EAX%29).*
4. ***Pre-Delay*** ist die Vorverzögerung des Halls und kann von 0 bis 300ms definiert werden. Ein kleines ***Pre-Delay*** mit weniger als 15 ms und hohem Nachhallpegel lässt die Soundquelle weiter entfernt klingen und wird dadurch beim Spieler hinter der Lautsprecherebene wahrgenommen. Wobei ein langes ***Pre-Delay*** die Soundquelle nah klingen lässt, gleichzeitig aber den Raumanteil erhöht. Dadurch lässt sich die Nachhallzeit verringern, ohne große Auswirkung auf den Räumlichkeitseindruck (*www.sengpielaudio.com/AnfangszeitlueckeUndPredelay.pdf*).
5. ***Late-Delay*** ist die Verzögerung der ***Late-Reflections*** relative zu den ***Early-Reflections***.Die Verzögerung kann von 0 – 0.1 Sekunden eingestellt werden.
6. Die ***Early-Reflections*** sind die ersten reflektierten Schallwellen, die am Ohr des Spielers ankommen. Ihr Pegel kann um -100dB abgesenkt und maximal bis 10dB angehoben werden:„The EARLY REFLECTIONS enable the human brain to quickly identify the room size. They are therefore the most critical part of a reverb effect, if a room simulation is the goal" (*audacity.sourceforge.net/manual-1.2/effects_reverb.html*).
7. ***Late-Relfections*** regelt die Pegelanhebung bzw. -absenkung der Schall-Reflexion, die erst nach mehreren Reflexionen am Ohr ankommen. (vgl. Abbildung 3)[26].

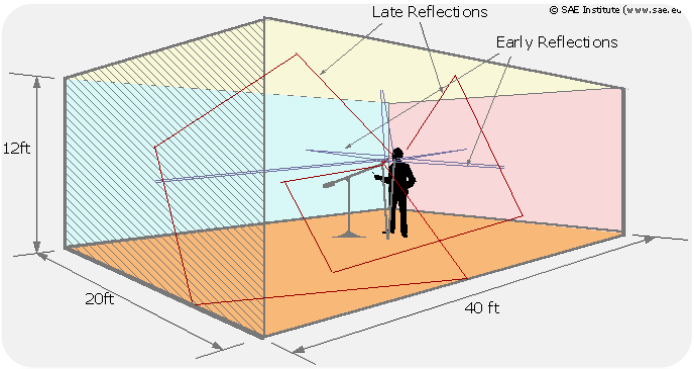


Abbildung 3

8. ***Diffusion*** gibt an, wie stark die Reflexionen im Laufe der Zeit geglättet werden. Der Wertebereich geht von 0 – 100. Mittels ***Diffusion*** können die unterschiedlichen Reflexionseigenschaften von Materialien simuliert werden. Hohe Werte können grobe und dadurch unregelmäßige Objekte simulieren, wie zum Beispiel Holzwände, wohingegen sich niedrige Werte für glatte Oberflächen, wie beispielsweise Fliesen eigenen.

[25] *www.sengpielaudio.com/Rechner-RT60.htm*

[26] Vgl. Abbildung *www.sae.edu/reference_material/audio/pages/Reverb.htm*

9. ***Density*** steuert die Dichte des Nachhalls. Je höher der Wert, desto mehr Reflexionen werden erstellt und desto dichter ist der Nachhall. Der Wertebereich geht von 0 – 100 (www.retrosynth.com/~analoguediehard/studio/effects/digitalreverb.html).
10. Der Parameter ***HF Gain*** verstärkt die hohen Frequenzen des Hallanteils. Hiermit kann die Schallabsorption des Raums bestimmt werden. Unmöblierte Räume klingen heller, wohingegen Möblierte wärmer klingen.
11. Mittels ***LF Gain*** wird die Verstärkung, der tiefen Frequenzen des Hallanteils geregelt. Die Verstärkung der Tiefen im Reverb eignet sich für solide und große Objekte, wie eine Höhle oder Kirche. Dadurch entsteht das typische Rumpeln im Hall. (www.soundonsound.com/sos/may00/articles/reverb.htm?print=yes)
12. ***HF Crossover*** definiert den Schwellwert bis zu welcher Frequenz der ***HF Gain*** anschlägt. Der ***HF Crossover*** hat einen Regelbereich von 20Hz bis 20kHz.
13. ***LF Crossover*** legt den Schwellwert fest, bei dem die Frequenz der ***LF Gain*** anschlägt. Der Regelbereich der ***LF Crossover*** liegt dabei von 20Hz bis 20kHz.

Des Weiteren kann jedem Sound Event im FMOD-Designer ein *Wet*- und *Dry* - Wert zugewiesen werden. Hiermit kann der Anteil des Orginal (*Dry*)- bzw. Effektanteiles (*Wet*) geregelt werden. Das daraus resultierende Reverb Preset „Huette“ für die Holzhütte ist in Abbildung 4 ersichtlich.

Huette	
nMasterLevel	-10
fDecayTime	918
fHFDecayRatio	0.814
fPreDelay	2
fLateDelay	2
nEarlyReflections	10
nLateReflections	15
fDiffusion	100
fDensity	75.1
nHFGain	-18
nLFGain	-47
fHFCrossover	5000
fLFCrossover	250

Abbildung 4: Reverb Preset „Huette“

5.2 Atmo (AmbientVolume)

Eine abwechslungsreiche und damit glaubwürdige Atmo besteht aus einem *Loop* und mehreren zufällig abgespielten *One-Shots*. Der *Loop* bildet den Grundsound der Atmo, wohingegen die *One-Shots* für Abwechslung sorgen.

5.2.1 Einfügen einer Atmo mit zusätzlichen One-Shots

Atmos können einer *Area* zugeordnet werden. Dazu muss sie mit einem *AmbientVolume Entity* verknüpft werden und die Eigenschaften dieses *Entity* entsprechend angepasst werden.

Enabled	Aktiviert bzw. deaktiviert den Entity
IgnoreCulling	Die Atmo wird mit zunehmender Entfernung, beziehungsweise wenn zu viele visAreas1 dazwischen sind, gedämpft. (bei aktiviertem IngoreCulling wird der Effekt deaktiviert)
IgnoreObstruction	Falls aktiviert haben Wände keine Auswirkung auf den Sound.
LogBattleValue	Der aktuelle battle Wert wird in der Konsole wiedergegeben. (vgl. *5.2.2*)
Name	Name des Atmo Sound Events, der abgespielt werden soll.
OuterRadius	Entfernung ab wann die Atmo außerhalb der Area zu hören ist.
SensitiveToBattle	Bestimmt ob die Atmo durch einen Kampf beeinflusst werden kann (vgl. *5.2.2*).

Der *Area* mit dem *AmbientVolume Entity* können noch *One-Shots* zugefügt werden. Dazu können beliebig viele *RandomSoundVolume Entities* zusätzlich mit der *Area* verknüpft werden. Die *RandomSoundVolume Entities* bieten folgende Einstellungsmöglichkeiten.

DiscRadius	Bestimmt den Umkreis in dem die *One-Shots* erscheinen.
Enabled	siehe oben
IgnoreCulling	siehe oben
IgnoreObstruction	siehe oben
LogBattleValue	siehe oben
MaxWaitTime	Definiert die maximale Zeit, die vergeht bis der *One-Shot* abgespielt wird.
MinWaitTime	Definiert die minimale Zeit, die vergeht bis der *One-Shot* abgespielt wird.
Name	Name des *One-Shot* Sound Events, der abgespielt werden soll.
RandomPosition	Bestimmt, ob die Position des *One-Shots* bei jedem erneuten Abspielen per Zufall geändert werden soll. Falls dies aktiviert wird (*true*), muss das *Entity* einer *Area* zugeordent werden. Ansonsten (*false*) darf es nicht mit einer *Area verbunden werden.*
SensitiveToBattle	siehe oben

Für das folgende Beispiel wurden die Einstellungen wie in Abbildung 5 gewählt.

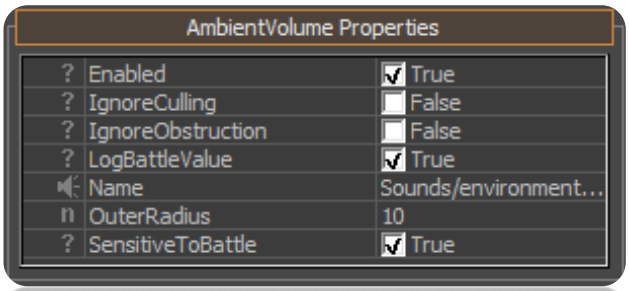


Abbildung 5: Einstellungsmöglichkeiten des *AmbientVolume Entity.*

[27] VisArea werden benutzt um Innenbereiche zu definieren.

5.2.2 Erstellen einer Atmo

Der Sound Event für die Wald Amto des Demo Levels besteht aus vier Ebenen und besitzt drei verschiedene Parameter, um auf Spielereignisse reagieren zu können. Der erste Parameter (*spread*) bestimmt, ob der Spieler sich außerhalb oder innerhalb des Bereichs der Atmo befindet. Nähert sich der Spieler der *Area Shape* Grenze, welche die Atmo enthält (vgl. *5.2.1*), sinkt der *spread* Wert langsam von 1 auf 0. Bei einem Wert von 0 befindet sich der Spieler vollständig innerhalb der *Area Shape*, wohingegen er sich bei einem Wert von 1 außerhalb der *Area* aufhält. Der *spread* Parameter regelt zwei Effekte, und zwar einerseits den *3D_Pan_Level* und andererseits den *3D_Speaker_spread*. Der *3D_Pan_Level* realisiert den Wechsel von einem 3D- zu einem 2D Sound (*spread* =0). Bei einem 3D Sound kann die Richtung der Soundquelle bestimmt werden, wohingegen 2D Sounds omnipräsent sind. Daher muss die Atmo als 3D Sound (*spread*=1) definiert werden, sofern der Spieler sich außerhalb der *Area Shape* befindet und im Gegensatz dazu als 2D Sound (*spread*=0), sofern dieser sich innerhalb der *Area Shape* aufhält.

Dabei determiniert der *3D_Speaker_spread*[28] die Breite des Sounds (hier in diesem Fall ist es die Atmo) über einen Öffnungswinkel. Ist die Soundquelle punktförmig, dann ist dieser Öffnungswinkel ebenso klein. Da es sich in dem Beispiel um einen Wald handelt, was bedeutet, dass der Sound von einer großen Fläche abgestrahlt wird, muss in diesem Fall ein großer Öffnungswinkel von 90° (*3D_Speaker_spread*=90) gewählt werden.

Der *battle* Parameter kann die Atmo im Falle eines Kampfes verändern. Der Wert dieses Parameters erhöht sich automatisch bei Beginn eines Kampfes innerhalb der *Area*, sofern *SensitiveToBattle* aktiviert (*true*) ist. Mit steigendem *battle*-Wert wird der Wald-Atmo-Loop für den Tag (Wald Atmo mit Vogelgezwitscher) und der Wald Atmo Loop für die Nacht (Wald Atmo mit zirpenden Grillen) aus-, sowie ein spezieller Wald Atmo Loop für den Kampf (Wald Atmo ohne Vogelgezwitscher) eingeblendet. Bei einem sinkenden Wert ist es entgegengesetzt. Der *battle* Parameter startet außerdem ab einem Wert von 0.5 einen *One-Shot* mit wegfliegenden Vögeln, die beim Kampfbeginn aufgeschreckt werden.

[28] *wiki.etc.cmu.edu/unity3d/index.php/3D_Sound*

Der letzte Paramater (*daylight*) beeinflusst die Atmo je nach Tageszeit. Mit sinkendem Wert, also abnehmendem Tageslicht, wird die Wald Atmo für die Nacht eingeblendet und die Wald Atmo für den Tag der ersten Ebene ausgeblendet.

Die *One-Shot* Sounds müssen ebenfalls über Parameter dem Spielverlauf angepasst werden. Der *„nice_bird_oneshot“* Sound Event (Vogelgesang) wird beispielsweise sowohl mit zunehmenden *battle* als auch *daylight* Werten komplett ausgeblendet, da der Vogelgesang weder während eines Kampfes noch in der Nacht zu hören sein soll. Die *One-Shots* der *RandomSoundVolume Entity werden* zufällig im Raum verteilt, daher muss über den *distance* Parameter mit größer werdender Entfernung zum Spieler der Sound Event ausgeblendet werden. Dadurch klingen weiter entfernte Soundquellen leiser als Nahe.

5.3 Sound Moods

Sound Moods bieten die Möglichkeit die Soundmischung in bestimmten Situationen zu ändern. Mit ihnen können, die in den FMOD Projekten erstellten Kategorien (vgl. *4.2.1.5*) in ihrer Lautstärke geändert, mit einem Pitch versehen, per EQ und Filter verändert und die Dynamik mittels Kompressor bearbeitet werden. Einstellungen die dabei für die jeweiligen Kategorien vorgenommen werden, wirken sich dabei auf alle Sound Events aus, die diesen Kategorien zugeordnet wurden.

Der sogenannte *„default“ Sound Mood* ist automatisch aktiviert sobald kein anderer *Sound Mood* aktiv ist. Dieser spezielle *Sound Mood* repräsentiert daher die Abmischung, die zum größten Teil des Spiels zu hören ist. Jedem *Sound Mood* kann außerdem eine Priorität zugeteilt werden. Sind mehrere *Sound Moods* gleichzeitig aktiv, regelt dieser Prioritätswert wie stark der Einfluss der einzelnen *Sound Moods* auf den Gesamt-Mix ist. Hat beispielsweise ein *Sound Mood* die Priorität 10 und das andere 20, werden die Werte aller Kategorien um den Faktor 2 in Richtung des *Sound Moods* mit der höheren Priorität verschoben.

5.3.1 Erstellen eines Sound Moods

Sound Moods können in der DataBase Ansicht unter dem Reiter *Sound Moods* erstellt und bearbeitet werden. Beim Erstellen eines neuen *Sound Mood* enthält dieser automatisch die Kategorien, die im FMOD Designer definiert wurden (vgl. *4.2.1.5*).

5.3.2 Verwenden von Sound Moods

Sound Moods können per Skript, *Flow Graph* oder über eine *Area* eingefügt werden. Wie ein *Sound Mood* eingebunden wird, ist vor allem davon abhängig, ob der *Sound Mood* global benötigt wird oder ob es sich auf einen lokalen Bereich innerhalb des Levels beschränkt.

5.3.2.1 Implementierung per Code:

Beispielsweise soll das *„low_health“ Sound Mood* immer dann aktiviert werden, wenn der Spieler nur noch wenige Gesundheitspunkte hat. Dieses Ereignis ist jedoch nicht ortsabhängig und folglich ist eine Einbindung über eine Area nicht möglich. Daher wird das *„low_health“ Sound Mood* wie folgt per Code eingebunden:

Auszug aus der Datei: *Player.cpp*

```
void CPlayer::SetHealth(float health )
 {
      if(m_stats.isGrabbed)
      health -=1;  //Trigger automatic thrown
      float oldHealth = m_health;
      CActor::SetHealth(health);
       if(IsClient())
 {
      float fHealth = m_health / m_maxHealth * 100.0f + 1.0f;
      const float fMinThreshold = 20.0f;
      const float fMaxThreshold = 30.0f;

      if(fHealth < fMaxThreshold && (m_fLowHealthSoundMood == 0.0f||fHealth < m_fLowHealthSoundMood))
 {
            float fPercent = 100.0f;
            if(fHealth >= fMinThreshold)
            {
            fPercent = (MAX(fHealth,0.0f) - fMinThreshold) * 100.0f / (fMaxThreshold - fMinThreshold);  }
            SAFE_GAMEAUDIO_SOUNDMOODS_FUNC(AddSoundMood(SOUNDMOOD_LOWHEALTH,fPercent));
                  m_fLowHealthSoundMood = fHealth;
            }
            else if(fHealth > fMaxThreshold && m_fLowHealthSoundMood > 0.0f)
      {           m_fLowHealthSoundMood = 0.0f;
            }     }
```

An dieser Stellle soll jedoch nicht näher auf die Einzelheiten des Codes eingegangen werden, da dies den Rahmen der Arbeit sprengen würde[29]. Entscheidend ist in diesem Fall die rot markierte Zeile, in der bei bestimmten Voraussetzungen (grüne Markierung) durch den Aufruf der Funktion *AddSoundMood* das *„low_health“ Sound Mood* aktiviert wird.

[29] Eine detaillierte Beschreibung der Programmiersprache C++ ist unter folgendem Link zu finden: www.cplusplus.com

5.3.2.2 Implementierung per Area:

Die Verknüpfung eines *Sound Moods* mit einem *Area* Objekt funktioniert wie zuvor bei den *AmbientVolume* oder *ReverbVolume Entities*. Sobald das *Sound Mood* mit der *Area* verknüpft ist, muss dessen Name unter ***SoundMoodName*** eingetragen werden. Über ***OuterRadius*** kann der Radius um die *Area* bestimmt werden, ab wann der *Sound Mood* eingeblendet wird. ***Enabled*** aktiviert den *SoundMood* (vgl. Abbildung 8). Im hier gewählten Beispiel sollen beim Betreten eines bestimmten Bereichs die Bewegungsgeräusche ausgeblendet und im Gegensatz dazu der Fokus auf die Musik gelegt werden. Um dies zu erreichen, wird zuerst ein *Sound Mood* mit dem Namen „*focus_music*“ erstellt, wobei die Lautstärke der Kategorie Musik höher sein muss als im „*default*“ *Sound Mood*. Um den Effekt noch weiter zu verstärken, kann zusätzlich ein Kompressor hinzugefügt werden, so dass der Sound der Musik verdichtet und noch etwas stärker hervorgehoben wird. Im Anschluss daran wird ergänzend ein *Sound Mood* Objekt mit dem ***SoundMoodName*** „*focus_music*“ mit der *Area* verknüpft.

Properties	
Volume [0..1]	1
Pitch [-4..4]	0
LowPass Filter Cutoff [10..22,000]	22000
HighPass Filter Cutoff [10..22,000]	10
HighPass Filter Resonance [1..10]	1
Compressor Threshold [-60..0]	-33.07
Compressor Attack [10..200]	50
Compressor Release [20..1000]	50
Compressor GainMakeup [0..30]	3.33
ParamEQ-1 Gain [0.05..3]	1
ParamEQ-1 Center [20..22000]	1000
ParamEQ-1 Bandwidth [0.2..5]	1
ParamEQ-2 Gain [0.05..3]	1
ParamEQ-2 Center [20..22000]	1000
ParamEQ-2 Bandwidth [0.2..5]	1

Abbildung 6: focus_music *Sound Mood*. Einstellungen für die Kategorie music

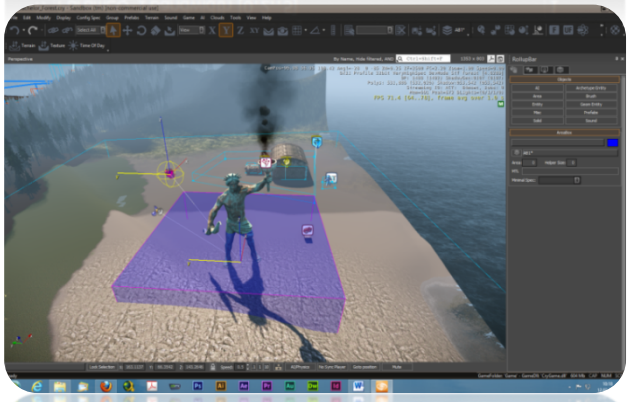


Abbildung 6: *Area Box* mit verknüpftem *Sound Mood*

SoundMoodVolume Properties	
Enabled	True
OuterRadius	5
SoundMoodName	focus_music

Abbildung 8: *SoundMoodVolume Entity*

5.3.2.3 Implementierung per Flow Graph:

Im folgenden Beispiel soll beim Betreten eines Fahrzeugs ein *Sound Mood* aktiviert werden, das die Umgebungsgeräusche leiser macht und deren Höhen reduziert. Da ein Fahrzeug beweglich und somit nicht ortsgebunden ist, entfällt die Einbindung über eine *Area*. Zur Umsetzung kommen sowohl eine codeba-

sierte Einbindung als auch eine Lösung mittels *Flow Graph* in Frage. Im Folgenden soll kurz die letztere Möglichkeit beschrieben werden:

Um eine Einbindung mittels *Flow Graph* zu erreichen, wird zunächst ein neues *Sound Mood* mit dem Namen *„in_vehicle“* erstellt. Dieses erhält eine hohe Priorität, um das *„default“ Sound Mood* zu übertönen. Im Beispiel wurde eine Priorität von 80 verwendet. Im Anschluss daran müssen mit Ausnahme der Musik und Fahrzeug Kategorie, alle Kategorien in ihrer Lautstärke reduziert werden und dumpfer klingen. Dazu wird die Lautstärke dieser Kategorien auf etwa 0.6 (Musik und Fahrzeug haben einen Lautstärkewert von 1.0) und den Gain des ersten Equalizers auf 0.2 gestellt, sowie die Center Frequenz auf 10000 und die Bandweite auf 3. Dadurch klingen nun alle Umgebungsgeräusche leiser und dumpfer.

Im nächsten Schritt wird mit Rechtsklick auf das Fahrzeug Objekt ein *Flow Graph* erstellt. Für das Beispiel werden ein *Game:LocalPlayer Node*, ein *Vehicle:VehiclePassenger Node*, zwei *Math:Equal Nodes* und ein *Sound:SoundMood Node* benötigt. Das *Game:LocalPlayer Node* gibt die *Id* des Spielers aus (hier der Wert: *30583 vgl. Abbildung 9*). Die Ausgänge *ActorIn* und *ActorOut* des *Vehicle:VehiclePasenger* geben die *Id* des Charakters aus, der das Fahrzeug betritt (*ActorIn*) bzw. verlässt (*ActroOut*). Die *Math:Equal Nodes* vergleichen die *Id* des Spielers mit der des ein- bzw. aussteigenden Spielers. Somit wird sichergestellt, dass der *Sound Mood* nur dann aktiviert wird, wenn der Spieler selbst das Fahrzeug betritt bzw. nur dann deaktiviert, wenn der Spieler es auch selbst wieder verlässt. Ist die *Id* gleich, wird je nachdem ob der Spieler ein- oder aussteigt der *Sound Mood „in_vehicle“* ein- (*TriggerFadeIn*) bzw. ausgeblendet (*TriggerFadeOut*). Die Ein- und Ausblendzeit des SoundMoods kann im Node definiert werden. Sie werden in diesem Beispiel auf kurze Zeiten eingestellt, da es unmittelbar beim Betreten des Fahrzeugs aktiv sein soll. Die Stärke des Effekts kann über *FadeInValue* reguliert werden, wobei 1 die maximale Stärke ist.

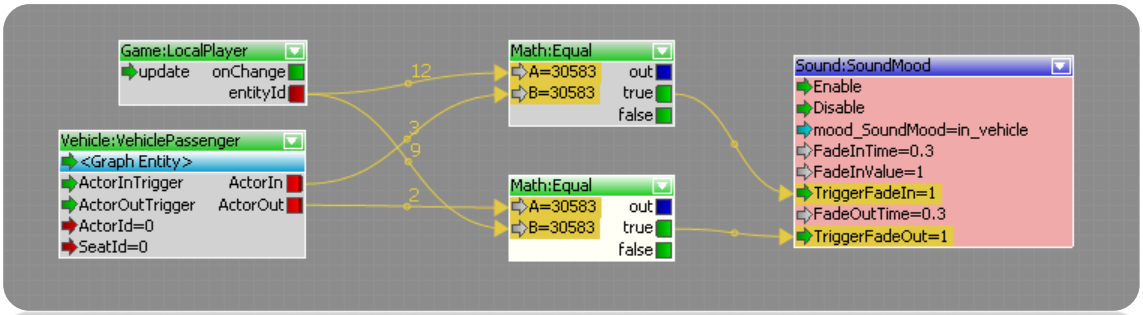


Abbildung 7: Sound Mood Aktivierung über einen Flow Graph.

5.4 Sound und Partikel Effekte

Explosionen, Rauch, Wettereffekte, Feuer usw. werden in modernen Shootern mit einem Partikelsystem umgesetzt. Ein Partikelsystem besitzt einen Emitter, der Partikel erschafft, verwaltet und wieder entfernt.

Eine einfache bildliche Darstellung findet sich auf codeworx.org:

> „Nehmen wir zum Beispiel einmal eine Landschaft in der es regnet. Die einzelnen Regentropfen wären hier unsere Objekte und der Regen im Ganzen das Partikelsystem. Die einzelnen Regentropfen verhalten sich gemäß den Gesetzen der Schwerkraft und können zum Beispiel von Winden beeinflusst werden. Regenwolken dienen als Austrittspunkte der Regentropfen, und würden dann als Emitter bezeichnet. Trifft ein Regentropfen auf den Boden so „stirbt" er und wird aus der Liste gelöscht" (*www.codeworx.org/opengl_par1.php*).

Das in die CryENGINE integrierte Partikelsystem bietet zahlreiche Einstellungsmöglichkeiten, wie die Lebenszeit der Emitter oder Partikel oder aber auch ob die Partikel von einem Punkt aus oder einer Fläche erstellt werden, sowie vieles mehr. In diesem Abschnitt soll nicht weiter auf die Erstellung eines Partikeleffekts eingegangen, sondern ausschließlich die Einbindung eines Sound Events in einen Partikeleffekt näher erläutert werden.

Die verknüpften Sound Events werden automatisch beim Starten des Partikeleffekts abgespielt. Durch das direkte Einbinden von Sound Events in die Partikeleffekte erspart man sich das Erstellen eines zusätzlichen Sound Events, der zeitgleich mit dem Partikeleffekte aufgerufen werden müsste.

Um einem Partikeleffekt einen Sound zuzuweisen, wechselt man in der Database Ansicht zu dem Unterpunkt *Particles* und wählt dort den gewünschten Partikeleffekt aus. Unter dem Reiter Sound finden sich nun die Einstellungsmöglichkeiten für die Sound Wiedergabe (vgl. Abbildung *10*).

Unter Sound wird der Sound Event angegeben, der mit dem Partikel abgespielt werden soll. Der Parameter Sound FXParam bestimmt den maximalen Wert des *particlefx* Parameters, der an den Sound Event übermittelt wird. Im Beispiel beträgt dieser Wert 1. Im FMOD-Designer muss daher der Wertebereich des *particlefx* Parameters ebenfalls für einen Maximalwert von 1 definiert werden. Über diesen Parameter kann beispielsweise ein Ausblenden des Sound Events realisiert werden.

Die Emitter Strength Kurve bestimmt den *particlefx* Wert, wobei die x-Achse für die Zeit und die y-Achse für den Wert steht. Die Zeitachse wird hierbei über den Parameter Sound Control Time bestimmt, der drei verschiedene Einstellungsmöglichkeiten bietet:

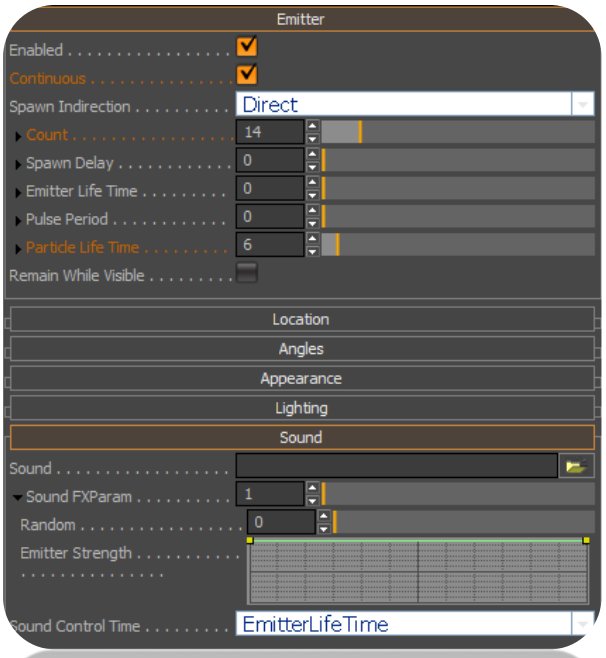


Abbildung 8: Emitter & Sound Parameter des Partikelsystems

EmitterLifeTime: Die Zeitachse entspricht der Lebenszeit des Emitters (*Emitter Life Time*).

EmitterExtendedLifeTime: Die Zeitachse entspricht der Zeit, bis der letzte Partikel noch „am Leben ist“ (*Emitter Life Time* + *Particle Life Time*).

EmitterPulsePeriod: Die Zeitachse wird in einer Endlosschleife wiederholt. Die Dauer eines Durchlaufs wird durch die Emitter Einstellung *Pulse Period* definiert.

Wenn der Parameter *particlefx* in FMOD einer Lautstärkekurve zugeordnet ist, wird je nach Kurvenform der Sound beim Abspielen des Partikels ein- oder ausgeblendet.

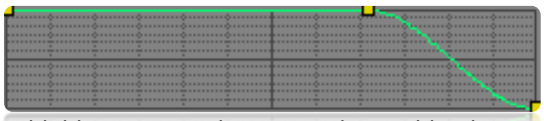


Abbildung 9: Sound Event wird ausgeblendet

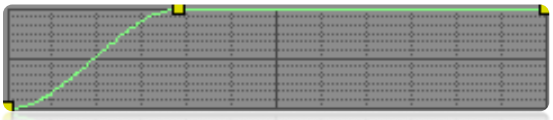


Abbildung 10: Sound Event wird eingeblendet

Der Parameter *Random* subtrahiert jedes Mal wenn der Partikeleffekt abgespielt wird einen zufälligen Wert zwischen 0.0 und dem eingestellten Random Wert von dem *particlefx* Wert. Das heißt, bei einem *Random* Wert von 0.1 liegt der maximal ausgegebene *particlefx* Wert zwischen 1 und 0.9.

Ist ein verzögertes Abspielen des Sounds notwendig, beispielsweise zur Wahrung der Synchronität, kann man dem Partikeleffekt ein leeres Partikelsystem hinzufügen und den Sound Event bei diesem einfügen. Erhöht man nun den *Spawn Delay*[30] Wert, wirkt sich die Verzögerung auch auf den Beginn des Sound Events aus.

[30] Verzögerungszeit, bevor der Emitter anfängt Partikel zu erstellen.

Im Beispielvideo ***13*** ist ein Partikeleffekt mit drei Sound Events zu sehen. Eine Explosion gleich zu Beginn, ein verzögertes Feuer-Geräusch im Anschluss und verzögerte Trümmergeräusche am Ende.

5.5 Waffensounds

Für Waffen werden eine ganze Reihe unterschiedlicher Soundeffekte benötigt. Je nach Waffe benötigt man Schüsse, Nachladegeräusche und Spezialsounds, die das Wechseln des Schussmodus, Ladehemmungen oder auch das Geräusch eines leeren Magazins simulieren.

Die folgenden Beispiele für das Erstellen der Sound Events und deren Implementierung in das Spiel werden anhand eines Sturmgewehrs (SCAR) erklärt.

5.5.1 Erstellen der Schuss Sound Events

Drei Sound Events sind für die Schusssounds der SCAR von Bedeutung[31]: Ein Sound Event für den Einzelschuss Modus („*fire_single_fp*“), einer für Dauerfeuer („*fire_loop_fp*“) und einer für den Nachhall („*fire_tail_fp*“), der automatisch im Anschluss an das Dauerfeuer Sound Event abgespielt wird.

Das „*fire_single_fp*“ Sound Event besitzt zwei Ebenen und darüber hinaus den Parameter *environment_sound*. Die erste Ebene enthält einen *Sound_Def* mit fünf Schüssen, wovon einer bei jedem Schuss per Zufall ausgewählt und immer unabhängig vom *environment_sound* Parameter abgespielt wird. Die zweite Ebene besteht aus zwei unterschiedlichen Nachhall (*tail*) Sounds, von denen je nach *environment_sound* einer zeitgleich mit dem Schuss selbst aus der ersten Ebene abgespielt wird. Der erste Nachhall ist für den Außenbereich vorgesehen und wird bei einem *environment_sound* Wert kleiner als 1.5 abgespielt. Der Zweite ist für den Innenbereich zuständig, der bei einem *environment_sound* Wert größer oder gleich 1.5 aktiviert wird (vgl. *5.1*).

Für den Dauerfeuer Modus des Sturmgewehrs werden Loops verwendet. Um Arbeitsspeicher zu sparen, ist es sinnvoll kurze Loops zu verwenden. Um dabei eine monotone Wirkung und eine Abnutzung des Loops zu vermeiden (vgl. *3.5.2*), ist der „*fire_loop_fp*“ Sound Event ein Loop, der aus mehreren

[31] Es werden nur die Schusssounds für die First Person Ansicht berücksichtigt.

kurzen Loops besteht. Dieser Loop wird ständig per Zufall aus mehreren, unterschiedlichen Loops generiert. Dazu wird zunächst ein vorhandener Maschinengewehr Sound mit einem Audio Editor in etwa fünf einzelne Audiodateien von je ca. vier bis sechs Schuss aufgeteilt.

Beim Schneiden muss darauf geachtet werden, dass die Audiofiles am Anfang und Ende jeweils an einem Nulldurchgang geschnitten wurden, um Störgeräusche beim späteren loopen zu vermeiden. Aus diesen fünf Dateien wird ein *Sound Def* erstellt und allen die gleiche Abspielwahrscheinlichkeit (*Play Percentage*) zugewiesen. Außerdem wird der Play Mode auf „*Random No Repeat"* gestellt, um zu vermeiden, dass zweimal das gleiche Audiofile unmittelbar nacheinander abgespielt wird. Des Weiteren muss die *Spawn Time*[32] auf einen Bereich von sowohl minimal 1ms als auch maximal 1ms gestellt werden, so dass die Pausen zwischen den Schussloops immer gleich groß und quasi nicht vorhanden sind.

Dieser *Sound Def* bildet im „*fire_loop_fp"* Sound Event die erste und einzige Ebene. Für einen sauber generierten Zufalls-Loop müssen Änderungen an den *Sound Instance Properties*[33] vorgenommen werden. Diese bestimmen das Abspielverhalten des *Sound Def*. Hierbei ist es wichtig den *Start Mode* auf „*Wait for previous"* zu stellen, wodurch der nächste Schussloop erst beim Beenden des Vorherigen abgespielt wird. Da jeder Schussloop immer nur einmal hintereinander gespielt wird, muss außerdem der *Loop* Mode auf „*Oneshot*" gestellt werden, um zu verhindern, dass der zuerst abgespielte Schussloop geloopt wird.

5.5.2 Einfügen von Waffensounds über XML[34]

Jede Waffe besitzt eine eigene XML-Datei in der die Eigenschaften der Waffe, wie zum Beispiel Schaden, Schussgeschwindigkeit und Magazingröße, definiert sind. Zudem werden in dieser XML-Datei auch mögliche Aktionen einer Waffe festgelegt. Den Aktionen können Animationen und Sounds zugewiesen werden. Der hier angeführte Auszug der XML-Datei eines Sturmgewehrs (SCAR.xml) zeigt, welche Sounds ausgelöst werden sobald der Spieler verschiedene Aktionen mit der Waffe durchführt:

---

[32] Der Sound wird zufällig innerhalb der definierten minimal bzw.- maximal Zeit abgespielt.

[33] Die Sound Instance Properties können über einen Rechtsklick auf den Sound Event angepasst werden.

[34] „Die Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache"), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten" (*www.a3w.de/2011/02/xml/*).

```
<action name="fire">
<animation target="firstperson" name="fire_bullets_right_%suffix%01" />
<animation target="owner" name="shoot" />
 <sound target="firstperson" name="sounds/weapons:scar:fire_single_fp" radius="200" static="0" />
 <sound target="thirdperson" name="sounds/weapons:scar:fire_single_3p" radius="200" static="0" />
</action>
<action name="rapid_fire">
 <sound target="firstperson" name="sounds/weapons:scar:fire_loop_fp" radius="20" static="1" synched="1"/>
 <sound target="thirdperson" name="sounds/weapons:scar:fire_loop_3p" radius="20"static="1" synched="1"/>
</action>
<action name="spin_down">
 <sound target="firstperson" name="sounds/weapons:scar:fire_tail_fp" radius="200" static="0" synched="1"/>
 <sound target="thirdperson" name="sounds/weapons:scar:fire_tail_3p" radius="200" static="0" synched="1"/>
</action>
```

Das Attribut des *action* Tag gibt dabei die Art der Aktion an. Die Aktion *fire* steht für einen einzelnen Schuss und *rapid fire* für Dauerfeuer. Die Aktion *spin_down* ist die Hallfahne des letzten Schusses beim Dauerfeuer und wird automatisch ausgeführt sobald der Spieler die Schusstaste loslässt.

Das *sound* Tag weist den Aktionen ein Sound Event zu und kann mehrere Attribute besitzen. Das Attribut *target* gibt an, für welche Spielansicht welches Sound Event abgespielt wird. Da man sich bei der „Third-Person"- Ansicht etwas weiter weg vom Spielecharakter befindet, sollten ebenso die Sound Events auch etwas weiter entfernt klingen. Die „First-Person"-Ansicht ist hier näher an der Schallquelle, wodurch der Waffensound direkter klingt.

Abbildung 11: Vergleich der Spielansichten "First Person" (oben) und "Third Person" (unten).

Das Attribut *name* enthält den Pfad zum Sound Event, welches von der Aktion ausgelöst werden soll und besitzt das folgende Format:

Verzeichnis/Projektname: Gruppe: Eventname

Der *radius* gibt die Entfernung an, in der ein Gegner auf eine Aktion des Spielers reagiert. Das *static* Attribut legt fest, ob der Sound beim Waffenwechsel unmittelbar gestoppt wird oder ob er noch zu Ende gespielt wird. Das *synched* Attribut bestimmt wiederum, ob der Sound bei vorhandenen Synchronisationspunkten bis zum

Abbildung 12: Gesetzer Marker in Sound Forge.

nächsten Marker gespielt oder direkt beim Beenden der Aktion angehalten wird. Diese Funktion sorgt dafür, dass Loops an bestimmten Positionen im Soundfile beendet werden, wodurch ein abruptes Ende des *Loops* verhindert wird. Um diese Funktion zu nutzen, müssen bereits beim Erstellen bzw. Bearbeiten der Audio Datei im Audio Editor (z.B. Sound Forge oder Audition) sogenannte Marker gesetzt werden. Beispielsweise sollten beim Dauerfeuer die Marker unmittelbar vor dem nächsten Schuss gesetzt werden (vgl. Abbildung *14*). Zusätzlich müssen in den Eigenschaften der FMOD Sound Bank die Synchronisations-Punkte aktiviert und schließlich das *synched* Attribut auf 1 gestellt werden.

5.5.3 Vertonen der Nachlade-Animation mit dem Charakter Editor

Während die meisten Sounds einfach über eine XML eingefügt werden können, erfordern manche animierte Vorgänge das Einfügen der Sound Events über den Charakter Editor.

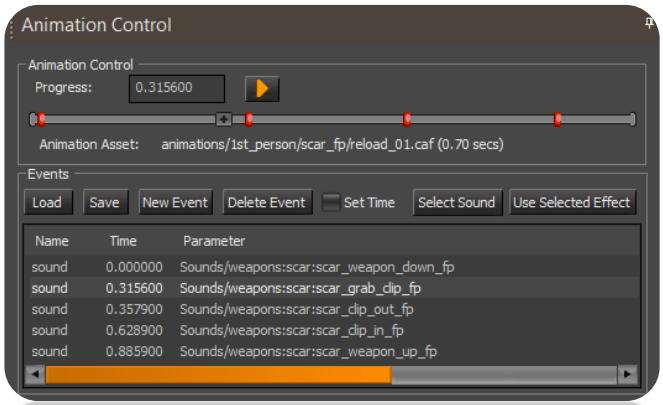


Abbildung 13: *Animation Control* des Charakter Editors

Aufgrund der Tatsache, dass beim Nachladen mehrere einzelne Sounds benötigt werden (Entfernen / Einfügen des Magazins, Heben / Senken der Waffe, etc.), die synchron zur Nachladeanimation abgespielt werden sollten, ist es nötig die einzelnen Sounds mit dem Charakter Editor zu platzieren. Ein bereits fertig montierter Sound Event mit allen erforderlichen Einzelsounds kann in diesem Fall nur schwer mit der Animation synchronisiert werden.

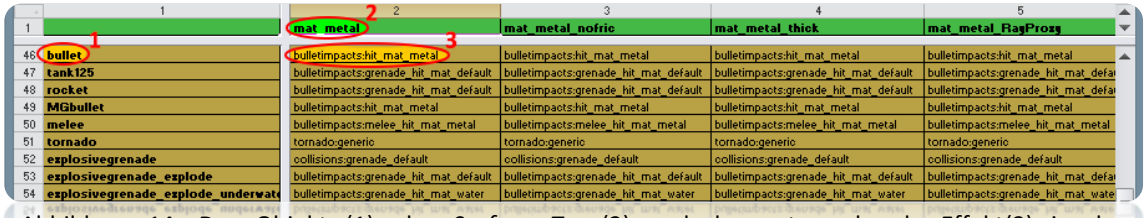
Um die Nachlade-Animation zu vertonen, muss die Waffe im Charakter Editor geladen und im Fenster „*Animations*“ die entsprechende Nachladeanimation ausgewählt werden. Unter dem Bedienfeld „*Animation Control*“ können Sound Events innerhalb der Animation verteilt werden. Durch das Klicken auf den Button „*New Event*“ wird ein neues Event erstellt, das durch den Button „*Select Sound*“ ein Sound Event zugeordnet werden kann. Die Events sind dabei standardmäßig vom Typ *sound*[35]. Zum Bestimmen der richtigen Position eines Sounds wird der Slider im „*Animation Control*“ Bereich an die

[35] vgl. hierzu die Typen foley und footstep unter 5.7.

passende Position geschoben und der entsprechende Timecode in die entsprechende *Time* Spalte übertragen. Wurden alle Sound Events an der richtigen Stelle platziert, muss diese Zuordnung in einer .animevents Datei gespeichert werden. Anschließend muss die Charakter Parameter Datei (.chrparams), die sich im selben Ordner wie das Objekt selbst befindet, mit dieser Zeile ergänzt werden.

5.6 Materialabhängige Sounds

Die materialabhängigen Sounds können anhand des nachstehenden Beispiels erklärt werden: Ein Schuss, der in eine Holzwand trifft klingt anders als ein Schuss, der in eine Fensterscheibe einschlägt. Auch Materialien, die unmittelbar aufeinandertreffen klingen je nach Konstellation unterschiedlich. Es muss daher festgelegt werden, wie Objekt A klingt wenn es auf Objekt B trifft. Dies erfolgt im sogenannten „Material Editor[36]“ der CryENGINE, in denen Oberflächen und Objekte unter dem Punkt *Surface Type* eine Bezeichnung für ihre Materialbeschaffenheit zugeteilt bekommen. Die *Surface Types* sind in der *MaterialEffects.xml*[37] Tabelle als Tabellenheader aufgelistet. Dabei stehen die auf die *Surface Type* wirkenden Elemente in der ersten Spalte. Diese Elemente können sowohl andere *Surface Types* sein, aber auch Kugeln, Schritte, Fahrzeuge und alle weiteren Objekte, die mit der Spielewelt interagieren können.



	1	2	3	4	5
1		mat_metal	mat_metal_nofric	mat_metal_thick	mat_metal_RagProxy
46	bullet	bulletimpacts:hit_mat_metal	bulletimpacts:hit_mat_metal	bulletimpacts:hit_mat_metal	bulletimpacts:hit_mat_metal
47	tank125	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_defa
48	rocket	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_defa
49	MGbullet	bulletimpacts:hit_mat_metal	bulletimpacts:hit_mat_metal	bulletimpacts:hit_mat_metal	bulletimpacts:hit_mat_metal
50	melee	bulletimpacts:melee_hit_mat_metal	bulletimpacts:melee_hit_mat_metal	bulletimpacts:melee_hit_mat_metal	bulletimpacts:melee_hit_mat_metal
51	tornado	tornado:generic	tornado:generic	tornado:generic	tornado:generic
52	explosivegrenade	collisions:grenade_default	collisions:grenade_default	collisions:grenade_default	collisions:grenade_default
53	explosivegrenade_explode	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_default	bulletimpacts:grenade_hit_mat_defa
54	explosivegrenade_explode_underwat	bulletimpacts:grenade_hit_mat_water	bulletimpacts:grenade_hit_mat_water	bulletimpacts:grenade_hit_mat_water	bulletimpacts:grenade_hit_mat_wate

Abbildung 14: Das Objekt (1), der *Surface Type*(2) und der entsprechende Effekt(3) in der MaterialEffects.xml

Beim Zusammentreffen zweier Objekte wird ein entsprechender Effekt ausgeführt. Trifft zum Beispiel eine Kugel auf Metall, wird der Effekt *bulletimpacts:hit_mat_metal* aufgerufen.

[36] vgl. hierzu *freesdk.crydev.net/display/SDKDOC2/Material+Editor+and+Shaders*

[37] Die *MaterialEffects.xml* befindet sich unter \Game\Libs\MaterialEffects.

bulletimpacts steht hierbei für eine xml-Datei (hier: die bulletimpacts.xml), in der der Effekt *hit_mat_metal* definiert ist. Anhand eines Auszugs aus der bulletimpacts.xml kann deren Aufbau beschrieben werden:

```
1.  <Effect name="hit_mat_metal" delay="0.05">
2.   <RandEffect>
3.    <Particle><Name minscale="0.18" maxscale="2" maxscaledist="60">bullet.hit_metal.a</Name></Particle>
4.    <Particle><Name minscale="0.18" maxscale="2" maxscaledist="60">bullet.hit_metal.b</Name></Particle>
5.    <Particle><Name minscale="0.18" maxscale="2" maxscaledist="60">bullet.hit_metal.c</Name></Particle>
6.   </RandEffect>
7.   <RandEffect>
8.    <Decal minscale="0.015" maxscale="0.03">
9.     <Filename>textures/decals/metal.dds</Filename>
10.    <Material>materials/decals/metal_tin</Material>
11.   </Decal>
12.  </RandEffect>
13.  <Sound name="mainsnd">
14.   <Name>sounds/physics:bullets/impacts:metal_thick</Name>
15.  </Sound>
16. </Effect>
```

In der ersten Zeile wird der Effektname vergeben, der in der MaterialEffects.xml aufgerufen wird. Neben Partikeleffekten (Zeilen 2-6) und Textureffekten (Einschusslöcher) (Zeilen 7-12) wird auch das passende Sound Event definiert (Zeilen 13-15). Im Beispielvideo ***17*** werden die materialabhängigen Einschusssounds gezeigt. Der Schusssound selbst wurde dabei zum Verdeutlichten deaktiviert.

```
<Animation name="$AnimEventDatabase" path="PFAD\DATEINAME.animevents"/>
```

5.7 Die Vertonung von Animationen

Animationen können über den Charakter Editor vertont werden. Die Vorgehensweise entspricht dabei der in Abschnitt 5.5.3 erläuterten Methode zur Vertonung des Waffen-Nachladens. Eine Sonderstellung hat die Vertonung von Schritten.

5.7.1 Schrittgeräusche

Damit Schrittgeräusche synchron zu den Laufanimationen eines Charakters abgespielt werden, müssen diese, ähnlich wie in Abschnitt 5.5.3, mittels des Charakter-Editors eingefügt werden. Da die Schritte jedoch je nach Untergrund unterschiedlich klingen sollen, muss hier anders vorgegangen werden. Beim Vertonen von Laufanimationen wird mit Platzhaltern gearbeitet, wodurch zwar bestimmt wird wann ein Sound Event abgespielt wird, jedoch nicht festgelegt ist welcher.

Die Ermittlung des richtigen Sound Events erfolgt erneut durch die MaterialEffects.xml (vgl. *5.6*)

Zuerst wird der gewünschte Charakter im Charakter Editor geladen und anschließend eine Laufanimation gewählt. Unter „*Animation Control*“ erstellt man nun statt der vorherigen Sound Events, 2 *foley* Events und 2 *footstep* Events. Für Letztere werden die Timecodes der Bodenkontakte des linken und des rechten Fußes in die *Time* Spalte übertragen (vgl. *5.5.3*). Die *foley* Events bestimmen den Zeitpunkt der Reibungsgeräusche durch Kleidung und werden daher in etwa dann gesetzt, sobald die Beine auf einer Höhe sind.

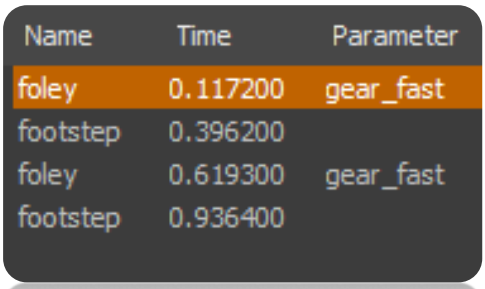


Abbildung 15: *foley* Parameter

Da die Kleidungsgeräusche nicht abhängig vom *Surface Type* sind, sondern vielmehr von der Geschwindigkeit, muss für jede unterschiedliche Laufgeschwindigkeit (Animation) ein eindeutiger Parameter für die *foley* Events eingetragen werden. Durch das Angeben eines Parameters werden die *foley* Events nicht erst in der „MaterialEffects.xml“ Tabelle abgeglichen, sondern direkt in der Datei „foley_player.xml“[38]. In dieser müssen die definierten Effekte genauso benannt werden wie der vergebene Parameter.

```
<Effect name="gear_fast">
<Sound><Name>Sounds/physics:foleys/player:gear_fast</Name></Sound>
</Effect>
```

Aufgrund der Materialabhängigkeit der Schritte darf beim *footstep* Event kein Parameter eingetragen werden, da das Sound Event über die MaterialEffekt.xml bestimmt werden muss. Die *footstep* Effekt-Definitionen werden in der Datei „footstep_player.xml“, wie unter 6.6.1 beschrieben, festgelegt.

5.8 Fahrzeugsounds

Der Sound eines Fahrzeugs wird von vielen Parametern, wie beispielsweise von der Geschwindigkeit, dem Untergrund oder auch dem Fahrzeugzustand determiniert. Die Werte dieser Parameter werden von der CryENGINE ständig aktualisiert und ändern sich entsprechend beim Beschleunigen (*rpm_scale*), beim Aufsetzen des Fahrzeugs (*intensity*) oder aber auch beim Befahren eines anderen Untergrundes (*surface*).

[38] Die XML Datei für die foley Sounds müssen in der Datei „player.lua“ bestimmt werden.

Diese Änderungen werden ständig in Echtzeit an das Sound Event übergeben, woraufhin sich der Sound des Fahrzeugs entsprechend der Fahraktion steuern lässt. Nachfolgend eine Übersicht aller Parameter:

distance	Entfernung des Spielers zum Fahrzeug
rpm_scale	Drehzahl des Motors
speed	Geschwindigkeit auf dem Boden (hat den Wert 0 wenn sich das Fahrzeug in der Luft befindet)
in_out	Bestimmt, ob der Zuhörer innerhalb (0) oder außerhalb des Fahrzeugs (1) ist.
surface	Gibt den Wert des Untergrundes aus 0.1 = Erdboden (soil) 0.2 = Schotter / Kies (gravel) 0.3 = Asphalt / Beton (concrete) 0.4 = Metall (metal) 0.5 = Vegetation / Pflanzenbewuchs (vegetation) 0.6 = Wasser (water) 0.7 = Eis / Schnee (ice)
thirdperson	Ähnlich wie in_out. Wird zum Ausblenden der Innenraum Atmo in der Third Person Ansicht benutzt.
intensity	Gibt den Ausschlag der Federung an, wobei 1 voller Ausschlag ist (z.B. beim Aufsetzen nach einem Sprung).
damage	Gibt an, wie sehr das Fahrzeug beschädigt ist wobei 1 maximaler Schaden bedeutet
scratch	Gibt die Anzahl der zerstörten Reifen an. Ein Wert von 1 bedeutet alle Reifen sind zerstört.
slip	Stärke des Drift.

Damit die Werte des Fahrzeugs an das Sound Event übergeben werden, muss im Fahrzeug Skript das richtige Sound Event zugewiesen werden.

```
<MovementParams>
   <SoundParams
      eventGroup="hmmwv" // Name der FMOD Gruppe
      eventGroupFunc="hmmwv" // FMOD Gruppe für weitere Sounds, wie z.B. die Hupe
      engineSoundPosition="engine" // Position des Sound Emitters am Fahrzeug
      id="idSound"/>
</MovementParams>
```

5.8.1 Aufbau eines Fahrzeugsounds am Beispiel des HMMWV[39]

Abbildung ***18*** und die Beispielvideos ***19 - 23*** sind für das Verständnis dieses Abschnitts eine gute Hilfe.

In der ersten Ebene befinden sich zwei Motorsound Loops, die vom Kontroll-Parameter *distance* ausgewählt werden. Der Erste ist aus der Nähe aufgenommen, wohingegen der Zweite weiter entfernt klingt. Die Loops

[39] High Mobility Multipurpose Wheeled Vehicle: amerikanisches Militär-Geländefahrzeug.

sind dabei für einen nahtlosen Übergang mit einem *Crossfade* versehen und werden je nach *distance* Wert abgespielt bzw. gemischt. Außerdem wird die Ebene mittels des *distance* Parameters mit größer werdender Entfernung zum Spieler (größerer Wert) ausgeblendet. Mit steigender Drehzahl (steigender *rpm_scale* Wert) steigt die Volume-Kurve und der Motorensound wird lauter. Zusätzlich regelt eine Pitch-Kurve die Tonhöhe des Geräuschs und simuliert somit die Drehzahl des Motors. Da ein Motor außerhalb des Fahrzeugs lauter ist als im Innenraum, hebt der *in_out* Parameter die Lautstärke für die Third-Person-Ansicht um etwa 3dB an. Die Parameter *speed* und *surface* haben dabei keinerlei Einfluss auf die erste Ebene.

Die zweite Ebene hingegen enthält sieben verschiedene Loops für die Reifenabrollgeräusche, die vom Kontroll-Parameter *surface* ausgewählt werden. Der *distance* Parameter blendet wieder den Sound der jeweiligen Entfernung entsprechend aus. Über den Parameter *speed* wird die Tonhöhe über den gesamten Wertebereich (0.0 - 1.0) langsam um etwa sieben Halbtöne angehoben, um einen Geschwindigkeitseffekt zu simulieren. Da das Fahrzeug bei Werten kleiner als 0.2 für *speed* (bzw. *rpm_scale)* noch still steht, darf die zweite Ebene erst ab einem Wert von größer als 0.2 zu hören sein. Daher wird die Lautstärke erst ab einem *speed* Wert von 0.2 eingeblendet - zunächst sehr steil und ab 0.3 flacher. Der *in_out* Parameter hebt auch hier wieder die Lautstärke für die Third Person Ansicht um etwa 3dB an und *distance* blendet die Ebene bei steigender Entfernung aus. Der *rpm_scale* Parameter hat auch auf die zweite Ebene keinen Einfluss.

Die dritte Ebene ist für den Motoren Sound im Stillstand zuständig. Da das Fahrzeug erst ab einem *rpm_scale* Wert größer 0.2 in Bewegung versetzt wird, wird diese Ebene ab diesem Wert schnell ausgeblendet. Die Funktion der *distance* und *in_out* Parameter sind wieder identisch zu den vorherigen Ebenen. Die Parameter *speed* und *surface* haben keinen Einfluss auf den Motoren Sound im Stillstand.

Die vierte Ebene ist mit einem Aufheulen des Motors versehen, das über den Kontroll-Parameter *rpm_scale* abgespielt wird sobald dieser größer als 2.5 ist.

Da das Fahrzeug nur kurz beim Beschleunigen aufheulen soll, handelt es sich dabei nicht um einen *Loop*, sondern um einen *One-Shoot* Sound. Bis auf den Parameter *dis-*

tance, der wieder die Ebene ausblendet, wird die Ebene von keinem weiteren Parameter beeinflusst.

Die fünfte und letzte Ebene unterstützt das Befahren von Wasser, indem sie jedes Mal sobald das Fahrzeug den Untergrundtyp Wasser betritt (Kontroll-Parameter *surface*=0.6) ein kurzes Wasserplätschern (*One-Shoot*) abspielt. Bis auf das typische Verhalten der Parameter *distance* und *in_out* haben keine weiteren Parameter einen Einfluss auf diese Ebene.

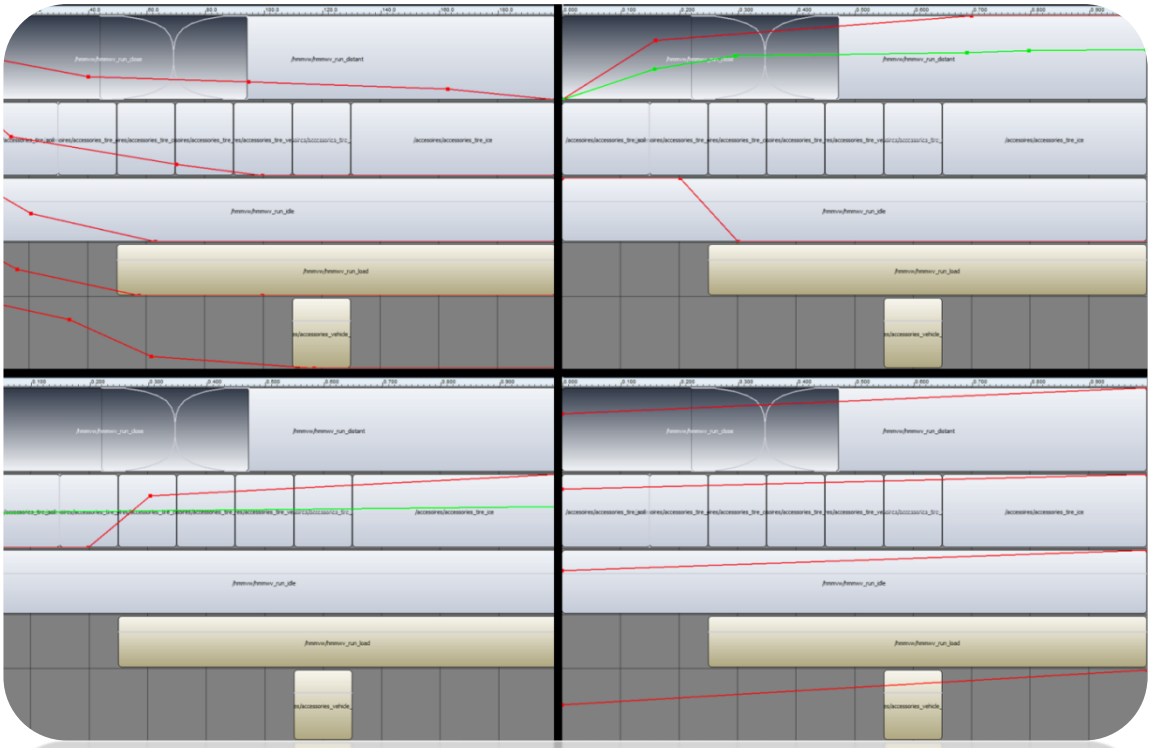


Abbildung 16: Der *run* Sound Event und dessen Volume- (rot) und Pitch- (grün) Kurven für die Parameter: *distance, rpm_scale, speed, in_out, surface* (von oben links nach unten rechts)

Weitere wichtige Sound Events sind *ambience*, der die Innenraum-Atmo simuliert; *bump_on_road*, der anhand des *intensity* Parameters das Aufsetzen des Fahrzeugs vertont; oder auch *damage* und *scratch*, wobei ersterer mit zunehmenden Schaden am Fahrzeug mittels des Parameters *damage* ein Motorraseln hinzufügt und letzterer bei steigendem Reifenverlust (*scratch* Parameter) ein Kratzgeräusch der Stahlfelgen dazu mischt.

5.9 Das Musik System

Das Musik System der CryENGINE besteht aus drei Teilen.

1. Es kann Musik abspielen, die per LUA, *Flow Graph* oder *Area Shapes* eingepflegt wird.

2. Das Music Logic System kann Musik automatisch aufgrund der Spielweise / Spielaktionen wechseln.
3. Der Musik Editor dient zum Ordnen der Musik und der Kategorisierung nach Stimmung (*Moods*) und Ebenen (*Layer*).

5.9.1 Musik Editor

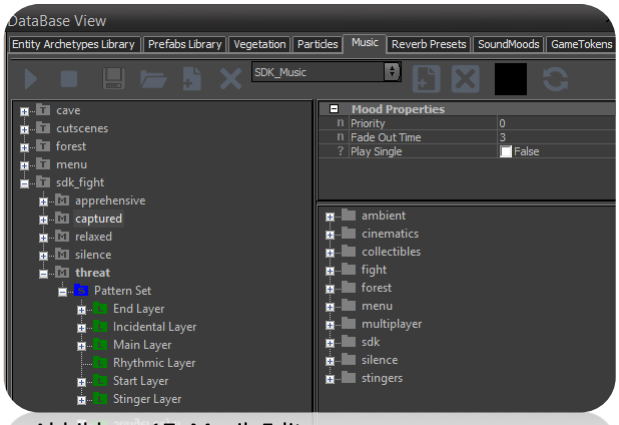


Abbildung 17: Musik Editor

Um Musik in einen Level zu integrieren, muss diese zunächst im Music Editor verwaltet werden. Im Music Editor, der sich in der DataBase Ansicht befindet, können Musik-Themen mit den unterschiedlichen *Moods* (Stimmungen) verwaltet werden. Ein Level kann zum Beispiel mehrere Spielabschnitte besitzen, die jeweils ihr eigenes Musik Thema aufweisen. Innerhalb dieses Abschnitts bzw. Themas können je nach Spielsituation verschieden Musikstimmungen (*Moods*) gespielt werden. Typische *Moods* sind *relaxed* (entspannt), ein *Mood* für entspannte Spielsituationen oder *threat* (bedroht), eine Musikstimmung bei Feindkontakt. Neben den Grundstimmungen: *apprehensive, captured*, *relaxed*, *silence* und *threat,* können auch eigene *Moods* mit individuellem Namen vergeben werden. Allerdings müssen die *Moods* zuvor für die Nutzung mit *Music Logic* im *Animation Graph* definiert werden (vgl. *5.9.3*). Jeder *Mood* besitzt ein *Pattern Set* mit mehreren *Layern*, welche schließlich ein *Pattern* (das Audiofile) enthalten. Die *Layer* enthalten die Teile des Musikstücks und erfüllen folgende Funktionen:

End Layer	Wird nach dem *Main Layer - Pattern* gespielt, um einen sauberen Ausklang zu gewährleisten.
Incidental Layer	Kurze *Patterns*, die per Zufall zusammen mit dem *Main Layer* gespielt werden. Sorgt für Abwechslung der Hintergrundmusik.
Main Layer	Das Hauptpattern bildet den Hauptteil des *Moods* und wird in einer Endlosschleife gespielt (*Loop*). Enthält es mehrere Patterns werden diese per Zufall abwechselnd gespielt.
Rhytmic Layer	*nicht weiter dokumentiert*
Start Layer	Wird vor dem *Main Layer - Pattern* gespielt, um einen sauberen Anfang zu gewährleisten.
Stinger Layer	Ähnlich wie der *Incidental Layer*, die Patterns werden jedoch unabhängig von Sync Punkten abgespielt. Elemente des *Stinger Layer* werden beim Wechsel von einem *Mood* mit niedrigem „Game_Intensity“ (vgl. *5.9.2*) Wert auf eines mit höherem abgespielt.

Für die Themen *Moods*, *Layer* und *Patterns* können jeweils weitere Einstellungen vorgenommen werden. Hier kann beispielsweise die Lautstärke oder die Einblenddauer festgelegt werden. Hat ein *Layer* mehrere Audiofiles (*Patterns*) kann mittels des Parameters *probability* die Abspiel-Wahrscheinlichkeit der einzelnen Elemente bestimmt werden. Um saubere Übergänge zwischen *Moods* zu generieren, sollten für die *Main Layer – Patterns fadepoints* gesetzt werden. Diese bestimmen wann ein *Mood* ein- bzw. ausgeblendet wird. Die Angabe der *fadepoints* erfolgt in *Samples*, das heißt, sie müssen zuvor mittels eines Audio Editor ermittelt werden.

5.9.2 Music Logic

Mit *Music Logic* kann die Musikwiedergabe automatisch durch die Spielweise beeinflusst werden. Spielereignisse, wie beispielsweise Feindsichtung, Schusswechsel oder aber wenn der Spieler von seinem Gegner verletzt wird, erhöhen den sogenannten *Game_Intensity* Wert und lösen beim Erreichen eines definierten Levels eine andere Musikstimmung (*Mood*) aus (vgl. *5.9.1*). Um welchen Wert Spielereignisse diesen Wert erhöhen, kann in der Datei MusicLogic.xml festgelegt werden. Weitere Spielaktionen, die diesen Wert erhöhen, können per Code oder LUA Skript eingefügt werden. Nachfolgend einmal ein Beispiel mit Code (C++) und einmal per Skript (LUA).

```
C++:    SendMusicLogicEvent(eMUSICLOGICEVENT_PLAYER_WOUNDED);
LUA:    MusicLogic.SetEvent(MUSICLOGICEVENT_ENEMY_KILLED);
```

Sobald der Spieler verwundet bzw. ein Gegner getötet wird, steigt der *Game_Intensity* Wert um den in MusicLogic.xml festgelegt Wert an.

```
<MusicLogicEvent Name="EnemyKilled" ChangePlayer="50.0"/> // Erhöhung um 50
<MusicLogicEvent Name="PlayerWounded" ChangePlayer="150.0" /> // Erhöhung um 150
```

Neu erstellte Aktionen- *MusicLogicEvents* genannt- müssen stets in der MusicLogic.xml erfasst und deren *Game_Intensity* Auswirkung definiert werden.

Der *Game_Intensity* Wert kann ebenfalls mittels des *Music:LogicControl Node* im *Flow Graph* verändert werden, indem er entsprechende Werte

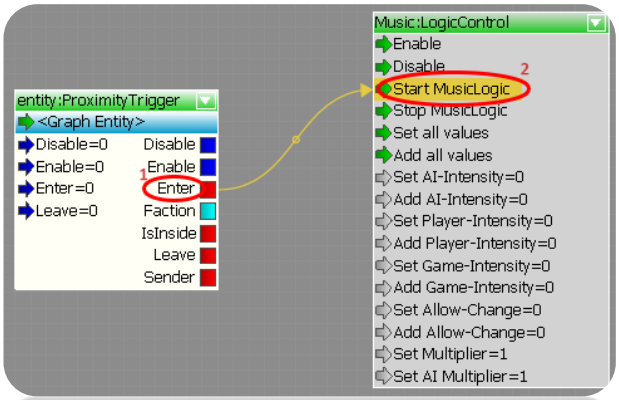


Abbildung 18: Beim Betreten des Triggers (1) wird der Input *„Start MusicLogic“* des *Music:LogicControl Flow Graph* ausgelöst (2) und *Music Logic* aktiviert.

über einen Trigger (zum Beispiel beim Betreten oder Verlassen eines Raums) übermittelt bekommt.

Um das adaptive Musik System nutzen zu können, muss zuerst *Music Logic* aktiviert werden. Dies geschieht entweder über einen *Flow Graph*, wie Abbildung *20* zeigt, oder per LUA-Skript, wie nachfolgend dargestellt:

```
MusicLogic.StartLogic(); // aktiviert MusicLogic
MusicLogic.StopLogic();  // deaktiviert MusicLogic
```

5.9.3 AnimationGraph

Der *Animation Graph Editor* - ehemals als *Music Logic Graph* bezeichnet - ist primär für das Auswählen und Starten von Animationen während des Spiels zuständig. Mit diesem Editor wird festgelegt, welche Animation wann abgespielt wird und welche Übergänge zwischen den einzelnen Animationen gezeigt werden sollen. Zum Beispiel wird beim Wechsel von einer Sitz-Animation zu einer Stand-Animation eine Aufsteh-Animation eingefügt.

Der *Animation Graph Editor* ist ebenfalls für das Verhalten der Musik bei aktiviertem *Music Logic* zuständig. Hierzu werden den einzelnen *Moods* minimal und maximal *Game_Intensity* Werte zugeteilt, bei denen das jeweils passende *Mood* abgespielt werden soll. Außerdem kann zum einen eine minimale und zum anderen eine maximale Abspieldauer der *Moods* festgelegt werden. Wird im Spiel ein für einen *Mood* definierter *Game_Intensity* Grenzwert erreicht, werden der *Mood* und dadurch die Musik gewechselt. Im Beispielvideo *25* wechselt die Musik automatisch bei Feindkontakt.

5.9.4 Einbindung von Musik

Neben der Verwendung von *Music Logic*, der die Musikwiedergabe größtenteils automatisiert, kann auch eine Einbindung über den *Flow Graph Editor*, LUA Skripts oder *Area Shapes* erfolgen. Oftmals werden die verschiedenen Möglichkeiten kombiniert, um ein authentisches Ergebnis zu erzielen. Allerdings sollte *Music Logic* dabei nicht gleichzeitig mit anderen Musikeinbindungen verwendet werden, da es sonst zu Konflikten kommen kann. Daher sollte es zuvor deaktiviert werden. Somit kann die Music beispielsweise zu Beginn eines Levels an bestimmten Checkpoints getriggert werden und beim Betreten eines Kampfgebiets wird *Music Logic* aktiviert.

5.9.4.1 Beispiel einer Umsetzung mittels Area Shape

Im hier angeführten Beispiel soll die Musik beim Betreten einer Hütte die Stimmung wechseln. Eine *Area Shape*, die mit einem *MusicThemeSelector Entity* verknüpft ist, umgibt die Hütte. Das gewünschte Thema und dessen *Mood* müssen zuerst in den Eigenschaften des *MusicThemeSelector* eingestellt werden. In diesem Fall ist es das *sdk_fight* Thema mit dem *Mood relaxed*. Die Hütte ist ebenfalls mit einer *Area Shape* abgesteckt und ist mit *einem MusicMoodSelector Entity* verknüpft, das auf das *Mood apprehensive* wechseln soll. Beim Betreten der äußeren *Area Shape* wird das *sdk_fight* Theme mit dem *relaxed* Mood abgespielt. Das Ergebnis besteht nun darin, dass sobald die Hütte betreten wird der *Mood* auf *apprehensive* wechselt und die Musik in das entsprechende *Pattern* überblendet wird.

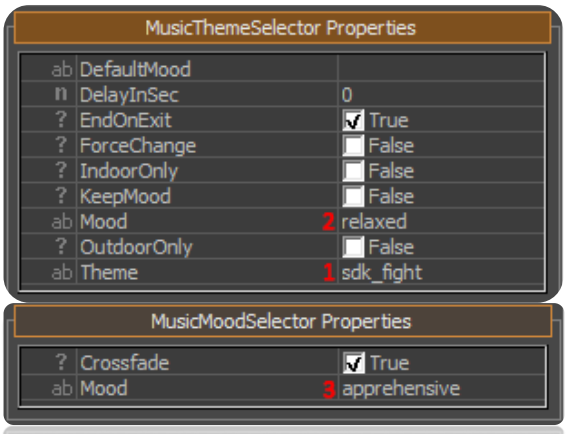


Abbildung 19: Das gewählte Theme *sdk_fight* (1) und der Mood *relaxed* (2)für den Außenbereich und der Stimmungswechsel für das Innere der Hütte. (3)

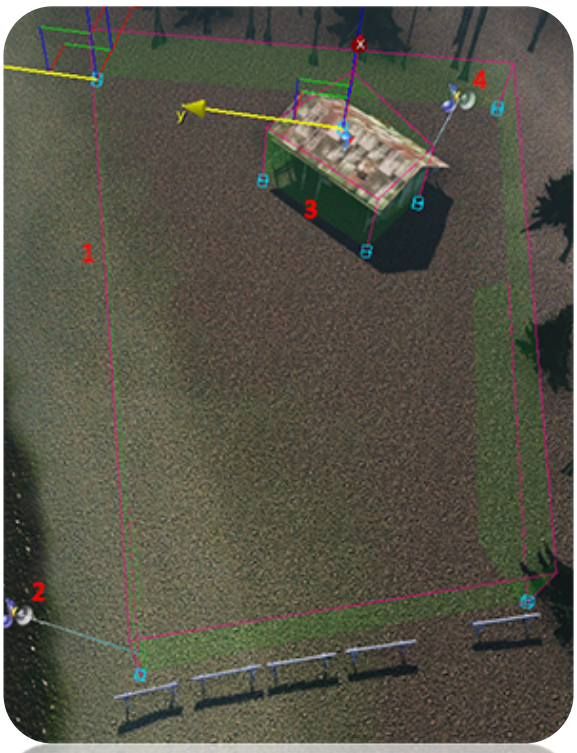


Abbildung 20: Area Shape des Außenbereichs (1) mit verknüpftem MusicThemeSelector (2) und Area Shape der Hütte (3) mit dem MusicMoodSelector (4)

5.9.4.2 Beispiel einer Umsetzung mittels Flow Graph

In diesem vereinfachten Beispiel wird beim Spielstart das *sdk_fight* Thema mit dem *relaxed Mood* gestartet. Beim Betreten einer festgelegten Fläche wird das *Mood* in *threat* gewechselt und beim Verlassen wieder zurück in *relaxed*. Um das Spiel mit dem sdk_fight Theme zu starten, muss der Ausgang (*output*) des *Start Node* mit dem *Play* Eingang des Music:ThemeSelector Node verbunden wer-

den. Thema und Stimmung werden zuvor im *Music:ThemeSelector Node* definiert. Anschließend gilt es die Fläche mit einem *Proximity Trigger* zu versehen. Dieser beinhaltet die notwendigen Ausgänge für das Betreten (*Enter*) und Verlassen (*Leave*) der Fläche. Die Stimmung (*Mood*) wird, je nachdem ob die Fläche nun Betreten bzw. Verlassen wird, mittels der *Music:MoodSelector Nodes* entsprechend geändert.

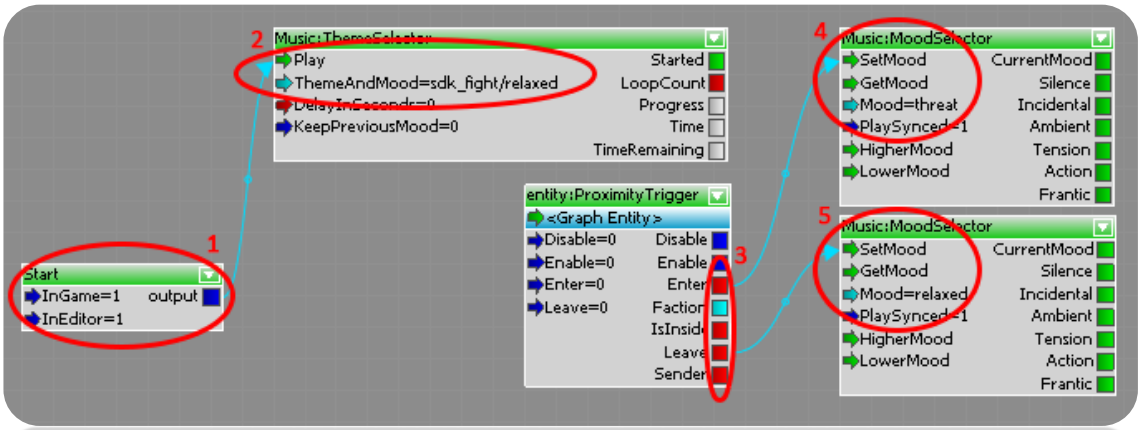


Abbildung 21: *Start Node* (1), *Music:ThemeSelector Node* (2), *Proximity Trigger* mit *Enter* bzw. *Leave Trigger* (3), *Music:MoodSelector* (4&5)

6 Fazit

Zu Beginn der Arbeit konnte dargelegt werden, dass das Computerspiel inzwischen zu Recht als bedeutsames Unterhaltungsmedium bezeichnet wird. Der Unterhaltungswert steht in unserer heutigen „Spaßgesellschaft“ beim Spielen im Vordergrund (Früh, 2002). Das Ziel dabei ist es, ein möglichst authentisches Spielerleben zu kreieren, so dass ein Eintauchen in die virtuelle Welt erfolgt. Die Bedeutung des Präsenzerlebens, des *Flow*-Erlebens und des kontrollierten Kontrollverlusts, die zur Identifikation mit dem Spiel bzw. der Spielfigur führen, wurde anfangs in der Arbeit ausführlich beschrieben.

Das vollkommene Eintauschen in die Spielewelt erfolgt jedoch nur dann, wenn alle Sinnesreize des Spielers angesprochen werden. Hierzu zählt auch die akustische Wahrnehmung des Spielers (Wünsch & Jenderek, 2009). Dies kann allerdings nur durch ein stimmiges Sound Design gelingen (Collin, 2008; Leenders, 2012).

Das Spiele-Genre der FPS zeichnet sich – im Gegensatz zu anderen Computerspielen – durch seine Ego-Perspektive aus. Diese Perspektive verstärkt wiederum das Eintauchen in die Spielewelt:

> „It may be stated that player immersion in the First-Person Shooter (FPS) game is one of the goals of the FPS developer. This being mental immersion […], the goal may be defined as the player’s perception that he is within the game environment, that he is the character whose hands he sees before him. Player immersion, then, may be supposed to be primarily perceptual and is manifested by a shift of perceptual focus, from an awareness of ‘being in and part of’ reality to ‘being in and part of’ virtuality such that, in the ideal case, virtuality becomes substituted for reality“ (Grimshaw, 2008: 2).

Hierbei kommt dem Sound und Sound Design eine enorme Bedeutung zu. Allerdings liegt der Fokus in der gängigen Fachliteratur, aber auch im Marketing von Computerspielen, häufig auf den visuellen Aspekten eines Spiels. Grimshaw (2008) ist der Ansicht, dass die Wichtigkeit von Sound in FPS noch immer unterschätzt wird:

> „Unlike graphics, sound rarely rates a direct mention in digital games marketing material – claims for graphical superiority over other games can be made with images (often rendered with a detail and realism that are never replicated in the game). The game box and packaging are incapable of providing a similar siren call for sound. However, it is my argument that sound too plays a part in the perceptual realism of the game that forms a basis for player immersion. In this, my argument parallels claims made for sound in other media.[…] my contention is that sound is of great importance, if not the greatest

> importance, in creating the perceptual realism of the FPS game that leads to player immersion“ (Grimshaw, 2008: 2ff.).

Insgesamt konnte im Verlauf der Arbeit aufgezeigt werden, dass die Ton- und Musikgestaltung im Rahmen von Computerspielen von großer Bedeutung für das Spielerleben und den Unterhaltungswert ist, so dass jedes einzelne Spiel ganz individuelle Anforderungen an die Arbeit des Sound Designers stellt (Leenders, 2012).

Die Besonderheiten des Sound Designs in Computerspielen ergeben sich dabei vor allem aus der Interaktivität und Nonlinearität des Mediums. Interaktivität bedeutet, dass der Spieler selbst aktiv agieren kann (ebd.). Durch die Interaktivität des Mediums kommt es zu einer nicht-linearen Struktur des Spiels und der Sound Designer muss eine Vielzahl von akustischen Entscheidungsmöglichkeiten vorhersehen und zuvor ausprobieren. Dies führt zu einem „dynamischen Ton“, das heißt, zur interaktiven und adaptiven Tonarbeit (Collins, 2008; Leenders, 2012).

In den Praxisbeispielen wurde die dynamische Tonarbeit mit Hilfe der CryENGINE 3 umgesetzt und anhand der Waffensounds, den materialabhängigen Sounds und den Fahrzeugsounds, sowie anhand der Erstellung von Atmos demonstriert. Insgesamt konnte mit Hilfe der verschiedenen Sound-Beispielen gezeigt werden, dass sich der Ton immer auf die Eingaben des Spielers und an die Variablen der Umgebung anpassen muss – und dies in einer sehr ausgeprägten Form, da die Besonderheit der FPS in ihrer starken Dynamik aufgrund der Bewegungsvielfalt und des hohen Umfangs an Interaktionsmöglichkeiten des Spielers liegt.

Alles in allem konnte durch die Arbeit aufgezeigt werden, dass durch die rasante Entwicklung der Computerspielbranche und aufgrund des ständigen technischen Fortschritts, die Arbeit des Sound Designers im Rahmen der Spieleproduktion immer mehr an Bedeutung gewinnt – auch wenn bisher nur wenige Publikationen in der gängigen Fachliteratur hierzu vorherrschen. Abschließend muss somit festgehalten werden:

> „Ton und Musik tragen entscheidend zu diesem Spielerlebnis bei, gerade auch wegen der sehr direkten psychologischen Einflussname, die akustische Reize auf den Konsumenten ausüben. Speziell emotionale Momente werden für Zuschauer und Spieler in visuellen Medien erst durch eine gut komponierte und platzierte Musik sowie gezieltes Sounddesign richtig erfahrbar“(Leenders, 2012: 66).

7 Literaturverzeichnis

Bareither, Christoph (2012). *Ego-Shooter Spielkultur. Eine Online-Ethnographie*. Tübingen: Tübinger Verein für Volkskunde Verlag.

Bartsch, Anne; Eder, Jens & Fahlenbrach, Edith (2007). Einleitung: Emotionsdarstellung und Emotionsvermittlung durch audiovisuelle Medien. In: Bartsch, Anne; Eder, Jens & Fahlenbrach, Edith (Hrsg.), *Audiovisuelle Emotionen. Emotionsdarstellung und Emotionsvermittlung durch audiovisuelle Medienangebote*. Köln: Halem Verlag, 8-38.

Collins, Karen (2008): *Game Sound*. Cambridge, MA, USA: MIT Press.

DUDEN (1973) Rechtschreibung. 17., neubearbeite. und erweiterte. Auflage Mannheim, S. 639.

Elson, Malte (2011). *The Effects of Displayed Violence and Game Speed in First-Person Shooters on Physiological Arousal and Aggressive Behavior*. Masterarbeit, Universität zu Köln (*kups.ub.uni-koeln.de/4291/* Zugriff am 22.02.2013).

Flückiger, Barbara (2010). *Sound Design. Die virtuelle Klangwelt des Films*. Marburg: Schüren Verlag GmbH.

Fritz, Jürgen & Fehr, Wolfgang (2003). Virtuelle Gewalt. Modell oder Spiegel? In: Dies. (Hg.): *Computerspiele. Virtuelle Spiel- und Lebenswelten*. Bonn: Bundeszentrale für politische Bildung, 49-60.

Früh, Werner (2002): *Unterhaltung durch das Fernsehen. Eine molare Theorie*. Konstanz: UVK.

Früh, Werner & Wünsch, Carsten (2007). Die Makroemotion ›Unterhaltung‹ in der Triadisch-Dynamischen Unterhaltungstheorie (TDU). In: Bartsch, Anne; Eder, Jens & Fahlenbrach, Edith (Hrsg.), *Audiovisuelle Emotionen. Emotionsdarstellung und Emotionsvermittlung durch audiovisuelle Medienangebote*. Köln: Halem Verlag,187-204.

Esser, Hartmut (1986). Können Befragte Lügen? Zum Konzept des „wahren Wertes“ im Rahmen der handlungstheoretischen Erklärung von Situationseinflüssen bei der Befragung. *Kölner Zeitschrift für Soziologie und Sozialpsychologie* 38, 314-336.

Grunwald, Stephan (2008). *Sound als Feedbackmöglichkeit in Computerspielen: Eine Analyse am Beispiel von Echtzeit –Strategie und First-Person-Shooter*. München: Grin Verlag.

Grimshaw, Mark (2008). Sound and immersion in the first-person shooter. *Games Computing and Creative Technologies: Journal Articles* (Peer-Reviewed). Paper 3. (*http://ubir.bolton.ac.uk/index.php?action=fileDownload&resourceId=255&hash=8dfb5f736260d248f7bdcb0397da3c91c24ecd24&filename=gcct_journalspr-3.pdf*, Zugriff am 22.02.2013).

Günzel, Stephan (2012). *Egoshooter: Das Raumbild des Computerspiels*. Frankfurt am Main: Campus Verlag.

Herrmann, Marcus (2009). *Psychoakustik und Sound-Engineering*. München: Grin Verlag.

Klimmt, Christoph (2001). Ego-Shooter, Prügelspiel, Sportsimulation? Zur Typologisierung von Computer- und Videospielen. *Medien- und Kommunikationswissenschaft*, Vol. 49 (4), 480-497.

Klimmt, Christoph (2009). Die Nutzung von Computerspielen: Interdisziplinäre Perspektive. In T. Quandt,T. Wimmer, J. & Wolling, J. (Hrsg.), *Die Computerspieler - Studien zur Nutzung von Computergames*. Wiesbaden: Verlag für Sozialwissenschaften, 57-72.

Leenders, Matts Johan (2012). *Sound für Videospiele. Besondere Kriterien und Techniken bei der Ton- und Musikproduktion für Computer-und Videospiele*. Marburg: Schüren Verlag GmbH.

Lehmann, Philipp; Reiter, Andreas; Schumann, Christina & Wolling, Jens (2009). Die First-Person-Shooter. Wie Lebensstil und Nutzungsmotive die Spielweise beeinflussen. In T. Quandt,T. Wimmer, J. & Wolling, J. (Hrsg.), *Die Computerspieler - Studien zur Nutzung von Computergames*. Wiesbaden: Verlag für Sozialwissenschaften, 241-262.

Lensing, Jörg U. (2009). *Sound-Design. Sound-Montage. Soundtrack-Komposition. Über die Gestaltung des Filmtons*. Berlin: Schiele & Schön.

Marks, Aaron (2001). *The Complete Guide To Game Audio. For Composers, Musicians, Sound Designers, and Game Developers*. Lawrence, Kansas: CMP Books.

Marks, Aaron & Novak, Jeannie (2009). Game Development Essentials: Game Audio Development. Clifton Park, NY, USA: Delmar, Cengage Learning.

Möller, Ingrid (2007). Emotionen beim Konsum von Bildschirmspielen. merz. *medien + erziehung*. 51. Jahrgang, 4/07. München, 31-37.

Möller, Ingrid (2003). *Mediengewalt und Aggression. Eine längsschnittliche Betrachtung des Zusammenhangs am Beispiel des Konsums gewalthaltiger Bildschirmspiele*. Dissertation: (*opus.kobv.de/ubp/volltexte/2006/773/pdf/moeller_diss.pdf*, Zugriff am 22.02.2013).

Raffaseder, Hannes (2010). *Audiodesign*. München: Carl Hanser Verlag.

Rudolph, Axel (1993). Akustik Design - Gestaltung der akustischen Umwelt. *Europäische Hochschulschriften*, Reihe 5, Band 1416 Frankfurt/Main: Lang

Schätzlein, Frank (2005). Sound und Sounddesign in Medien und Forschung. In: Segeberg, Harro & Schätzlein, Frank (Hrsg.), *Sound. Zur Technologie und Ästhetik des Akustischen in den Medien*. Marburg: Schüren (*www.frank-schaetzlein.de/texte/sound_gfm2005.pdf*, Zugriff am 20.02.2013).

Thimm, Caja (2010). Spiel – Gesellschaft – Medien: Perspektiven auf ein vielfältiges Forschungsfeld. In: Thimm, Caja (Hg.), *Das Spiel: Muster und Metapher der Mediengesellschaft*. Wiesbaden: VS Verlag für Sozialwissenschaften, 7-13.

Tracy, Sean & Reindell, Paul (2012). *CryENGINE 3 Game Development Beginner`s Guide*. Brimingham: Packt Publishing Ltd.

Wolf, Mark J.P. (2001). *The Medium of the Video Game*. University of Texas Press.

Wünsch, Carsten & Jenderek, Bastian (2009). Computerspielen als Unterhaltung. In T. Quandt,T. Wimmer, J. & Wolling, J. (Hrsg.), *Die Computerspieler - Studien zur Nutzung von Computergames*. Wiesbaden: Verlag für Sozialwissenschaften, 41-56.

Internetquellen:

Audacity Manual
audacity.sourceforge.net/manual-1.2/effects_reverb.html (Zugriff am 12.01.2013).

Börsing, Christian (2009). Aspekte des Sounddesigns.
www.mediadesign.de/sites/default/files/Dateien/PDFs/MDH-Fokus0209.pdf (Zugriff am 06.01.2013).

BIU - Bundesverband Interaktive Unterhaltungssoftware e. V.
www.biu-online.de/de/fakten/marktzahlen.html (Zugriff am 20.02.2013).
www.biu-online.de/de/fakten/gamer-statistiken.html (Zugriff am 20.02.2013).

Clark, Andrew (2007): Defining Adaptive Music
www.gamasutra.com/view/feature/1567/defining_adaptive_music.php (Zugriff am 26.02.2013).

Codeworx.org
www.codeworx.org/opengl_par1.php (Zugriff am 29.01.2013).

Crytek Firmenhomepage
www.crytek.com/company (Zugriff am 08.01.2013).

Deutscher Computerspielpreis
http://www.deutscher-computerspielpreis.de/164.0.html (Zugriff am 22.12.2012).

Erhebung des Instituts für Demoskopie Allensbach zur Mediennutzung:
de.statista.com/statistik/daten/studie/171129/umfrage/haeufigkeit-des-computerspielens-in-der-freizeit/ (Zugriff am 18.02.2013).

ETC Internal Unity3D Wiki
wiki.etc.cmu.edu/unity3d/index.php/3D_Sound (Zugriff am 05.02.2013).

FMOD Designer 2010 user manual 2.0
http://www.stephanschutze.com/uploads/3/1/0/6/3106267/fmod_designer_user_manualv2.0_ai_march16.docx (Zugriff am 19.12.2012).

Forum für Mikrofonaufnahmetechnik und Tonstudiotechnik
www.sengpielaudio.com/Rechner-RT60.htm (Zugriff am 12.01.2013).
www.sengpielaudio.com/AnfangszeitlueckeUndPredelay.pdf (Zugriff am 13.01.2013).

Infoplattform Wissenswertes.at
www.wissenswertes.at/index.php?id=spiele-aaa (Zugriff am 08.01.2013).

Online Manual des SDK der CryENGINE 3
freesdk.crydev.net/display/SDKDOC2/ (Zugriff am 20.12.2012).

pcgames.de - Wissen, was gespielt wird!
www.pcgames.de/Panorama-Thema-233992/Specials/Die-15-meistverkauften-PC-Spiele-aller-Zeiten-und-ihre-Verkaufszahlen-PCG-Top-Artikel-Mai-2010-747800/ (Zugriff am 20.02.2013).

retrosynth.com
www.retrosynth.com/~analoguediehard/studio/effects/digitalreverb.html (Zugriff am 14.01.2013).

SAE Institute
www.sae.edu/reference_material/audio/pages/Reverb.htm (Zugriff am 12.01.2013).

Sound on Sound
www.soundonsound.com/sos/may00/articles/reverb.htm?print=yes (Zugriff am 14.01.2013).

The DarkMod Wiki
wiki.thedarkmod.com/index.php?title=Setting_Reverb_Data_of_Rooms_%28EAX%29 (Zugriff am 12.01.2013).

8 Anhang

DVD-ROM mit Beispielvideos und PDF-Version der Abschlussarbeit.

Beispielvideo Links

01 **Audio Spezial-Effekt** *Obsctruction*	*15* **Waffensound** *Dauerfeuer*
02 **Audio Spezial-Effekt** *Pitch*	*16* **Waffensound** *Nachladen*
03 **Audio Spezial-Effekt** *Doppler-Effekt*	*17* **Materialabhängige Sounds** *Einschüsse*
04 **Audio Spezial-Effekt** *HDR Audio*	*18* **Animation** *Schrittgeräusche*
05 **Reverb** *Environment Parameter*	*19* **Fahrzeugsound** *Ebene 1*
06 **Reverb** *Preset Huette*	*20* **Fahrzeugsound** *Ebene 2*
07 **Atmo** *spread Parameter*	*21* **Fahrzeugsound** *Ebene 3*
08 **Atmo** *battle Parameter*	*22* **Fahrzeugsound** *Ebene 4*
09 **Atmo** *daylight Parameter*	*23* **Fahrzeugsound** *Ebene 5*
10 **Sound Mood** *low health*	*24* **Fahrzeugsound** *Fahrt*
11 **Sound Mood** *focus_music*	*25* **Musik Music** *Logic*
12 **Sound Mood** *in_vehicle*	*26* **Musik** *Area*
13 **Partikel Effekt**	*27* **Musik** *Flow Graph*
14 **Waffensound** *Einzelschuss*	