Das Gamepad als Musikinstrument – Konzeption und Umsetzung verschiedener Bedienkonzepte

Masterarbeit

im Studiengang
Audiovisuelle Medien

vorgelegt von

Şahin Kablan
Matrikelnummer: 34953

am 23.12.2019
an der Hochschule der Medien Stuttgart

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

Ehrenwörtliche Erklärung

„Hiermit versichere ich, Şahin Kablan, ehrenwörtlich, dass ich die vorliegende Masterarbeit mit dem Titel: „Das Gamepad als Musikinstrument – Konzeption und Umsetzung verschiedener Bedienkonzepte“ selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn 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 ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§ 23 Abs. 2 Master-SPO (3 Semester) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.“

Stuttgart, 23.12.2019

_____

Unterschrift

Kurzfassung

Die vorliegende Masterthesis untersucht den Einsatz von herkömmlichen Gamepads als Musikinstrument. Es wurden neue Bedienkonzepte erarbeitet, die es erlauben, Gamepads als Musikinstrument zu nutzen. Diese wurden im Rahmen des musikalischen Rätselspiels *Sonority* umgesetzt, das die Eingabe von Tönen als Spielmechanik benutzt, um damit Rätsel zu lösen.

Dazu werden zunächst die Grundlagen für Musik, Musikinstrumente und Gamepads dargelegt, um dann drei bereits existierende Spiele auf diese spezielle Nutzung von Gamepads zu untersuchen. Da Musikmachen vordergründig eine kreative Tätigkeit ist, wurde ein besonderes Augenmerk auf die in diesen Spielen gegebenen Freiheitsgrade gelegt. Vor allem das Spiel *Rocksmith* sticht hervor, in dem es Musik in ihrer Reinform bietet, dafür jedoch nicht mit einem herkömmlichen Gamepad spielbar ist.

Für das Spiel Sonority, das sich zum Zeitpunkt dieser Arbeit in Entwicklung befindet, wurden Usertests durchgeführt, bei denen unter anderem die aktuelle Steuerung des Spiels evaluiert wurde. Diese wurde mehrheitlich für weniger intuitiv empfunden in Bezug auf das Gefühl beim Eingeben der Töne. Auf Basis dieser Ergebnisse wurden alternative Konzepte zur Steuerung erarbeitet und in prototypischen Testszenarien ausprobiert.

Schlagwörter: Musik, Musikinstrument, Gamepad, Videospiele, Rocksmith, Guitar Hero, Sonority

Abstract

This master thesis investigates the use of conventional gamepads as a musical instrument. New operating concepts have been developed that allow gamepads to be used as musical instruments. These were implemented in the musical puzzle game *Sonority*, which uses the input of sounds as game mechanics to solve puzzles.

First the basics of music, musical instruments and gamepads will be explained and then three already existing games will be examined for this special use of gamepads. Since making music is primarily a creative activity, special attention was paid to the degrees of freedom given in these games. Especially the game Rocksmith stands out in that it offers music in its pure form but cannot be played with a conventional gamepad.

For the game Sonority, which is in development at the time of this work, user tests were carried out, in which, among other things, the current control of the game was evaluated. This was mostly felt to be less intuitive in terms of the feeling when entering the sounds. Based on these results, alternative concepts for control were developed and tested in prototype test scenarios.

Keywords: music, musical instrument, videogames, gamepad, Guitar Hero, Rocksmith, Sonority

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

bpm	beats per minute
D-Pad	Directional Pad (Steuerkreuz)
EDM	Electronic Dance Music
HdM	Hochschule der Medien
Hz	Hertz
LB	Left Bumper (linke Schultertaste)
LT	Left Trigger
MDA	Mechanics, Dynamics, Aesthetics
MIDI	Musical Instrument Digital Interface
MPTK	Midi Player Tool Kit
NES	Nintendo Entertainment System
o. D.	ohne Datum
PC	Personal Computer
RB	Right Bumper
RT	Right Trigger
USB	Universal Serial Bus

1 Einführung

Eine Welt ohne Musik ist kaum vorstellbar. Archäologische Funde von ungefähr 50.000 Jahre alten prähistorischen Musikinstrumenten legen nahe, dass Musik von Anfang an Teil der menschlichen Kultur war (Spitzer, 2002, S. 1). Neben der menschlichen Stimme, die vermutlich als das erste Instrument überhaupt gilt, haben sich mit der Zeit eine Reihe von unterschiedlichsten Instrumenten herausgebildet – mitunter werden auch unscheinbare Gegenstände als Musikinstrument benutzt. Mit jeder neuen technologischen Errungenschaft entstanden auch neue Arten, Töne und Klänge zu erzeugen (Wolf, 2018, S.340).

So wie Musik heute aus der Welt nicht mehr wegzudenken ist, sind Videospiele mit ihren Möglichkeiten der Interaktion mittlerweile auch Teil unserer Gesellschaft geworden. Für das Jahr 2020 wird für den deutschen Markt im Bereich von Konsolen- und PC-Spielen sowie Casual Games ein Umsatz von ca. 4,5 Milliarden Euro erwartet (Statistisches Bundesamt, zitiert nach de.statista.com, 2019). Videospiele dienen jedoch nicht nur dem reinen Spielspaß, sondern werden auch vermehrt im Kontext von Edutainment eingesetzt – also Spiele, die beim Spielen zusätzlich Wissen vermitteln und somit eine größere bildungsrelevante Komponente aufweisen.

Wie auch in Filmen spielt Musik oft eine große Rolle in Videospielen, findet allerdings meistens im Hintergrund statt, um die Rezeption des Spielgeschehens zu unterstützen. Die Gattung jener Spiele, die Musik darüber hinaus zum Gameplay-Element machen, ist eher eine kleine Nische in der Welt der Videospiele. Hier muss Spieler*innen die Möglichkeit gegeben werden, Töne über ein herkömmliches Gamepad einzugeben.

Die vorliegende Arbeit untersucht die Einsatzmöglichkeiten von Musik in Videospielen im Zusammenhang mit der Eingabe von einzelnen Tönen. Zunächst werden die Grundlagen von Musik und Musikinstrumenten dargelegt; ebenso wird ein Überblick über die Geschichte von Gamepads und ihrer Funktionsweise gegeben. Außerdem wird das so genannte MDA-Framework erläutert, das als hilfreiches Werkzeug in der Beurteilung von Games dient. Auf Basis dieser Grundlagen werden insgesamt drei bereits veröffentlichte Spiele aus unterschiedlichen Genres untersucht, die Musik als prominentes Gameplay-Element nutzen. Es wird dabei auch ein Augenmerk auf die vom Spiel gegebenen Freiheitsgrade gelegt.

Die Erkenntnisse aus diesen Untersuchungen werden genutzt, um im Rahmen des derzeit entstehenden Prototyps für das musikalische Rätselspiel *Sonority* neuartige Steuerungen und Eingabemöglichkeiten zu konzipieren und prototypisch umzusetzen. Das Konzept zu Sonority wurde 2019 mit dem Deutschen Computerspielepreis in der

Kategorie *Nachwuchskonzept* mit dem zweiten Platz ausgezeichnet (Deutscher Computerspielepreis, 2019).

Zunächst werden die Spielmechaniken des Spiels erläutert und die bisherige Steuerung untersucht. Im Zuge der Entwicklung des Prototyps wurde bereits eine Evaluation durchgeführt, worin auch die Perzeption der Steuerung bewertet wurde. Auf Basis dessen werden alternative Konzepte dargelegt und in prototypischer Manier umgesetzt.

2 Grundlagen

Musik ist eine Kunstgattung, die aus Elementen der Welt des Hörbaren besteht und somit nicht fassbar oder materiell ist – zu „sehen“ ist Musik z.B. als Notenabschrift, jedoch ist sie allein dadurch nicht erlebbar wie beispielsweise ein Gemälde (Moser, 2013, S. 5). In diesem Kapitel wird zunächst ein Überblick über Töne, Klänge und ihre Beziehungen zueinander gegeben; eine grobe Zusammenfassung der Musiktheorie wird ebenfalls dargelegt. Anschließend soll die Bedeutung von Musikinstrumenten erklärt werden. Um die Grundlagen für diese Arbeit zu vervollständigen, werden zum Schluss verschiedene Gamepads und ihre Funktionsweise beleuchtet.

2.1 Regeln der Musik

Zunächst gilt es, gewisse Begrifflichkeiten und Regeln innerhalb der Musik zu klären und gegebenenfalls voneinander abzugrenzen. Um musizieren (erlebbar/wahrnehmbar machen) zu können, müssen Töne in einen bestimmten Zusammenhang gebracht werden, der für jede Kultur der Welt unterschiedlich ist. Jedoch fußen alle Interpretationen von Musik auf physikalischen Gesetzmäßigkeiten, die dazu dienen, Töne einer für den Menschen sinnvollen Systematik unterzuordnen. (Moser, 2013, S. 5)

2.1.1 Töne und Geräusche

Unsere Ohren nehmen Schall meist über das Medium Luft wahr. Dieser überträgt sich darin mit einer Geschwindigkeit von 343 Meter pro Sekunde in Form von Veränderung des Luftdrucks (Spitzer, 2002, S. 23). Gelangen gleichmäßige (periodische) Schwingungen an das Trommelfell im Ohr, nehmen Menschen das als Ton wahr. Deshalb wird auch beispielsweise der laufende Rotor eines Hubschraubers oder Segelflugzeugs mitunter als Ton wahrgenommen. Je höher dabei die Frequenz der Schwingung ist, desto höher klingt der Ton. Physikalisch wird die Frequenz als Anzahl der Schwingungen pro Sekunde beschrieben, gemessen in der Einheit *Hertz* (Hz). Die Höhe des Schwingungsausschlag wird Amplitude genannt. Der simpelste Ton ist die Sinusschwingung, dessen Namen vom bekannten Kurvenverlauf abgeleitet wird, wie in Abbildung 1 zu sehen ist.

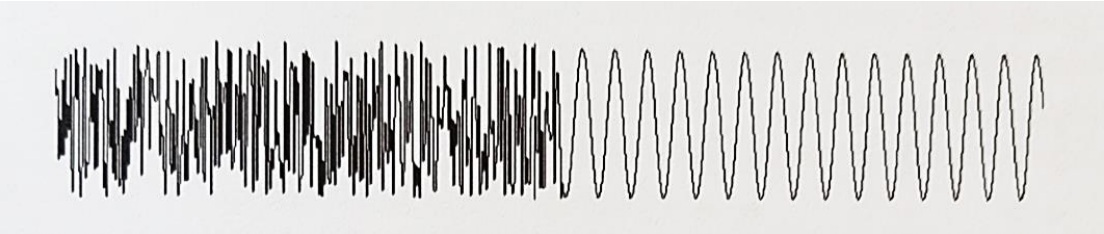

Abbildung 1: Rauschen und Ton (Spitzer, 2002, S.29)

Grundsätzlich wird zwischen Tönen und Rauschen unterschieden. Rauschen – oder Geräusche – zeichnet sich dadurch aus, dass sich die Luftteilchen nicht gleichmäßig, sondern völlig zufällig bewegen (Spitzer, 2002, S. 27). Abbildung 1 zeigt zunächst einen ungeordneten Schwingungsverlauf beim Rauschen (links) und anschließend eine gleichmäßige Sinusschwingung (rechts).

Reine Sinustöne kommen selten in der Natur vor und haben auch keine besondere Klangfarbe. Der Klangcharakter eines Tons wird durch die so genannten Obertöne des Grundtons definiert, die jeweils die ganzzahligen Vielfache des Grundtons sind; wenn die Frequenz eines Grundtons also als *f* bezeichnet wird, wäre 2*f* die doppelte Frequenz, 3*f* die dreifache etc. (Dickreiter, 1977, S.22).

Musikinstrumente klingen deshalb so unterschiedlich, weil zur Grundtonfrequenz des Instruments jeweils spezifische Vielfache derselben addiert werden beziehungsweise mitschwingen. Doch nicht nur die Obertöne entscheiden über Klang, sondern auch die zeitliche Ausdehnung des Tons – ob sie langsam oder plötzlich und schnell anklingen, oder nach dem Anschlag immer lauter oder leiser werden. Dieses Verhalten gilt auch für die Obertöne eines Tons (Spitzer, 2002, S. 39). Der Verlauf der Lautstärke eines Tons von Anfang bis Ende des Erklingens wird durch die so genannte *Hüllkurve* beschrieben. In Kapitel 2.2.3 wird im Zusammenhang von Tonerzeugung bei Musikinstrumenten näher darauf eingegangen.

2.1.2 Beziehungen zwischen Tönen

Ein *Intervall* beschreibt den Frequenzunterschied zwischen zwei Tönen. Das einfachste Intervall ist die *Oktave*, die bereits weiter oben als Verdopplung der Frequenz eines Tons beschrieben wurde – also dem ersten Oberton des Ausgangstons entspricht. Da die Obertöne beider Töne deckungsgleich sind, klingen diese auch gleich, mit dem einzigen Unterschied, dass der zweite Ton höher klingt. Wenn beide Töne zusammen gleichzeitig gehört werden, klingen sie außerdem aufgrund der gleichen Obertonreihe sehr klar (Spitzer, 2002, S. 82).

Ein Ton, der im Vergleich zum Ausgangston diesmal anders klingt, ist die so genannte *Quinte*, die dem nächsten Oberton entspricht. Die Quinte (Verhältnis 3:2) beschreibt dabei den Abstand vom ersten Oberton (220 Hz) zum zweiten Oberton (330Hz). Wenn der Ausgangston beispielsweise eine Frequenz *f* von 110 Hz hat, entspricht die doppelte Frequenz (2*f* = 220 Hz) der Oktave und die dreifache Frequenz (3*f* = 330 Hz) der Quinte. Führt man das fort, erhält man für *f* = 440 Hz die doppelte Oktave. Der Abstand vom zweiten Oberton (330 Hz) zum dritten Oberton (440 Hz) wird als *Quarte* (Verhältnis 4:3) bezeichnet.

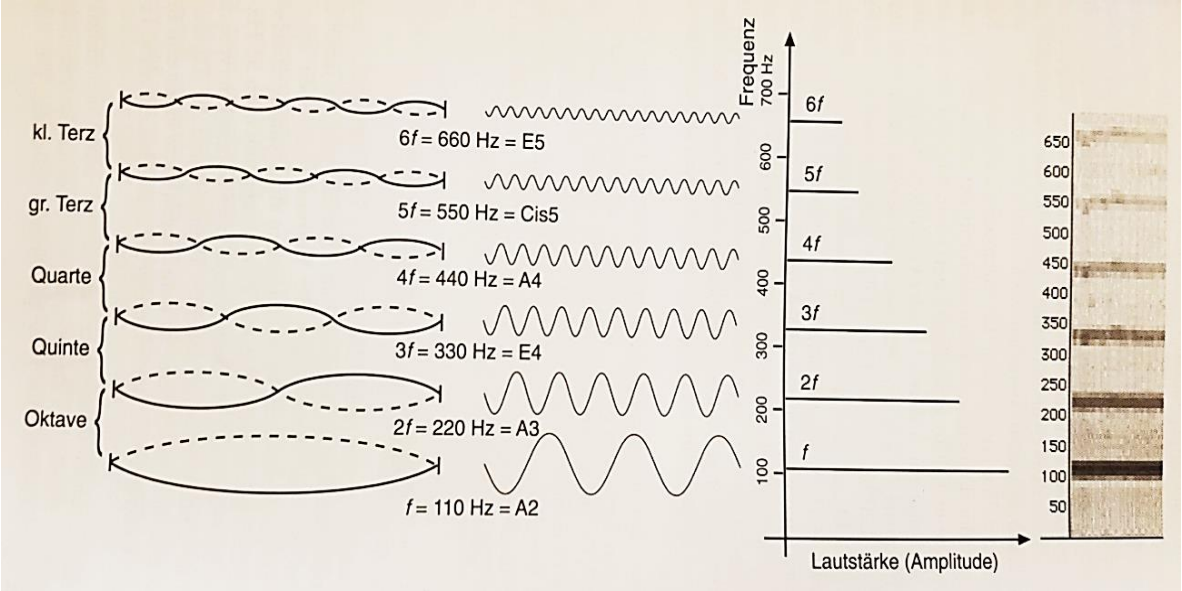

Abbildung 2: Die ersten sechs Obertöne einer schwingenden Saite (Spitzer, 2002, S. 84)

In Abbildung 2 ist links eine idealisierte Saite und ihr Schwingungsverhalten aufgezeigt. Sie schwingt zunächst mit *f* = 110 Hz und wird dann sukzessive aufgeteilt: zunächst halbiert, dann gedrittelt, geviertelt usw., jeweils in gleich große Teile. Die oben aufgezählten Intervalle *Oktave*, *Quinte* und *Quarte* sind entsprechend die ersten drei Schritte von unten beginnend gezählt. Der Abstand des vierten Obertons mit *f* = 550 Hz zum dritten Oberton (440 Hz) wird *große Terz* (Verhältnis 5:4) genannt. Anschließend tritt ein weiteres Intervall auf, die *kleine Terz* (Verhältnis 6:5), die den Abstand zum nächsten Oberton (660 Hz) beschreibt.

Auf der Abbildung rechts neben den Schwingungen sind die Amplituden der jeweiligen Frequenzen abgebildet: zunächst als Diagramm, dass für die exakten Töne die jeweiligen Amplituden anzeigt; dann als so genanntes Spektrogramm, dass die Amplituden der Töne als Grauwert anzeigt – je dunkler die Darstellung, desto lauter ist die

Frequenz (Spitzer, 2002, S. 84). Es ist deutlich zu erkennen, wie die Amplitude mit jedem Oberton absinkt. Die Saite schwingt also mit der Grundschwingung *f* = 110 Hz, gleichzeitig aber auch – wenn auch nicht mehr so stark - mit den jeweiligen Obertönen.

Wie baut man nun eine ganze Tonleiter? Die Grundlage des uns bekannten westlichen Tonsystems geht im Wesentlichen zurück auf Pythagoras, der sich schon ca. 500 v. Chr. mit dem Bauen von Tonleitern beschäftigt haben soll (Görne, 2014, S. 68). Er habe sich dabei nur Quinten (und Oktaven) bedient, um eine Tonleiter mit zwölf Tönen zu konstruieren. Das auf ihn zurückgehende Prinzip war wie folgt:

Man wähle einen beliebigen Ausgangston *f* – beispielsweise 100 Hz - und bilde seine Oktave 2*f* = 200 Hz. Wie schon weiter oben beschrieben, liegt der zweite Oberton (3*f)* eine Quinte über der Oktave. Da die obere Grenze der zu bildenden Tonleiter aber 2*f* ist, wird diesmal beim Grundton angesetzt und von dort aus eine Quinte nach oben gegangen – man erhält somit 3/2*f* = 150 Hz. Um die neuen Töne zu erhalten, wird der letzte also mit 3/2*f* multipliziert. Da dieser neue Ton allerdings mit 225 Hz über 2*f* liegt, wird der Ton halbiert; der neue Ton liegt also bei 112 Hz.

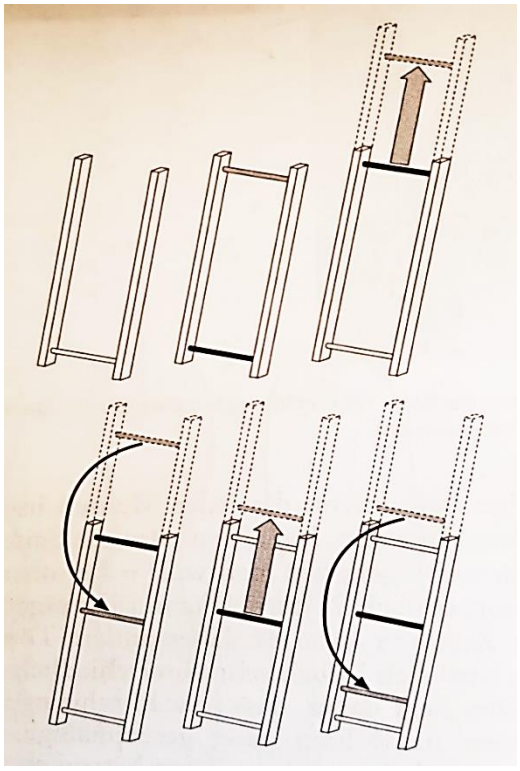

Abbildung 3: Schematische Darstellung des Baus einer Tonleiter nach Pythagoras (Spitzer, 2002, S. 86)

Schematisch lässt sich das gut mit den Sprossen einer Leiter darstellen, wie in Abbildung 3 gezeigt. Die erste Sprosse ist der Grundton, die letzte (oberste) die Oktave. Von dort aus wird eine Quinte gebildet und halbiert; dieser Prozess wird so lange wiederholt, bis die Oktave erreicht wird – so entstehen insgesamt 12 Töne (Spitzer, 2002, S. 87). Wenn man das Ganze mathematisch betrachtet und durchrechnet, wird man feststellen, dass es gar nicht möglich ist, exakt auf die Oktave im Verhältnis 2:1 zu kommen: faktisch geht man insgesamt zwölf Quinten nach „oben“ und sieben Oktaven nach „unten“.

Mathematisch müsste also folgendes gelten:

$$(\frac{3}{2})^{12} = (\frac{1}{2})^{7}$$

Allerdings sind die Zahlen ausgerechnet ungleich:

$$(\frac{3}{2})^{12} = \mathbf{129,7463} \neq \mathbf{128} = \left(\frac{1}{2}\right)^{7}$$

Dieser Fehler ist das so genannte *pythagoreische Komma* und liegt bei deutlich hörbaren 1,36433% (Görne, 2014, S. 70). Bedient man sich also den natürlichen Schwingungsverhältnissen eines Tons, lässt sich eine in sich geschlossene Tonleiter gar nicht bauen (Spitzer, 2002, S. 89).

Im Laufe der Geschichte wurden unterschiedliche Stimmungen angewandt, bei denen manche Intervalle gut miteinander klangen und manche weniger gut. (Görne, 2014, S. 71-72). Doch erst zu Beginn des 20. Jahrhunderts setzte sich die uns bekannte *gleichschwebende Stimmung* durch: hier werden bei der Einteilung der Töne in einem Oktavraum nicht die natürlichen Schwingungsverhältnisse betrachtet; vielmehr werden alle Töne minimal so „verstimmt“, dass sie alle den gleichen Abstand zueinander haben (die gleiche *Schwebung*). Dieser Abstand wird als Halbton bezeichnet und beträgt die zwölfte Wurzel aus 2, also

$$\sqrt[12]{2} = 1{,}05946 \ldots.$$

Um das Rechnen und Stimmen von Instrumenten zu vereinfachen, wird ein Halbton als 100 *cent* angegeben; somit umfasst ein Oktavraum insgesamt 1200 *cent* (Görne, 2014, S. 73). Eine Quinte wird nun aus sieben Halbtönen zusammengesetzt und beschreibt nicht mehr ein Verhältnis von 3:2 = 1,5, sondern folgendes Verhältnis:

$$2^{7/12}{:}\,1 = 1{,}4983$$

Der Fehler der pythagoreischen Stimmung wurde also damit gleichmäßig auf alle Intervalle verteilt - mit dem Vorteil, dass alle Tonarten gleich klingen, da alle Intervalle identisch sind (Görne, 2014, S. 73).

2.1.3 Das heutige Tonsystem in der westlichen Welt

Wie oben bereits genannt, hat sich zu Anfang des 20. Jahrhunderts die gleichschwebende Stimmung durchgesetzt, basierend auf der Überarbeitung des Prinzips nach Pythagoras. Aus den zwölf möglichen Tönen können nun verschiedene Tonleitern gebildet werden, die (meist) sieben Stammtöne enthalten. Die unterschiedlichen Zusammensetzungen der Stammtöne entstanden im Laufe der Zeit aufgrund der Wahrnehmung von *konsonanten* und *dissonanten* Tönen – also, ob zwei Töne zusammen gut oder schlecht klingen. Dabei werden Töne dann als konsonant empfunden, wenn sie viele identische Obertöne gemein haben – als dissonant werden Töne wahrgenommen, wenn die Obertöne unterschiedlich sind und nah beieinander liegen (Spitzer, 2002, S. 98).

Auf Basis dessen entstand mit der Zeit die *Dur-Tonleiter* mit sieben Stammtönen, die mit den Buchstaben *C-D-E-F-G-A-H* bezeichnet werden. Dies sind auch die weißen Tasten des Klaviers (s. Abbildung 4). Zusammen mit den schwarzen Tasten ergeben sie die *chromatische* Tonleiter, die alle zwölf Töne enthält.

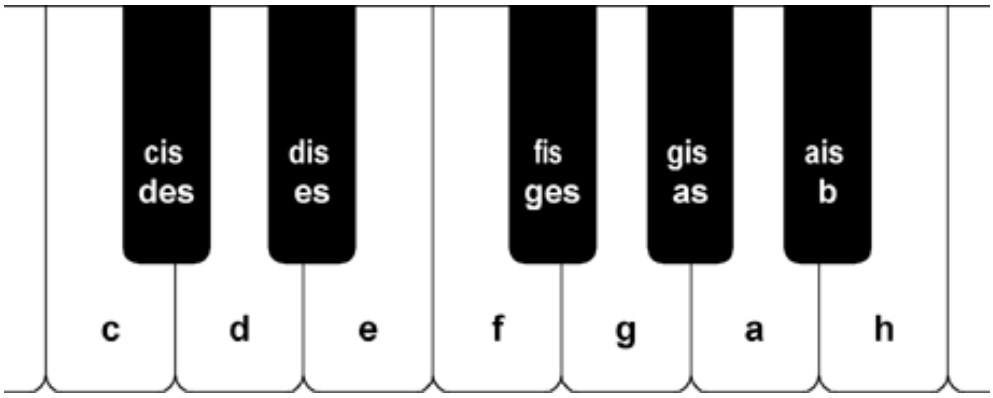

Abbildung 4: Schematische Darstellung einer Klaviatur (Qpaly / Wikimedia Commons, 2014)

Um die übrigen Töne auf den schwarzen Tasten zu erhalten, lassen sich Töne um einen Halbton erhöhen (mit dem Suffix *-is*) bzw. erniedrigen (mit dem Suffix *-es*). Dabei wird jedoch *H* zu ***B*** statt zu *Hes*, und *A* wird zu ***As*** statt zu *Aes*.

Die Töne auf den weißen Tasten des Klaviers bilden die C-Dur-Tonleiter mit den Tönen C-D-E-F-G-A-H-C2 (C2 = Oktave). In Abbildung 4 lässt sich erkennen, dass zwischen den meisten Tönen jeweils zwei Halbtöne (= ein Ganzton) liegen. Zwischen E und F (dritter und vierter Ton) ist ein Halbtonschritt, ebenso zwischen H und C2 (siebter und achter Ton). Mit der Zahlenfolge *3-4-7-8* kann man sich also leicht merken, wie eine Dur-Tonleiter aufgebaut ist, weil genau zwischen diesen Stufen die Halbtonschritte sind. Dies gilt für alle Tonarten, also ebenso für zum Beispiel *E-Dur*: E-Fis-Gis-A-H-Cis-Dis-E. Auch hier ist jeweils ein Halbtonschritt zwischen Gis und A respektive Dis und E.

Wenn als Ausgangston das A gewählt und die weißen Tasten bis zum nächsten A gespielt werden, erhält man die Moll-Tonleiter (auch genannt: Parallele) zur Dur-Tonleiter. In diesem Fall ist *A-Moll* die Parallele zu C-Dur: A-H-C-D-E-F-G-A2. Die Moll-Parallele einer Dur-Tonleiter kommt also durch Verschiebung des Ausgangston (C) um drei Halbtonschritte nach unten (A) zustande und enthält damit die gleichen Töne. Durch die Verschiebung sind die Ganz- und Halbtonschritte ebenfalls verschieden: die Halbtonschritte sind nun zum einen zwischen zweiter (H) und dritter (C), zum anderen zwischen fünfter (E) und sechster (F) Stufe. Wie bereits bei der Dur-Tonleiter kann man hier auch eine Art Telefonnummer – *2-3-5-6* - bilden und sich damit die Position der Halbtonschritte leichter merken.

Mit diesen zwei „Telefonnummern“ als Indikator für die Position der Halbtonschritte innerhalb einer Tonleiter lassen sich also insgesamt jeweils zwölf verschiedene Dur- und Molltonleitern bilden. Zu beachten ist dabei, dass manche Töne entsprechend erhöht oder erniedrigt werden müssen.

Beispiel H-Moll

H – **Cis – D** – E – **Fis – G** – A - H2

Beispiel H-Dur

H – Cis – **Dis – E** – Fis – Gis – **B - H2**

Diese Art der Verschiebung des Ausgangstons lässt sich noch weiterführen, um weitere Tonleitern zu erhalten, die je ihre eigenen Klangcharakter haben und mit unterschiedlichen Emotionen assoziiert werden. Um allerdings den Rahmen dieser Arbeit nicht zu sprengen, wird außer der Dur- und Molltonleiter keine weiteren Tonleitern behandelt.

2.1.4 Intervalle

In den vorangegangenen Kapiteln wurden bereits einige Intervalle genannt. Nachfolgend werden nun alle bisher genannten und fehlenden Intervalle eines Oktavraums in Tabelle 1, Spalte 1, aufgelistet. Die zweite Spalte beinhaltet die Anzahl der Halbtonschritte ausgehend vom Ausgangston. Das Frequenzverhältnis wird in der dritten Spalte beschrieben.

Das Verhältnis zweier Töne ohne Abstand voneinander wird als *Prime* bezeichnet. Beide Töne mit einem Frequenzverhältnis von 1:1 sind also identisch. Geht man zunächst von einzelnen Stufen aus, folgt der Prime die *Sekunde*, dann die *Terz*. Weiter oben bereits genannt wurden die *Quarte* und die *Quinte*. Der Ton zwischen Quarte und Quinte wird Tritonus genannt. Anschließend folgt die *Sexte*, die *Septime* und zum Schluss die *Oktave*.

Tabelle 1: Auflistung aller Intervalle in einem Oktavraum mit Anzahl ihrer Halbtonschritte und Frequenzverhältnis (In Anlehnung an Bernstein, 2019, S.147).

Intervall	**Halbtonschritte**	**Frequenzverhältnis**
Prime	0	1:1
Kleine Sekunde	1	16:15
Große Sekunde	2	9:8
Kleine Terz	3	6:5
Große Terz	4	5:4
Quarte	5	4:3
Tritonus	6	45:32
Quinte	7	3:2
Kleine Sexte	8	8:5
Große Sexte	9	5:3
Kleine Septime	10	16:9
Große Septime	11	15:8
Oktave	12	2:1

Aus der Tabelle lassen sich bei einigen Intervallen Unterscheidungen herauslesen. So wird bei der Sekunde, der Terz, der Sexte und der Septime zwischen *klein* und *groß* unterschieden. Ebenfalls gibt es zur weiteren Beschreibung die Adjektive *vermindert* und *übermäßig*. Der Tritonus kann so beispielsweise auch als verminderte Quinte oder übermäßige Quarte beschrieben werden. Quarte, Quinte und Oktave werden auch als

rein bezeichnet. Es gibt auch Namen für Intervalle über den Oktavraum hinaus (zum Beispiel die *None* als Sekunde über der Oktave), jedoch soll hier nur der Oktavraum behandelt werden.

Manche Intervalle werden konsonanter und manche dissonanter als andere wahrgenommen. Tabelle 2 listet links die Intervalle in abnehmender Konsonanz und rechts in zunehmender Dissonanz auf. Das dissonanteste Intervall ist die kleine Sekunde – hier liegen die Obertöne und sogar der Grundton sehr eng beieinander, weshalb hier sehr viel Schwebung entsteht und unseren Ohren eher nicht gefällt (Spitzer, 2002, S. 98). Die kleine Sexte wird dabei nach Bernstein oft zu den Dissonanzen gezählt (Bernstein, 2019, S. 148).

Tabelle 2: Intervalle abnehmender Konsonanz bzw. zunehmender Dissonanz (in Anlehnung an Bernstein, 2019, S. 148)

Konsonanz	**Frequenzverhältnis**	**Dissonanz**	**Frequenzverhältnis**
Prime	1:1	Kleine Septime	9:5
Oktave	2:1	Große Sekunde	9:8
Quinte	3:2	Tritonus	45:32
Quarte	4:3	Große Septime	15:8
Große Sexte	5:3	Kleine Sekunde	16:15
Große Terz	5:4		
Kleine Terz	6:5		
Kleine Sexte	8:5		

Es wird deutlich, dass hier keine eindeutige Aussage über die Empfindung und den Geschmack eines Menschen auf Basis von rein physikalischen Grundsätzen getroffen werden kann (Spitzer, 2002, S. 103).

Hermann von Helmholtz (1821-1894), ein bedeutender Wissenschaftler des 19. Jahrhunderts, formulierte dazu folgende treffende Aussage:

> *Ob ein Zusammenklang mehr oder weniger rauh ist als ein anderer, hängt nur von der anatomischen Struktur des Ohres, nicht von psychologischen Motiven ab. Wie viel Rauhigkeit aber der Hörer als Mittel musikalischen Ausdrucks zu ertragen geneigt ist, hängt von Geschmack und Gewöhnung ab; daher die Grenze zwischen Konsonanzen und*

> *Dissonanzen sich vielfältig geändert hat. […] Daraus folgt […], daß das System der Tonleitern, der Tonarten und deren Harmoniegewebe nicht bloß auf unveränderlichen Naturgesetzen beruht, sondern daß es zum Teil auch die Konsequenz ästhetischer Prinzipien ist, die mit fortschreitender Entwickelung der Menschheit einem Wechsel unterworfen gewesen sind und ferner noch sein werden." (Helmholtz, 1913, S. 386)*

Bis zu einem gewissen Grad lassen sich also Konsonanzen und Dissonanzen in der Musik physikalisch erklären beziehungsweise ergründen. Was jedoch *gefällt,* liegt schlussendlich im Auge des Betrachters - oder eher im Ohr des Hörers.

2.1.5 Takt, Rhythmus

Bisher wurde vornehmlich die Höhe von Tönen betrachtet und in Bezug gesetzt. Sie unterscheiden sich jedoch auch in ihrer Länge - wie lange sie also erklingen. Wenn Töne nun in einen zeitlichen Bezug gesetzt werden, spricht man von *Rhythmus* (griechisch: *rhythmos* = geregelte Bewegung, Zeitmaß). Er ist untrennbar verwoben mit Melodie und Harmonie eines musikalischen Werkes (Bellosa, 2000, S. 17). Die Basis jedes zeitlichen Verlaufs einer Musik ist das so genannte *Metrum*, ein gleichgewichtiger Puls (Strange-Elbe, 2015, S. 28). Das gliedernde Prinzip, das dem Zeitwert von Tönen obliegt, wird *Takt* genannt. Es beschreibt die relative Länge eines Tons im Kontext eines Werkes und wird als entsprechender Notenwert in der Notation abgebildet. Die absolute Länge eines Tons (in Sekunden) lässt sich ermitteln anhand der Tempoangabe in *beats per minute (bpm)*, also in Schlägen pro Minute (Strange-Elbe, 2015, S. 28). Der bekannteste Takt ist der Viervierteltakt, dargestellt durch den Bruch 4/4; dabei beschreibt der Zähler die Anzahl an Schlägen pro Takt, wohingegen der Nenner die Art dieser Schläge definiert. Dies können beispielsweise Viertelnoten, halbe Noten oder Achtelnoten sein. Im ¾-Takt zum Beispiel finden in einem Takt also drei Viertelnoten Platz, bevor der nächste Takt beginnt. Bei einem Tempo von 60 *bpm* würde dies bedeuten, dass pro Sekunde eine Viertelnote erklingt (Grabner, 2015, S. 35).

2.1.6 Melodie

Am charakteristischsten für die Wahrnehmung von Musik ist die Melodie. Sie macht ein musikalisches Werk wiedererkennbar, da sie ihm einen unverwechselbaren Klang verleihen kann. Melodien, oder auch Musik allgemein, sind „Struktur[en] der Zeit" (Spitzer, 2002, S.115). Eine Melodie ist also eine Kombination von verschiedenen, zeitlich

aufeinander folgenden Tönen. Dies wird als eine fortwährende Bewegung erlebt, da zwischen den einzelnen Tönen eine bestimme Bewegungsenergie herrscht: steigende Tonfolgen werden als spannend, fallende als entspannend empfunden (Grabner, 2015, S. 58). Jeder Ton hat im Kontext der dazugehörigen Tonleiter seine Funktion. In der C-Durtonleiter beispielsweise wird der siebte Ton (H) als *Leitton* bezeichnet, da der Ton eine starke Bewegungstendenz zur Oktave des Grundtons C aufweist. Eine ähnliche Bewegungsenergie weist der dritte Ton zum vierten Ton auf, also von der Terz zur Quarte. Jedem Ton innerhalb einer Tonleiter wohnt eine bestimmte Bewegungstendenz zu anderen Tönen inne, die schlussendlich die „Tonfolge als geschlossenes Ganzes“ (Grabner, 2015, S. 161) definieren. Das erklärt auch, weshalb Menschen sich vordergründig am relativen Verhältnis zwischen Tönen orientieren statt am absoluten. Deshalb werden auch beispielsweise die bekannten Melodien von *Hänschen Klein* oder *Alle meine Entchen* gemeinhin von den meisten Menschen ohne Probleme erkannt, egal in welcher Tonart die Melodie vorgegeben wird (Spitzer, 2002, S. 237).

Das *Motiv* ist die kleinste sinngebende Einheit einer Melodie und stellt damit den kleinsten musikalisch fassbaren Teil eines Werks dar (Spitzer, 2002, S. 130). Es besteht aus einer Gruppe von einer Handvoll Tönen, deren Funktion ähnlich dem eines Wortes in der Sprache entspricht. Gruppiert man mehrere Motive weiter zu einer größeren Gruppe, entstehen so genannte *Phrasen*. Wie in der Sprache können viele Phrasen in einem Satz enthalten sein, zum Beispiel in verschachtelten Sätzen, oder auch nur eine einzige. Aus diesen Bausteinen können Melodien konstruiert werden, in denen Motive mehrmals aufgegriffen und innerhalb eines Werks vielfältig variiert werden. So lassen sich beispielsweise Rhythmus und auch Intervalle einer Melodie so verändern, dass sie trotzdem weiterhin erkennbar bleibt. Eine Phrase kombiniert also mehrere Motive zu einem ausgereifteren musikalischen Ausdruck, der jedoch trotzdem noch im Kurzzeitgedächtnis hängen bleiben kann, da die Länge einer Phrase typischerweise kurz genug dafür ist. Dies entspricht ungefähr der Länge eines Atemzugs (Spitzer, 2002, S. 131).

2.2 Musikinstrumente

Gegenstände, die Schall erzeugen, werden Musikinstrumente genannt. Menschen spielen mit ihnen und drücken sich künstlerisch aus. Der Bau von Instrumenten ist inspiriert von den Eigenschaften der menschlichen Stimme – dem Instrument, das am verbreitetsten ist. Der Körper ist somit immer Teil des Musizierens (Wolf, 2018, S. 338). In den verschiedenen Kulturen der Welt sind unterschiedlichste Instrumente zugegen, denen jeweils eigene Bedeutungen zugeschrieben werden. Dabei entstehen abhängig vom Verständnis von Musik und Klang verschiedenste Musikinstrumente, die auch verschiedenen Zwecken dienen, um Informationen zu übermitteln oder zeitliche Begebenheiten zu strukturieren. Instrumente werden vielfältig eingesetzt: um Träger in Religion und Politik zu repräsentieren; zur Unterhaltung; für die Begleitung tänzerischer Darbietungen und zu therapeutischen Zwecken; als künstlerischer Ausdruck, ob individuell oder im Ensemble. So vielfältig Musik in all ihren Auslegungen ist, so unschätzbar groß sind auch ihre Einsatzmöglichkeiten (Wolf, 2018, S. 338).

Abbildung 5: Eine Auswahl an verschiedenen Musikinstrumenten (Quelle: eigenes Foto)

Abbildung 5 zeigt eine Reihe unterschiedlicher Musikinstrumente verschiedener Musikrichtungen und Länder: ganz links im Bild ist eine *Darbuka*, daneben eine *Daf*. Beide sind Schlaginstrumente, die vor allem in orientalischer Musik zu hören sind. Ebenfalls im Orient heimisch ist die *Oud*, das Saiteninstrument mit dem breiten Korpus ganz rechts im Bild, und die *Ney-Flöte*, im Bild die dünne, lange Flöte links von der Mitte. Sie ist eines der ältesten Instrumente der Menschen, die von vor 4500-5000 Jahren bis heute noch gespielt werden (Taibzadeh & Moezzi, 2014).

Im Laufe der Geschichte haben sich verschiedene Klassifikationen von Musikinstrumenten herausgebildet. Zum einen lassen sich danach kategorisieren, in welchem Kontext sie eingesetzt werden: als Orchesterinstrumente, Instrumente von Rockbands oder Volksmusik, etc. Darüber hinaus wird auch nach der Spielart, wie eine Person das Instrument bedient, unterteilt: als Tasteninstrumente, Zupfinstrumente, Streichinstrumente, Schlaginstrumente, etc. Zuletzt können Musikinstrumente auch nach ihrer physikalischen Tonerzeugung klassifiziert werden (Redaktion Duden Learnattack GmbH, 2019):

- **Chordophone**:
    - Tonerzeugung wird durch das Schwingen von Saiten ermöglicht (zum Beispiel Gitarre, Klavier, …)
- **Membranophone**:
    - Töne werden durch ein schwingendes Fell erzeugt (Pauke, Trommel, …)
- **Aerophone**:
    - Töne werden durch eine schwingende Luftsäule erzeugt (Trompete, Flöte, …)
- **Idiophone**:
    - Töne werden vom Klangkörper selbst erzeugt (Xylophon, Becken, …)
- **Elektrophone:**
    - Tonerzeugung durch elektrische Generatoren, die Schwingungen erzeugen (Synthesizer

So kann ein Cello beispielsweise als Streich-, oder je nach Spielweise als Zupfinstrument betitelt werden. Der Tonerzeugung nach ist es ein Chordophon. Celli können sowohl als Orchesterinstrument als auch als Instrument der Rockmusik verwendet werden (Redaktion Duden Learnattack GmbH, 2019).

Bei den Chordophonen (Saiteninstrumenten) und Membranophonen (Schlaginstrumente) werden Teile des Instruments in Schwingung gebracht und der Schall über

einen Korpus an die Luft übertragen. Der Korpus dient dazu als Resonanzkörper, der den Klang verstärkt und formt (Dickreiter, 1977, S.8). Im Gegensatz dazu wird beim Spielen von Blasinstrumenten eine „schwingende Luftsäule innerhalb des Instruments zum Mitschwingen gebracht“ (Dickreiter, 1977, S.8). Es schwingt also nicht das Instrument selbst, sondern die darin entstehende Luftsäule.

Synthesizer gehören zu den Elektrophonen, sind also elektronische Tasteninstrumente. Sie wurden in den 1960er Jahren zum ersten Mal konstruiert. Durch ihren neuartigen Klang für die damaligen Verhältnisse entstanden im Zuge ihrer Nutzung in der Musik neue Genres, wie zum Beispiel *EDM* (Electronic Dance Music) (Brockhaus, 2017, S. 97). Bekannte Musiker wie Stevie Wonder setzten die neuen Sounds der Synthesizer stilprägend in der Popmusik ein. Die Tonerzeugung geschieht hier durch eine bestimmte Art der Klangsynthese; hiervon haben sich im Laufe der Zeit verschiedene Arten entwickelt.

Abbildung 6: Minimoog Synthesizer der Firma Moog aus den 1970er Jahren (Krash / Wikimedia Commons, 2005)

Die verbreitetste ist die so genannte *Subtraktive Synthese*: hier werden zunächst Schwingungen durch Generatoren erzeugt. Diese sind meist Sinus-, Rechteck-, Dreieck- oder Sägezahnschwingungen, die bis auf die Sinusschwingung entsprechend reich an Obertönen sind. Nun wird das Ausgangssignal mit mehreren Filtern verändert und dabei Frequenzen abgezogen – daher der Name *subtraktive* Synthese (Brockhaus, 2017, S. 98).

Abbildung 6 zeigt einen analogen Synthesizer, der mit subtraktiver Synthese arbeitet. Sofort zu erkennen ist die bekannte Klaviatur – neu im Vergleich zu anderen Instrumenten sind die Knöpfe und Drehregler. Damit lässt sich der Sound sehr vielfältig verändern.

2.2.1 Der Klang von Musikinstrumenten

Nicht nur feine Körper wie Saiten können schwingen - auch andere Körper, wie zum Beispiel Holzbretter, Rohre oder Glas sind in der Lage zu schwingen. Dabei klingen alle unterschiedlich, da sie jeweils ein ganz eigenes Obertonspektrum haben, das mitschwingt (Spitzer, 2002, S. 35), wenn ein Ton erzeugt wird. Daher klingt ein Klavier nicht wie ein Kontrabass oder eine Violine wie ein Saxofon, obwohl sie durchaus „gleiche“ Töne spielen können und bei ihnen dieselben in Kapitel 2.1 erläuterten Regeln Anwendung finden. Durch die Addierung von Grundschwingung und Obertonschwingungen entstehen teils sehr komplexe Schwingungen. Jedoch lässt sich bei diesen Schwingungen trotzdem eine Periodizität ausmachen. Wie eingangs in Kapitel 2.1.1 erläutert, nehmen Menschen genau dann einen *Ton* wahr, wenn sich eine regelmäßige Wiederkehr im Schwingungsverlauf ergibt.

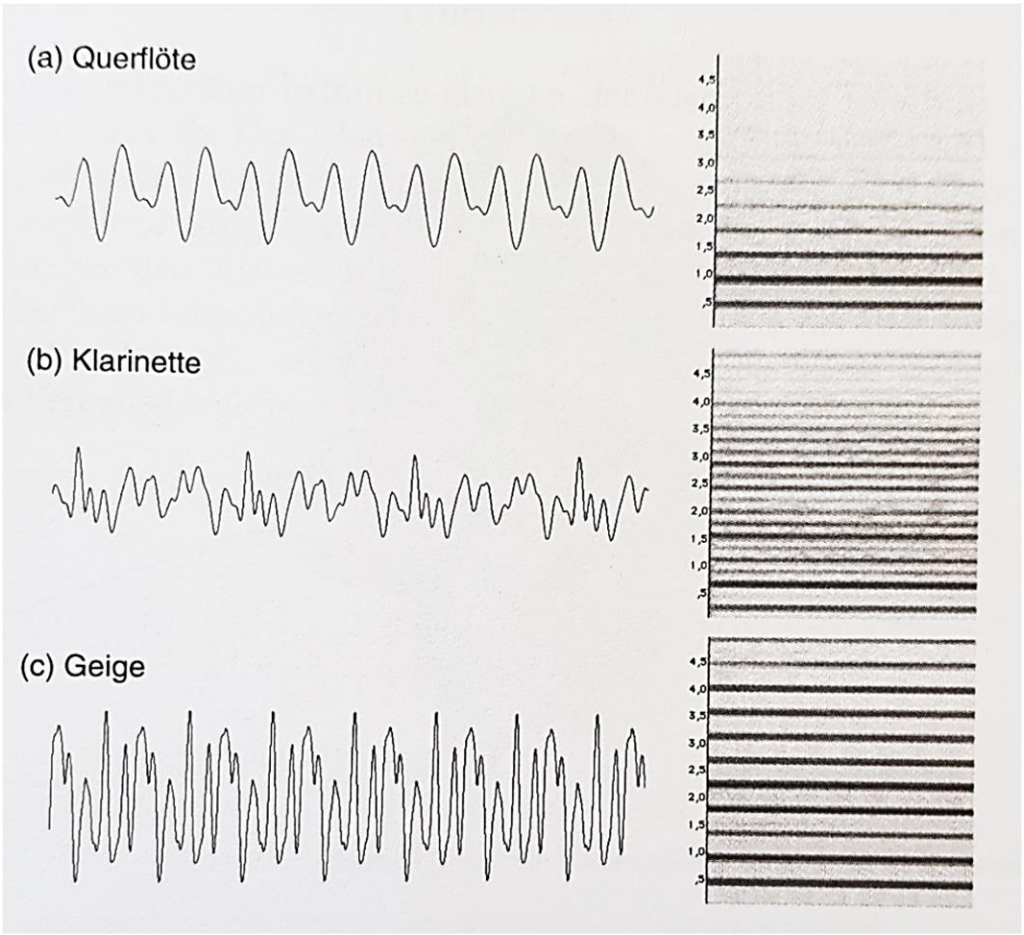

Abbildung 7: Wellenform und Spektrogramm einiger Instrumente (Spitzer, 2002, S.37)

In Abbildung 5 sind Schwingungsverläufe einiger Instrumente dargestellt. Obwohl die Kurven teils komplexe Verläufe haben, ist dennoch eine Periode zu erkennen. Auf der rechten Seite sind zusätzlich die Intensitäten der jeweiligen Obertöne als Spektrogramm dargestellt. Auffällig ist dabei, dass bei der Querflöte relativ wenig Obertöne vorhanden sind, weshalb sich die Wellenform auch simpler gestaltet im Vergleich zu denen der anderen Instrumente.

In Kapitel 2.1.1 wird auch erläutert, was ein Geräusch in Abgrenzung zu einem Ton ist. Dementsprechend gibt es nicht nur Musikinstrumente, die Töne erzeugen, sondern auch jene, die Geräusche (*Rauschen*) erzeugen. Dabei werden nicht wahllos irgendwelche Geräusche erzeugt, sondern oft welche, die in der Natur des Menschen vorkommen – beispielsweise sind die Becken eines Schlagzeugs den Zischlauten unserer Sprache nachempfunden (Spitzer, 2002, S. 37).

2.2.2 Tonerzeugung bei Saiteninstrumenten

Bei Saiteninstrumenten können verschiedene Techniken zur Tonerzeugung angewandt werden. Die Saiten lassen sich mit Fingern, Fingernägeln oder Plektren zupfen, mit Hämmerchen anschlagen oder mit einem Bogen streichen (Spitzer, 2002, S. 39). Mitunter haben sich kreative Alternativen zu diesen Techniken entwickelt, wie zum Beispiel das *Slapping* auf Bass-Gitarren, bei dem mit dem Daumen auf die dicken Saiten eines elektronischen Basses geschlagen wird, um einen schnalzenden Ton zu erzeugen. Abbildung 6 zeigt drei verschiedene Techniken mit Wellenform (oben), Amplitudendarstellung (Mitte) und Spektrogramm (unten).

Je nach Anschlagtechnik klingen die in der Abbildung gezeigten Töne, die jeweils auf der gleichen Saite erzeugt wurden, anders. Schon an der Wellenform lassen sich Unterschiede erkennen. Deutlicher wird es bei der Darstellung der Amplitude: wenn die Saite mit einem Hämmerchen angeschlagen wird, fängt der Ton direkt am lautesten an (Mitte). Beim Zupfen (mit Fingerkuppe) ist es etwas schwächer, aber dennoch am Anfang am lautesten mit abfallender Lautstärke.

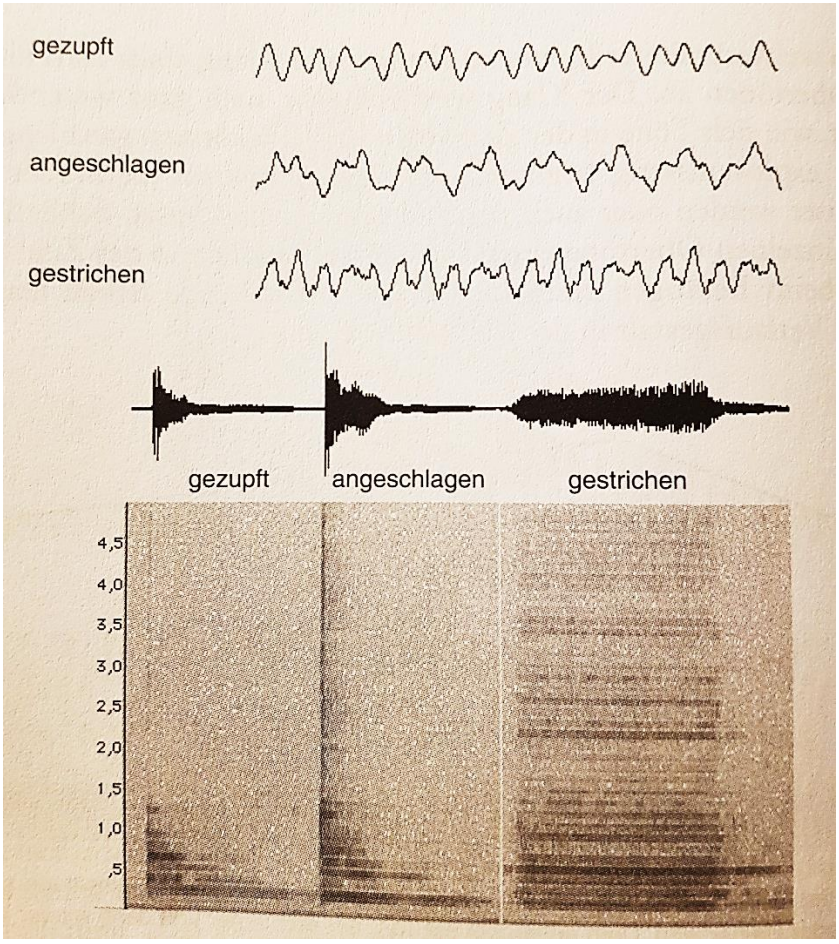

Abbildung 8: Verschiedene Anschlagtechniken auf der gleichen Saite, mit Wellenform, Amplitudendarstellung und Spektrogramm (Spitzer, 2002, S. 40)

Wenn die Saite mit einem Bogen gestrichen wird, erklingen die Saiten sanfter und steigen allmählich in der Lautstärke an. Wie in Kapitel 2.1.1 erwähnt, äußert sich der Klangcharakter eines Tons auch im zeitlichen Verlauf der Obertöne. Das Spektrogramm der jeweiligen Techniken (unten in der Abbildung) zeigt dies deutlich. Während beim Zupfen und Schlagen die Obertöne mit der Zeit abschwellen, bleiben sie beim Streichen relativ konstant in ihrer Intensität.

Die Art, wie ein Ton beim Streichen mit Bogen erklingt, wird auch als *Einschwingen* bezeichnet (Spitzer, 2002, S. 40). Auch die verschiedenen Obertöne erklingen beim Einschwingen unterschiedlich lang. Dies erklärt, warum natürlich erzeugte Töne im Vergleich zu elektrisch erzeugten Tönen oft lebendiger klingen.

2.2.3 MIDI

MIDI bedeutet *Musical Instrument Digital Interface* und ist ein Protokoll zur Echtzeit-Übertragung von musikalischen Informationen zwischen verschiedenen Hardware-Geräten, vornehmlich elektronische Klangerzeuger (*Synthesizer*) und Computer. Die 1982 standardisierte Spezifikation ermöglicht es Musikern, ohne größeren Aufwand auch zuhause Musik zu produzieren und dabei auf eine große Vielfalt an (virtuellen) Musikinstrumenten zurückzugreifen, ohne ein teures Tonstudio mit entsprechenden Musikern nutzen zu müssen (Ackermann, 1991, S. 186). Dabei enthält ein MIDI-Befehl kein echtes Audiomaterial, sondern reine Steuerdaten, die unter anderem verschiedene Informationen zur Ausführung und Gestaltung eines Tons enthalten können: Notenwert, Anschlagstärke, Veränderung der Tonhöhe und so weiter (Ackermann, 1991, S. 193). MIDI entstand ursprünglich dadurch, mehrere Synthesizer zu verbinden, um sie von einer einzigen Tastatur aus zu bedienen (Stange-Elbe, 2015, S. 243).

2.3 Gamepads

Das *Gamepad* (auch: Controller, Joypad) ist das primäre Eingabegerät für Videospiele und hat seit den 1970er Jahren eine Reihe an Entwicklungen hinter sich. Mit dem Gamepad werden Objekte oder Charaktere eines Videospiels gesteuert, meist mittels Eingabemöglichkeiten, wie zum Beispiel (Dreh-) Knöpfe in unterschiedlichster Anordnung. Die Entwicklung des Gamepads hängt eng mit dem technologischen Fortschritt, der Forschung und der Ausgestaltung von Videospielen zusammen.

In den 1970er Jahren, als Videospiele erstmals auf so genannten Arcade-Automaten erschienen sind, hatte jeder Automat eine eigene Bedienung, passgenau für das jeweilige Spiel. Das sehr bekannte Spiel *Pong*, das dem Spielprinzip von Tischtennis ähnelt, benutzte so genannte *Paddles*; das sind Drehregler, mit denen man die Schläger nach oben und unten bewegen kann. Ein anderes Spiel namens *Marble Madness* benutzte einen so genannten *Trackball* – eine Kugel, die sich in alle Richtungen drehen lässt, um eine virtuelle Kugel im Spiel durch einen Hindernislauf zu manövrieren (Brown, 2016, 00:25-00:53).

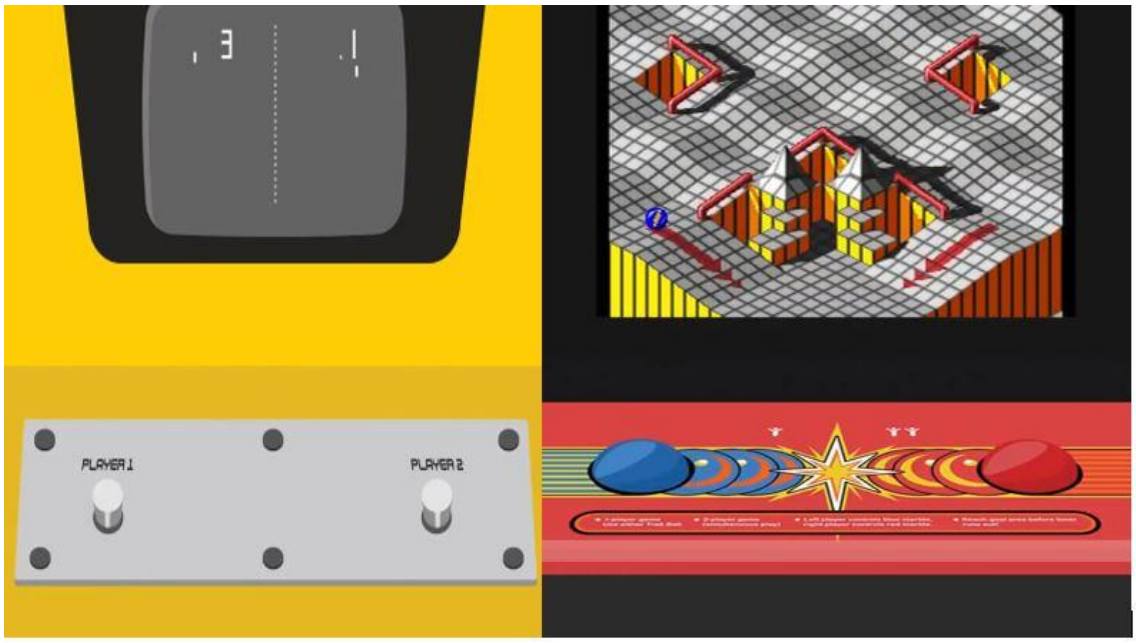

Abbildung 9: Arcade-Automaten "Pong" und "Marble Madness" (eigene Darstellung, in Anlehnung an: Brown, 2016, 00:28-00:39)

In Abbildung 9 sind für beide Spiele jeweils stilisierte Grafiken der Arcade-Automaten zu sehen. Es ist gut erkennbar, dass beispielsweise mit der Steuerung von *Pong* kaum möglich wäre, *Marble Madness* zu spielen. Universelle Controller, die für die meisten Spiele passen, kamen erst dann auf, als die Heimkonsole auf den Markt kam. Nennenswert war der Joystick der erfolgreichen Konsole *Atari 2600* (Brown, 2016, 00:42-00:50). Die japanische Firma *Nintendo* setzte dann 1983 für die damalige Zeit den Standard mit dem Controller des *Nintendo Entertainment System (NES)*.

Abbildung 10: Stilisierte Grafik eines NES-Controllers (Screenshot aus: Brown, 2016, 00:59)

In Abbildung 10 links ist das so genannte *D-Pad* (Directional Pad) zu sehen; ein Steuerkreuz, mit dem man einen Charakter sehr gut in einer 2D-Umgebung bewegen kann. Weltweite bekannte Spiele sind zum Beispiel *Super Mario* und *The Legend of Zelda* (Brown, 2016, 00:58-01:02). Dieser Controller war universeller und konnte somit für viele Spiele eingesetzt werden.

Als Videospiele zunehmend in 3D-Welten spielten – einhergehend mit dem technischen Fortschritt -, etablierte Nintendo erneut einen Standard mit dem 1996 erschienenen *Nintendo 64* Controller.

Abbildung 11: Stilisierte Grafik eines Nintendo 64 Controllers (Screenshot aus: Brown, 2016, 01:29)

Wie in Abbildung 11 zu sehen ist, unterscheidet sich dieser sehr stark von den übrigen Controllern. Auffällig ist die dreizackige Ausführung mit dem Drehknüppel in der Mitte. Dieser ermöglicht es, einen Character wie Mario in einer 3D-Umgebung in alle Richtungen zu bewegen (Heatherly & Howard, 2014, S. 2). Das bereits bekannte D-Pad ist ebenso vorhanden sowie weitere Tasten, die immer mehr Spielinteraktionen ermöglichen.

Diese Entwicklung wurde durch den 1997 eingeführten *DUALSHOCK* Controller der Sony PlayStation manifestiert. Sony kombinierte diesmal zwei analoge Joysticks mit dem D-Pad und führte insgesamt vier Schultertasten ein, ebenso viele weitere Tasten (Lu, 2003, S. 17).

Abbildung 12: Stilisierte Grafik eines DUALSHOCK Controllers der Sony PlayStation (Screenshot aus: Brown, 2016, 01:39)

Eine Besonderheit des in Abbildung 12 gezeigten Gamepads ist die Vibrationsfunktion: bei bestimmten Ereignissen im Spiel vibriert der gesamte Controller für eine kurze Zeit (Brown, 2016, 01:38). Die meisten der zeitlich darauffolgenden Controller basieren auf dem Design dieses Controllers. Sie haben zwar alle ihre individuellen Platzierungen für D-Pad, Analogsticks und Buttons, doch im Kern sind sie sehr ähnlich (Lu, 2003, S. 18).

Eine innovative Neuheit wurde – erneut – von Nintendo geboten, als 2006 die *Wii Remote* veröffentlicht wurde, der Controller für die Konsole *Nintendo Wii* (Margel, o. D., S. 3). Der Controller enthält wie die anderen ein D-Pad und Buttons, allerdings völlig anders angeordnet, wie in Abbildung 13 zu sehen ist. Neu war jedoch ein Beschleunigungssensor, der die Bewegungen an drei Achsen erkennen konnte; hinzu kam eine eingebaute Infrarot-Kamera, mit der man die Richtung, in die der Controller gehalten wird, anzeigen konnte.

Abbildung 13: Wii Remote von Nintendo (Screenshot aus: Brown, 2016, 03:09)

Die Wii Remote war insofern auch sehr eigen, da sie im Vergleich zu den anderen Controllern teilweise ganzen Körpereinsatz beim Spielen einforderte. Einfache und intuitive Sportspiele wie Bowlen oder Tennis, die auf der Wii gespielt werden können, setzten sich sogar bei älteren Menschen durch. Dies führte zum Beispiel bei Menschen eines Pflegeheims teilweise zu mehr Bewegung, höherem Selbstbewusstsein und weniger Einsamkeit also bei jenen, die nur Brettspiele spielten (Margel, o. D., S. 3).

Mit dem Aufkommen von immer leistungsfähigeren mobilen Endgeräten wuchs auch der Markt für mobile Videospiele. Smartphones mit Touchscreens ermöglichen eine weitere Bedienung von Spielen, die mitunter noch intuitiver ist, da man direkt mit den Fingern interagiert.

Abbildung 14: Stilisierte Grafik eines iPads der Firma Apple (Screenshot aus: Brown, 2016, 03:28)

Mobile Endgeräte wie das in Abbildung 14 gezeigte *iPad* der Firma Apple sind im Prinzip leistungsstarke Computer, die man überall hin mitnehmen kann. Durch ihre berührungssensitive Bedienung ist es sogar möglich, virtuelle Musikinstrumente wie Piano oder Schlagzeug zu spielen.

Abbildung 15: Eine Schlagzeug-App auf dem iPad (Pierini / Appleman, 2016)

Abbildung 15 zeigt eine App, die ein spielbares Schlagzeugset anbietet. Musiker auf YouTube haben diese Möglichkeit für sich entdeckt und produzieren damit teils sehr erfolgreiche Videos (Pierini, 2016). Diese Art des Musizierens ist relativ natürlich im Vergleich zu einem bloßen Tastendruck auf Buttons eines Gamepads, da die Finger direkt auf einem Touchscreen sind und somit direkt auf dem virtuellen Instrument – dies ist insgesamt eine intuitive Umsetzung und relativ nah am *richtigen* Musizieren.

2.4 Das MDA-Framework

Das MDA-Framework wurde in den Jahren 2001-2004 von der Game-Designerin Robin Hunicke, dem Pädagogen und Game-Designer Marc LeBlanc und Games-Entwickler Robert Zubek im Rahmen der *Game Developers Conference* in San José entwickelt. MDA steht dabei für *Mechanics, Dynamics* und *Aesthetics* und beschreibt eine formale Herangehensweise, Games besser zu verstehen. Das Ziel ist, eine Brücke zwischen Game-Design, Game-Entwicklung und der Rezeption und Kritik an Games zu schlagen. Game-Design bedeutet dabei Konzeption von Games, Game-Entwicklung die tatsächliche Ausprogrammierung des Konzepts. Die Hoffnung der Entwickler ist, dass diese Methodik den iterativen Prozess der Designer und Entwickler stärkt und gleichermaßen verständlicher macht, und es somit allen Parteien einfacher fällt, Games als Ganzes zu studieren und zu verstehen (Hunicke et al., 2004, S. 1). Die drei Schlagwörter bedeuten im Einzelnen:

- **Mechanics**
  Die Mechanics beschreiben die grundlegenden Regeln und die speziellen Komponenten eines Spiels wie Algorithmen und Datenstrukturen. Sie beinhalten alle Funktionalitäten und Interaktionsmöglichkeiten für Spieler*innen, die selten direkt sichtbar für sie sind.
- **Dynamics**
  Dynamics definieren das Verhalten zur Laufzeit des Spiels, das sich durch die Mechanics ergibt. Es ist das, was Spieler*innen sehen, womit sie interagieren können, ob sie miteinander oder gegeneinander agieren, welche Strategien sich als beste herausstellen und so weiter…
- **Aesthetics**
  Aesthetics beschreiben nicht etwa die visuellen Merkmale des Spiels, sondern welche Emotionen bei Spieler*innen ausgelöst werden, wenn sie mit dem Spiel

interagieren. Es wird also danach gefragt, ob das Spiel Spaß macht, frustrierend ist, traurig stimmt, etc.

Dadurch sollen klarere Design Entscheidungen getroffen werden können und eine Analyse auf allen Ebenen der Games-Entwicklung ermögliche (Hunicke et al., 2004, S. 2).

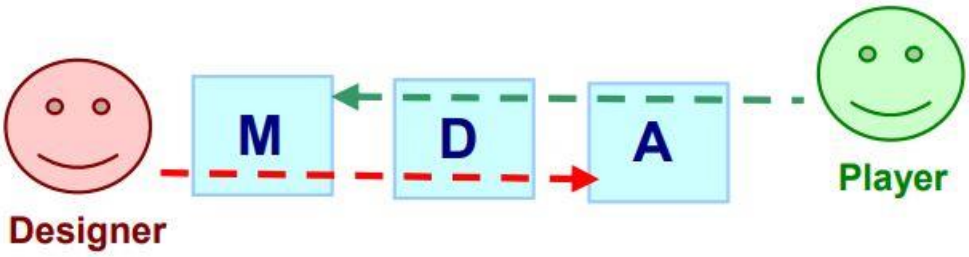

Abbildung 16: Verschiedene Perspektiven im MDA-Framework (Hunicke et al., 2004, S. 2)

Game-Designer*innen und Spieler*innen haben jeweils eine unterschiedliche Perspektive auf Spiele. Spieler*innen starten ihre Spielerfahrung mit dem Aesthetics-Aspekt des Spiels und können mit den Dynamics interagieren, haben aber keinen Einfluss auf die Mechanics. Game-Designer*innen und Entwickler*innen, da sie das Spiel produzieren, fangen natürlicherweise zunächst mit den Mechanics an und arbeiten sich zu den Dynamics vor. Es ist also sehr hilfreich, sich mit beiden Perspektiven auseinanderzusetzen, wenn Spiele designt entwickelt werden, um so die bestmögliche Erfahrung für Spieler*innen zu schaffen (Hunicke et al., 2004, S. 2).

Formulierungen wie „es macht Spaß“ sind zu ungenau, um daraus konkrete Aussagen zu treffen. Das MDA-Framework definiert deshalb eine nicht-finale Liste von Kategorien, die dazu helfen sollen, die Aesthetics eines Spiels besser zu beschreiben. Tabelle 3 zeigt eine Auflistung dieser Begriffe und ihre jeweiligen Bedeutungen mit ergänzender Erklärung.

Sensation	Spiel zum Verwöhnen der Sinne
	Spiele, die auf eine besondere Art und Weise die fünf Sinne des Menschen treffen; dies können solche sein, die einen herausstechenden Soundtrack haben oder eine einzigartige grafische Ausgestaltung aufweisen.

Fantasy	Spiel mit surrealen Elementen
	Diese Aesthetic ermöglicht es Spieler*innen, in Rollen zu schlüpfen, die im „echten“ Leben nicht möglich sind oder unter realistischen Umständen nicht oder kaum möglich sind.
Narrative	Spiel mit Story, Geschichten
	Spiele, in denen es hauptsächlich um die Geschichte der Charaktere und deren Beziehungen geht.
Challenge	Spiel, bei dem Hindernisse überwunden werden müssen
	Hier geht um das Engagement und den Spaß der Spieler*innen, Hindernisse in Spielen zu überwinden.
Fellowship	Spiel im sozialen Rahmen
	Spiele, die die Kooperation zwischen Spieler*innen fordern und fördern, um gemeinsam Ziele zu erreichen.
Discovery	Spiel zum Entdecken von Unerforschtem
	Spiele, die dazu anregen, unbekanntes Terrain zu entdecken. Dies können große, landläufige Gebiete sein, aber auch geheime Tricks und Kniffe in einem Spiel.
Expression	Spiel zur Selbstfindung und Ausdruck unseres selbst
	Spiele, die den dem Menschen innewohnenden Antrieb zur Selbstentfaltung ermöglichen, also jene mit kreativen Gestaltungsmöglichkeiten statt nur vorgegebenen Funktionen.
Submission	Spiel als Zeitvertreib
	Spiele, die es Menschen ermöglichen, abzuschalten, zu entspannen, sich berieseln zu lassen.

Tabelle 3: Liste der Begriffe zur Beschreibung der Aesthetics eines Spiels (eigene Tabelle; in Anlehnung an: Hunicke et al., 2004, S. 2)

Diese Formulierungen lassen sich nun auf jegliche Spiele – nicht nur Videospiele – anwenden. Das allseits bekannte *Pantomime-Spiel* (auch bekannt als *Scharade*) macht den meisten Menschen *Spaß*, allerdings ist diese Information zu ungenau. Es gibt verschiedene Ausführungen des Spiels, aber grob geht es darum, Begriffe

pantomimisch darzustellen. Meist spielen zwei Team gegeneinander. Betrachtet man das Ganze nun aus der Aesthetics-Perspektive, lässt sich feststellen, dass die *Fellowship* Aesthetic aufgrund der Teams hervorsticht; ebenso *Expression*, weil die Spieler*innen eine ganz eigene spontane pantomimische Interpretation des Begriffs haben. Auch die *Challenge* Aesthetic lässt sich heranführen, da die Teams ja gegeneinander spielen und auch gegen die Zeit spielen (Hunicke et al., 2004, S. 2).

Die Autor*innen führen an, dass es keine Superformel für die perfekte Zusammensetzung gebe, um ein Spiel „Spaß“ machen zu lassen, jedoch helfe diese Klassifizierung zu beleuchten, wie und wieso verschiedene Spiele verschiedenen Spieler*innen gefallen (Hunicke et al., 2004, S. 3).

3 Untersuchung existierender Musikspiele

In diesem Kapitel sollen bereits existierende Musikspiele analysiert werden, bei denen Musik eine zentrale Rolle spielt und Klänge durch eine Eingabemöglichkeit getriggert werden können. Pro Spiel soll ein kurzer Überblick gegeben werden; anschließend wird der Fokus auf der Steuerung liegen, wie genau die Töne eingegeben werden.

Das Genre der Musikspiele ist eher klein, geht aber bis in die 1970er Jahre zurück (Kayali & Pichlmair, 2007, S. 1). In jüngster Zeit gab es allerdings viele erfolgreiche Musikspiele, von denen welche auch Gegenstand dieser Untersuchung sein werden. Allgemein gibt es nach den Autoren Kayali und Pichlmair drei Varianten an Musikspielen:

- **Rhythmusspiele** erfordern vor allem das Drücken von Buttons zu einem vorgegebenen Rhythmus. Geschwindigkeit und Komplexität steigen mit dem Fortschritt im Spiel. Meist werden Punkte vergeben und ein großer Faktor ist das Kompetitive (Kayali & Pichlmair, 2007, S. 1).
- Andere Spiele fungieren prinzipiell wie ein **elektronisches Musikinstrument**, werden jedoch mit einem Gamepad gesteuert. Somit gewähren sie viel höhere Freiheitsgrade, da Töne beziehungsweise Rhythmen von Spieler*innen selbst kreiert werden können, ähnlich wie bei einem richtigen Instrument (Kayali & Pichlmair, 2007, S. 2).
- Eine weitere Gruppe an Spielen sind jene, die **musikalische Rätsel** oder Herausforderungen enthalten. Sehr bekannt ist dabei das Spiel *The Legend of Zelda: Ocarina of Time*, bei dem im Spiel bekannte Melodien an verschiedenen Stellen nachgespielt werden müssen, um bestimmte Ereignisse auszulösen.

Es wird ersichtlich, dass Rhythmusspiele nur eine geringe Freiheit bieten, sich musikalisch auszudrücken, da vom Spiel alles vorgegeben wird (Kayali & Pichlmair, 2007, S. 3). Dies zwingt den Spieler dazu, die Darbietung auf ganz bestimmte Art und Weise auszuführen. Es ist also nicht möglich, eigene Klangwelten zu erschaffen. Manche Spiele bieten jedoch eine Art Freiform-Spiel (*free-form* play). Dies ist oft ein spezieller Modus des Spiels, in dem gewisse Mechanics des Spiels zurückgestellt werden zu Gunsten des freien Spiels mit Tönen oder Rhythmen ähnlich eines Instruments. Die Autoren merken dabei an, dass ein Spiel, bei dem in solchen Fällen das zuvor definierte Ziel des Spiels außer Kraft gesetzt wird, eher als *Spielzeug* oder als *non-game* betrachtet wird (Kayali & Pichlmair, 2007, S. 5). Allerdings ergänzen sie auch, dass damit andere Ziele geschaffen werden, und zwar die eigenen Ziele der Spieler*innen.

3.1 Rhythmusspiel: Guitar Hero

Guitar Hero ist der Name einer ganzen Videospielserie, deren erstes Spiel im Jahre 2005 vom Entwicklerstudio *Harmonix* erschien. Es war ein überraschender Verkaufsschlager mit über 1,5 Millionen verkauften Einheiten. Auch die darauffolgenden Spiele verkauften sich sehr gut (Petkovic, 2008). Ziel des Spiels ist es, zu einem vorgegebenen Song – meist weltweit bekannte Stücke – die im Spiel angezeigten Farben zur richtigen Zeit nachzuspielen. Es gibt eine *Karrieremodus*, in dem man anfangs etwas leichtere Songs spielt und nach einigen Songs zu höheren Stufen aufsteigt – dadurch werden neue Songs freigeschaltet. Insgesamt gibt es sechs Stufen mit je fünf Songs. Der Karrieremodus ist durchgespielt, wenn man alle Songs jeder Stufe erfolgreich beendet hat. Nun lassen sich die Songs erneut mit anderen Schwierigkeitsgraden spielen. Weiterhin gibt es einen Multiplayermodus, der es erlaubt, lokal gegen andere Spieler*innen anzutreten – dazu bedarf es einen zweiten Gitarrencontrollers.

Zum Spielen wird ein Controller in Form einer Gitarre (später auch andere Instrumente) benutzt, der statt richtigen Saiten nur eine Handvoll Buttons, einen breiten Kippschalter zum Anschlagen und ein Tremolo enthält. Außerdem sind unten am Korpus noch *Start* und *Select* Buttons angebracht, um das Navigieren durch die Spielmenüs ohne extra Controller zu ermöglichen und damit zu vereinfachen.

Abbildung 17: Y2kcrazyjoker4 / Wikimedia Commons (o. D.)

Wie in Abbildung 17 zu sehen ist, sind die fünf Buttons oben auf dem Griffbrett farbig entsprechend gestaltet. In der Mitte unten ist der Kippschalter, der mit dem Daumen bedient wird. Gelegentlich können längere Töne im Spiel mit dem Tremolo-Hebel (rechts vom Kippschalter) gehalten werden; dazu bewegt man den Hebel hin und her und kann so minimal die Tonhöhe nach oben und nach unten verändern. Extra Punkte gibt es allerdings nicht dafür. Abbildung 18 zeigt das Griffbrett, auf dem sich die einzelnen farblichen Buttons zum Spielenden hinbewegen. Die Buttons mit einem schimmernden Licht können gedrückt werden, ohne den Kippschalter zu bewegen, vorausgesetzt, man hat den vorherigen regulären Ton getroffen.

Abbildung 18: Die farblichen codierten Buttons von Guitar Hero (HowStuffWorks, o. D.)

Hin und wieder haben Buttons auf dem Griffbrett eine sternförmige Umrandung. Wenn diese Töne richtig gegriffen werden, steigt die so genannte *Star Power*. Wenn die Star Power Leiste eine bestimmte Grenze überschritten hat, lässt sich die Star Power aktivieren und die Punkte, die nachfolgend erzielt werden, werden vervielfacht. Die Star Power lässt sich entweder durch schnelles Aufrichten des Gitarrenhalses oder durch Drücken der Select-Taste auslösen. Außerdem werden diverse Animationen und visuelle Effekte ausgelöst, die zur jeweiligen Spielfigur passen; so wird beispielsweise das Publikum lauter und feuert die Spielfigur an. In Abbildung 19 ist rechts die Spielfigur zu sehen, darunter die Star Power Leiste und die *Rock* Anzeige. Grün bedeutet hier, dass

man die meisten Noten trifft und insgesamt eine gute Leistung abgibt. Rot wird angezeigt, wenn man die viele Noten falschspielt und somit eine schlechte Performance darbietet – dies wird oft begleitet von Buh-Rufen des Publikums. Mit dem Star Power Feature jedoch lässt sich diese Anzeige schnell wieder ins Grüne bewegen, wenn man eine Reihe von Sternnoten richtig spielt.

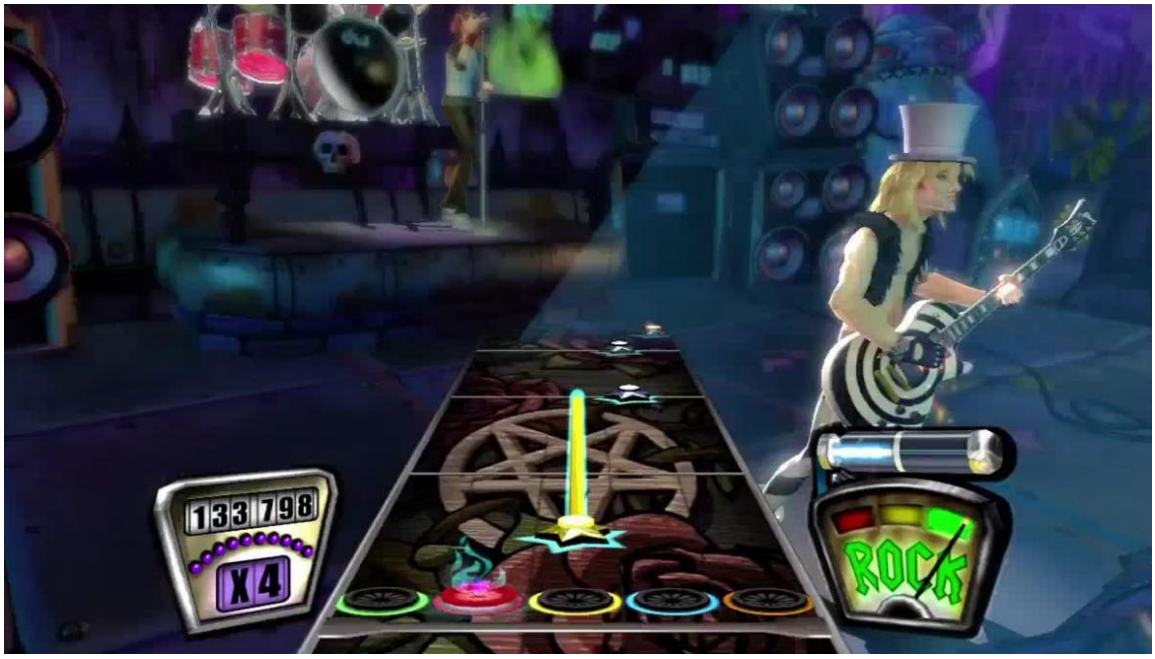

Abbildung 19: Die einzelnen Anzeigeelemente während dem Spiel (WikiHero / Gtamysteries, o. D.)

In Guitar Hero gibt es verschiedene Schwierigkeitsgrade, die sich darin unterscheiden, wie schnell die Noten zu spielen sind und welche der farbigen Buttons zum Einsatz kommen. Bei der niedrigsten Stufe kommen zunächst nur Noten für die oberen drei Tasten; bei höheren Schwierigkeitsgraden kommen dann mit jeder Stufe eine neue Farbe hinzu. Zudem steigen die Geschwindigkeit und Anzahl der aufkommenden Noten. Wenn zu viele Noten verfehlt werden – die *Rock* Anzeige also zu lange im roten Bereich ist -, kann der Song sogar abgebrochen werden.

Es wird deutlich, dass Rhythmus der Kern von Guitar Hero ist (Kayali & Pichlmair, 2014, S. 5). Das Spiel erlaubt es nicht, Musik in reiner Form, wie in Kapitel 2.1 zu praktizieren. Eher lässt es die Musik weiterspielen, wenn man zum richtigen Zeitpunkt die richtigen Knöpfe drückt (Kayali & Pichlmair, 2014, S. 6). Aus MDA-Perspektive betrachtet wird die Aesthetic *Fantasy* erfüllt: beim Spielen von Guitar Hero bekommen Spieler*innen das Gefühl, ein Rockstar zu sein. Dies wird durch bestimmte Mechanics und Dynamics ermöglicht, wie beispielsweise durch die Star Power Mechanik – durch

das Hochreißen des Gitarrencontrollers und das Aktivieren der Star Power feuert das Publikum die Spielfigur intensiver an; man spielt sich in eine Art Rausch. In Guitar Hero zu protzen bedeutet, perfekt zu spielen und eine beeindruckende Performance abzuliefern – jedoch nicht musikalisch, sondern speziell auf das Spiel abgestimmt (Kayali & Pichlmair, 2007, S. 5).

3.2 Puzzle-Plattformer: Wandersong

Wandersong ist ein 2018 veröffentlichtes Indie-Spiel, das für PC, PlayStation 4 und Nintendo Switch erschienen ist. Es ist ein Puzzle-Plattformer, in dem man einen bescheidenen Barden spielt, der hauptsächlich mithilfe seines Gesangs mit der Umwelt interagiert. Sein Ziel ist es, die Melodie des so genannten *Earthsongs* zu finden, um damit die Welt zu retten. Dazu wird die Spielfigur in einer 2D Umgebung aus seitlicher Perspektive durch die bunte Spielwelt bewegt und erlebt zahlreiche Abenteuer, löst Rätsel und lernt viele Charaktere kennen (Lobanov, 2018).

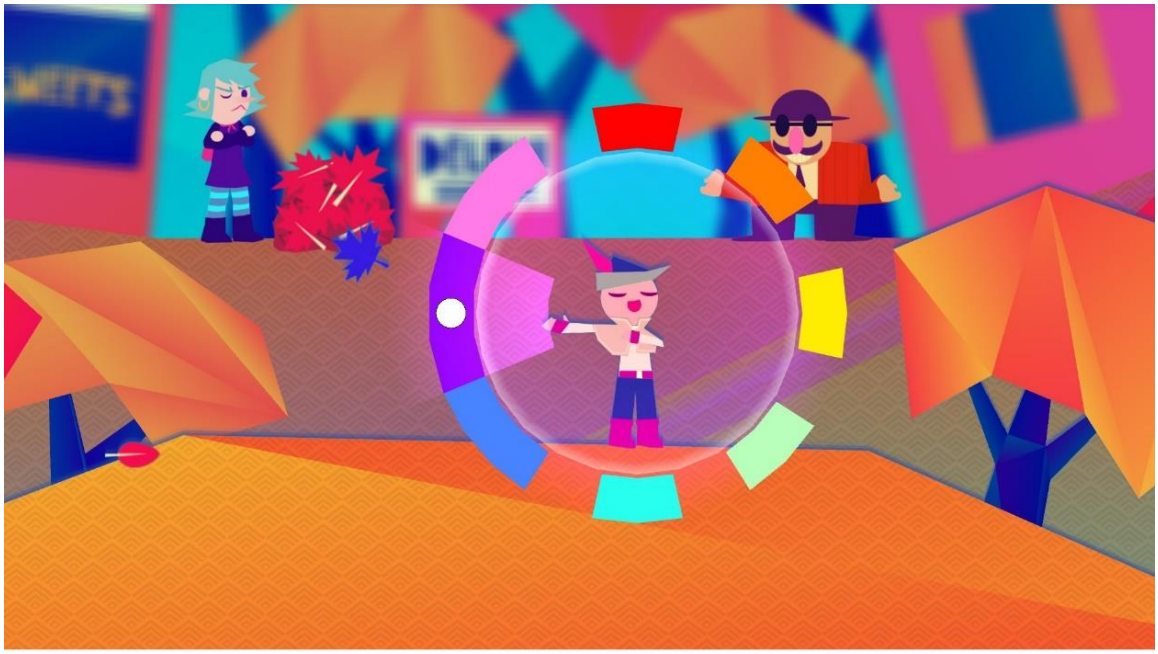

Abbildung 20: Die bunte Spielwelt von Wandersong mit der Hauptfigur und Nebencharaktere (Screenshot aus dem Spiel Wandersong, 2018)

Die Hauptfähigkeit der Spielfigur ist das Singen – dazu wird ein Kreis eingeblendet, der in acht unterschiedlich gefärbte Teile unterteilt ist, wie in Abbildung 20 zu sehen ist. Jeder dieser Teile bildet einen Ton, und zwar reihum jene Töne der C-Durtonleiter. Mit dem rechten Analogstick eines Controllers (oder mit der Maus am PC) lassen sich die Töne auswählen, indem man in die Richtung der farblichen Einzelteile zielt und sie berührt. Der ausgewählte Ton bestimmt auch eine Richtung, in die beispielsweise eine

bewegliche Plattform bewegt werden kann. Der musikalische Aspekt tritt bei dieser Art von Mechanik etwas in den Hintergrund, da die gespielten Töne in diesem Fall keine besondere Bedeutung haben.

Anders ist es an diversen Stellen im Spiel, wo Richtungen durch farbige Partikeleffekte gegeben werden, die es gilt, nachzuspielen. Abbildung 21 zeigt eine solche Spielmechanik. Hier ist ein Rätsel zu lösen, indem man die angezeigten Richtungen entsprechend nachspielt. Das Lösen dieser Rätsel schaltet meist den nächsten Schritt des Spiels frei.

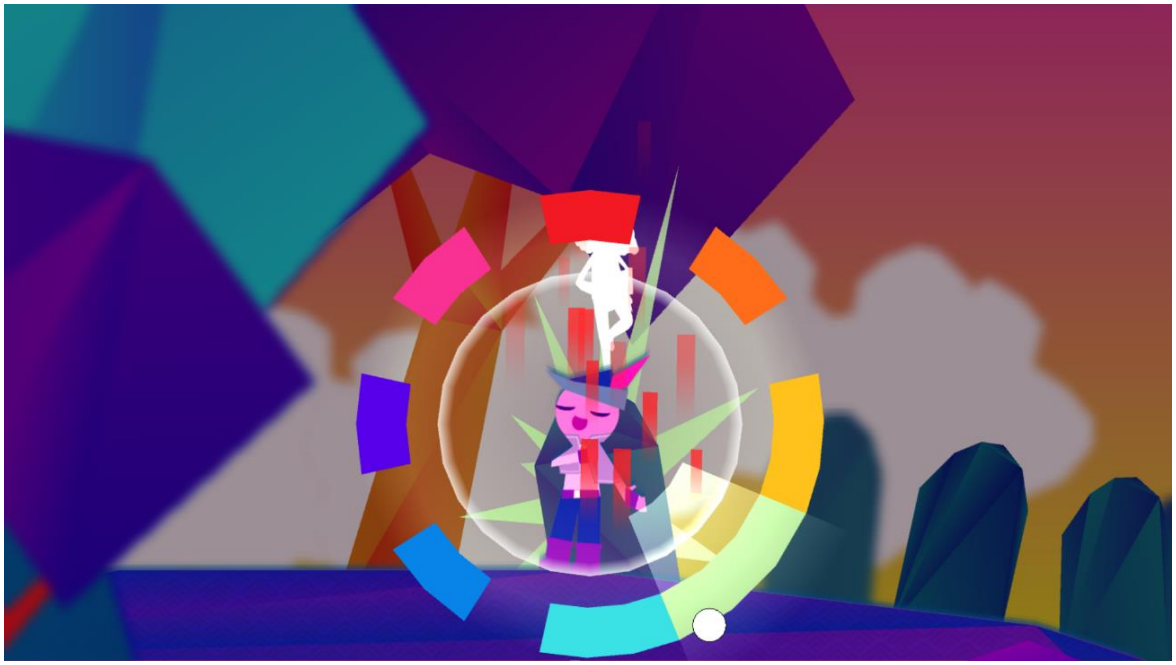

Abbildung 21: Manchmal werden Richtungen zum Nachspielen vorgegeben (Screenshot aus dem Spiel Wandersong, 2018)

Eine weitere Mechanik hat eine gewisse Ähnlichkeit zu der Spielmechanik von Guitar Hero: hier muss die Spielfigur bestimmte Töne zum richtigen Zeitpunkt spielen. In Abbildung 22 sind links die farbigen Balken zu sehen, die auf den Barden zukommen. Dieser muss genau dann die entsprechenden Töne spielen, wenn der Balken den festen farbigen Balken des Kreises trifft. Durch die sich ankündigenden Markierungsbalken hat man etwas Zeit, um den richtigen Ton zu treffen.

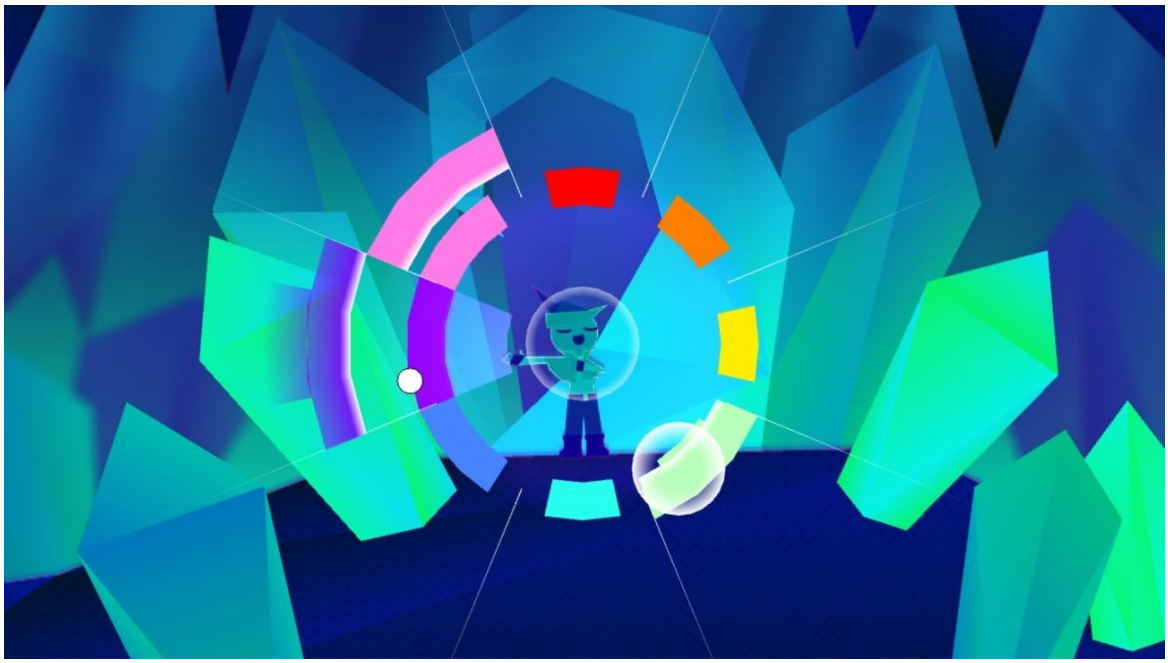

Abbildung 22: Töne zum richtigen Zeitpunkt spielen

Die Möglichkeit, Töne zu singen, ist fast durchgängig gegeben. Wenn der Barde durch die bunte und malerische Landschaft wandert und neben Blumen und Vögeln singt, reagieren diese meist durch Verfärbung in die zuvor ausgewählte Farbe des Tons und bewegen sich zum Rhythmus der Musik – wie es überhaupt sehr viele Spielobjekte, wie zum Beispiel Bäume im Hintergrund tun. Die Spielwelt ist überwiegend sehr positiv, freundlich und optimistisch gehalten. Die Musik ist meist aus simplen Akkordmustern aufgebaut. Insgesamt wird so eine sehr warme und harmonische Atmosphäre geschaffen.

Im Vergleich zu Guitar Hero können hier tatsächlich Töne frei über das Gamepad durch die Spieler*innen getriggert werden. Somit ist eine gewisse Freiheit im Spiel gegeben, wie schon eingangs in Kapitel 3 in Bezug auf das Freiform-Spiel erwähnt wurde. Allerdings werden die Töne nicht mit den entsprechenden Notennamen benannt, sondern nur durch Farben codiert.

3.3 Musikspiel: Rocksmith

Rocksmith ist der Name einer Videospielereihe von Ubisoft, deren erstes Spiel mit dem gleichnamigen Titel 2012 in Europa erschien. 2013 folgte dann *Rocksmith 2014 Edition* und im Jahre 2016 *Rocksmith 2014 Remastered Edition*. Diese Nachfolger bringen neue Features und Verbesserungen in der Stabilität und Performance des Spiels, das Spielprinzip ist aber im Kern gleichgeblieben. Rocksmith erinnert von der Aufmachung her etwas an die Guitar Hero Spielereihe. Es wird als „der schnellste Weg, Gitarre zu lernen“ vermarktet (Ubisoft, 2017).

Abbildung 23: Ähnlicher Ansatz wie bei Guitar Hero (Ubisoft, 2017)

Das Spiel lässt sich nicht mit einem gewöhnlichen Gamepad spielen. Stattdessen wird dafür eine richtige E-Gitarre (bei den Nachfolgern auch E-Bass) benutzt, die über ein spezielles Kabel an die Konsole oder an den PC angeschlossen wird. Dieses so genannte *Real Tone Cable* fungiert im Prinzip wie ein USB-Audiointerface, denn es wandelt die analogen Signale der Gitarre in elektrische um, die dann im Spiel weiterverarbeitet werden. Somit ist Rocksmith im Prinzip ein Lernprogramm für Gitarrist*innen. Die zahlreichen Modi des Spiels haben das übergeordnete Ziel, einen Lerneffekt für die Spieler*innen zu erwirken.

Die Hauptfunktion ist das Erlernen eines Songs, das für die meisten angehenden Gitarrist*innen das wichtigste Feature ist. Wie in Abbildung 23 gezeigt, werden parallel zum laufenden Song die genauen Griffmuster eingeblendet, die sich im Rhythmus auf die Spieler*innen hinzubewegen. Im unteren Teil der Abbildung ist die Silhouette des

Gitarrenhalses zu sehen, deren Saiten ähnlich farblich codiert sind wie bei Guitar Hero. Zur Orientierung werden zum einen links oben der Songtext eingeblendet und zum anderen ganz oben in einer Leiste die aktuelle Position im Song. Die Songs lassen sich in unterschiedlichen Schwierigkeitsgraden spielen – bei den leichteren Graden werden weniger Töne angezeigt und die Geschwindigkeit der *zufliegenden* Töne ist langsamer. Dies steigt mit den höheren Schwierigkeitsgraden an. Bei der höchsten Einstellung werden die originalgetreuen Griffmuster angezeigt.

Es gibt ebenfalls spezielle Modi, bei denen viele Funktionen gegeben werden, um einen Song zu meistern. Beispielsweise lässt sich die Geschwindigkeit händisch herabsenken, um sehr schnelle Abschnitte eines Songs Stück für Stück zu erlernen, bevor man dann diese Abschnitte auf voller Geschwindigkeit spielt. Zudem kann der Schwierigkeitsgrad sich an die eigene Spielweise mit der Zeit dynamisch anpassen. Die so genannten *Technique Challenges* sind Übungen für spezielle Techniken des Gitarrenspiels wie zum Beispiel *hammer-ons* – eine Technik, bei der ein Ton angeschlagen wird und ein zweiter durch festes *Draufhämmern* mit dem nächsten freien Finger zum Klingen gebracht wird, ohne dass dieser extra mit dem Plektrum angeschlagen wird.

Abbildung 24: Guitarcade Modus (Screenshot aus dem Spiel Rocksmith, 2014)

Eine weitere Besonderheit ist der sogenannte *Guitarcade* Modus. Hier werden Minispiele angeboten, die ebenfalls ausschließlich mit der Gitarre gesteuert werden. Abbildung 24 zeigt ein solches Minispiel, bei dem man die Höhe des Wassers der Fontäne steuern kann, um Hindernissen auszuweichen und Bananen einzusammeln. Hier rückt

die Hauptmechanik des Spiels etwas in den Hintergrund, um neue Spielerlebnisse zu ermöglichen – denn um das Erlernen von Songs geht es hier vordergründig nicht.

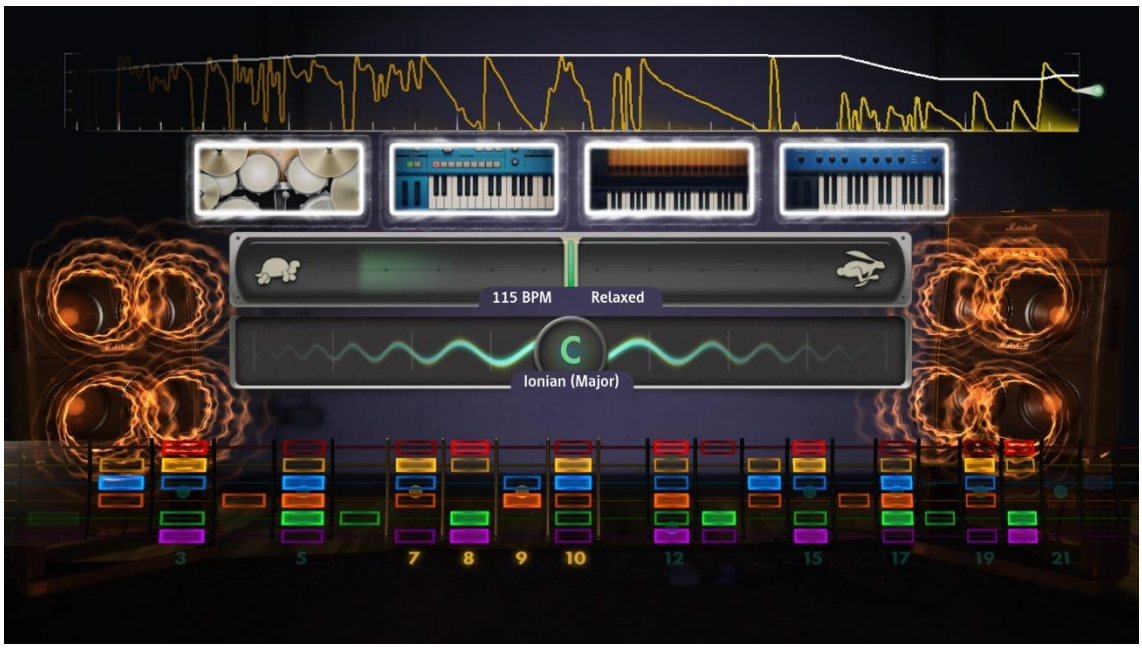

Abbildung 25: Der "Session Mode" (Screenshot aus dem Spiel Rocksmith, 2014)

Das im Eingang dieses Kapitels erwähnte Konzept des *Freiform-Spiels* findet sich sehr gut abgebildet im so genannten *Session Mode* von Rocksmith wieder. Hier kann man sich seine eigene *virtuelle* Band zusammenstellen. Dazu wählt man zunächst aus bis zu vier verschiedenen Instrumenten aus, die in einer eigens festgelegten Tonart *mitspielen*. Zudem lassen sich Tempo, Komplexität der Akkorde und vieles mehr festlegen. In Abbildung 25 ist der Session Mode in Aktion zu sehen. In der Mitte ist die bpm und die Tonart (C-Dur) angegeben. Im unteren Teil ist wieder der Gitarrenhals als Silhouette angezeigt, auf denen die Griffmuster der C-Durtonleiter mit leuchtenden Farben abgebildet sind. Oberhalb der Mitte sind die mitspielenden Instrumente eingeblendet, und ganz oben wird die Lautstärke des eigenen Spiels angezeigt. Die mitspielenden Instrumente, die quasi den Hintergrundsong spielen, passen sich in ihrem Spiel dynamisch an das eigene Spiel an. Spielt man also wenig Töne und sehr langsam, musizieren die übrigen Instrumente in relativ leiser Manier. Legt man allerdings an Geschwindigkeit zu und spielt viele Töne in kürzeren Abständen, intensiviert sich das Spiel aller Instrumente.

3.4 Fazit

Drei verschiedene Spiele im Bereich der Musikspiele wurden unter diversen Gesichtspunkten analysiert. Jedes dieser Spiele benutzt Musik als Kernelement im Gameplay, jedoch auf unterschiedliche Art und Weise. Guitar Hero ist demnach eigentlich kein *richtiges* Musikspiel, sondern eher ein reines Rhythmusspiel, da hier die rhythmische Performance im Vordergrund steht. Durch die fünf farblich codierten Buttons werden keine direkten Töne getriggert, sondern es wird durch das richtige Timing dafür gesorgt, dass die Musik weiterläuft.

Wandersong hingegen bietet mehr Freiheitsgrade durch die ständig verfügbare Singmöglichkeit, getriggert über das Gamepad mit dem rechten Analogstick. Dabei werden alle Töne der C-Durtonleiter benutzt, jedoch ohne Namen der Töne. Die Einstiegshürde ist jedoch sehr gering, da das Spiel mit einem Gamepad gespielt werden kann, das den meisten Spieler*innen von anderen Spielen bekannt ist. Selbst wenn es auf dem PC gespielt werden, wissen die meisten mit Tastatur und Maus umzugehen. Allerdings lässt es sich damit nicht so gut und intuitiv musizieren wie mit einem tatsächlichen Instrument.

Rocksmith wiederum ist ein Spiel, was Musikmachen in Reinform bietet. Es ist im Endeffekt ein Lernspiel für Gitarrist*innen (und Bassist*innen) und somit weniger ein Videospiel im klassischen Sinne und mehr eine Art virtueller und interaktiver Gitarrenunterricht. Im Modus *Guitarcade* wird die Domäne des reinen Musikmachens etwas verlassen und das Spiel bekommt etwas mehr den Videospielcharakter, denn hier werden verschiedene Minispiele angeboten – mit der Besonderheit, dass sie trotzdem mit dem tatsächlichen Musikinstrument gespielt werden.

Wie lässt sich nun die die Reinform des Musikmachens bei Rocksmith und die einfache Zugänglichkeit von Spielen wie Wandersong und Guitar Hero zusammenbringen – anders ausgedrückt: wie schafft man es, eine niedrige Einstiegshürde für ein Spiel zu finden, das jedoch trotzdem möglichst intuitiv und frei ist im Kontext des Musizierens? Dies wird Gegenstand des folgenden Kapitels werden.

4 Konzeption und Umsetzung

Nachdem die Grundlagen von Musik, Musikinstrumenten und Gamepads dargelegt wurden, sowie eine Auswahl existierender Musikspiele untersucht worden sind, sollen nun neue Steuerungsmechaniken konzipiert und implementiert werden. Diese Konzepte werden für das Spiel *Sonority* umgesetzt, dessen Prototyp im Rahmen einer Masterarbeit entstanden ist und darüber hinaus weiterentwickelt wird (Schorrig, 2019, S. 49).

4.1 Musikalisches Rätselspiel: Sonority

Sonority ist ein Rätselspiel, das Musik als Gameplaymechanik einsetzt. Die Protagonistin des Spiels kann mit ihrer Panflöte verschiedene Töne spielen, die in der Spielwelt verschiedene Aktionen auslösen können.

> *Der Spieler schlüpft in die Rolle der jungen Esther, die sich auf die Reise gemacht hat, die Geheimnisse der Musik zu enträtseln. Die Spielwelt reagiert auf die Töne, die Esther auf ihrer Flöte spielt, und verändert sich. Die Lösung der Rätsel sind mit dem Klang einer bestimmten Melodie verbunden.* (Reinaldo et al., 2019, S. 1)

Abbildung 26: Konzeptgrafik eines Levels aus Sonority (Reinaldo et al., 2019, S. 29)

Das Spiel wird aus der isometrischen 3D-Ansicht gespielt. Die Levels bestehen aus verschiedenen Objekten, die sich auf bestimmte Art und Weise verhalten, wenn Töne gespielt werden. Die zur Auswahl stehenden Töne bilden die C-Durtonleiter. Abbildung 26 zeigt ein solches Level als Konzeptgrafik. Mitte links ist die junge Protagonistin Esther mit ihrer Panflöte zu sehen. Vor ihr stehen drei so genannte Musiksteine, die als Platzhalter für Töne dienen. Befindet sich Esther in unmittelbarer Nähe eines Steins und spielt einen beliebigen Ton, merkt sich der Stein diesen Ton. Diese Töne bilden eine Sequenz. Werden allen Steinen Töne zugewiesen, kann durch Betreten des bläulich schimmernden Kreises links neben der Treppe die Sequenz abgespielt werden, indem die entsprechende Taste gedrückt wird (Reinaldo et al., 2019, S. 16). Die Steine leuchten der Reihe nach auf und spielen jeweils ihren zugewiesenen Ton ab. Je nach Beziehung zwischen den einzelnen Tönen werden verschiedene Bewegungen ausgelöst: in diesem Fall wurden bereits die passenden Töne eingegeben, die die Brücke und einen Teil der Treppe in die richtige Lage brachten, damit Esther die obere Plattform erreichen kann und ihren Weg fortführen kann.

Die Rätsel in den Levels dienen somit hauptsächlich dazu, die Hindernisse des Levels zu überwinden, um voranschreiten zu können (Schorrig, 2019, S. 50). So kann beispielsweise eine Plattform, die sich zu weit oben für die Spielfigur befindet, durch eine absteigende Tonfolge herunterbewegt werden. Der Abstand zwischen den Tönen gibt dabei an, wie viele Schritte die Plattform nach unten bewegt wird. Allgemein lässt sich jedes Hindernis durch eine Tonfolge bestehend aus zwei Tönen bewegen. Die Richtung der Bewegung wird dabei durch eine aufsteigende oder absteigende Tonfolge determiniert, die Anzahl der Schritte durch den Abstand dieser Töne (Schorrig, 2019, S. 51).

4.2 Technologie, Anwendungsumgebung

Das Spiel wird mit der Spiele-Engine *Unity3D* realisiert. Unity3D ist eine weit verbreitete Entwicklungsumgebung für 2D- und 3D-Echtzeitanwendungen, deren Applikationen auf mehreren Plattformen lauffähig sein können. Für die Umsetzung der Steuerungskonzepte wird – so wie für das Spiel selbst - die objektorientierte Programmiersprache C# benutzt.

4.2.1 Funktionalität in Unity

Die Spielwelt einer Anwendung in Unity besteht zunächst aus so genannten *GameObjects*, die wiederum verschiedene *Components* haben können. Da Unity eine Spiele-Engine ist und nicht nur eine Entwicklungsumgebung, werden den Entwickler*innen viele in der Spieleentwicklung gebräuchliche Funktionen in Form von Components zur Verfügung gestellt. Eigene Funktionalitäten lassen sich in C#-Skripten programmieren und als Component den GameObjects zuweisen.

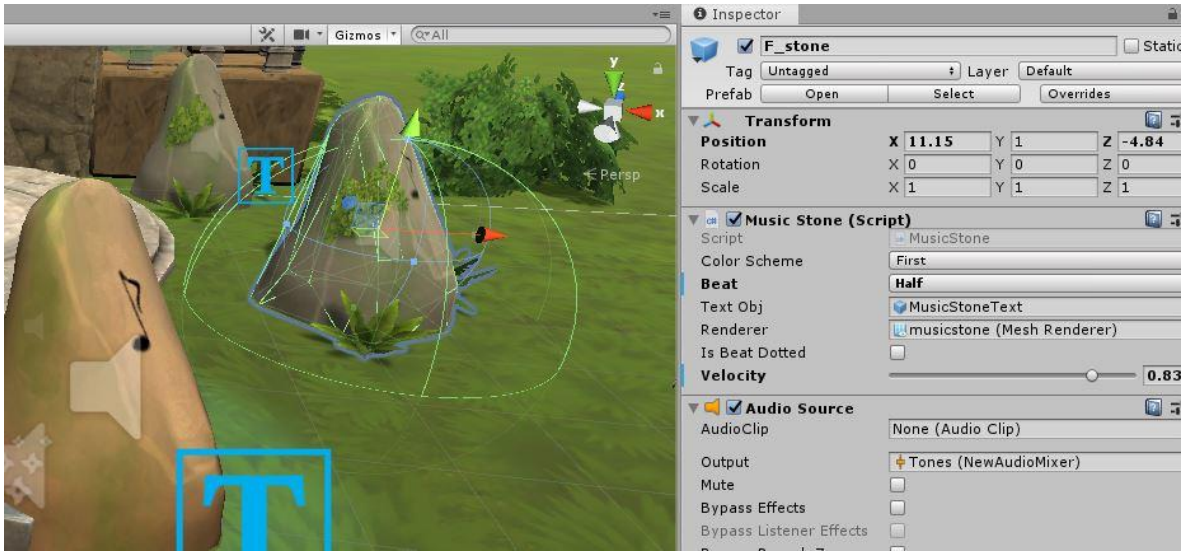

Abbildung 27: Ein Ausschnitt aus dem Editor in Unity (Screenshot vom Programm Unity)

Jedes GameObject hat die *Transform* Component, wie oben rechts in Abbildung 27 zu sehen ist. Sie enthält Informationen wie beispielsweise die Koordinaten der Position in der Spielwelt. Ganz links ist die konzeptuelle grafische Repräsentation eines Musiksteins zu sehen, rechts davon ein Musikstein, der im Editor ausgewählt ist. Dessen Components werden rechts im Inspector angezeigt. Die zweite Component heißt *Music Stone* und ist ein vom Entwickler selbst geschriebenes Script. Die Werte darunter entsprechenden den Instanzvariablen (*public*) der dazugehörigen Klasse. *Audio Source* ist eine weitere Component. Sie ermöglicht dem GameObject eine Audiodatei abzuspielen.

Alle GameObjects sind in so genannten *Scenes* organisiert. In Abbildung 27 ist links ein Teil der geladenen Scene zu sehen mit ihren enthaltenen GameObjects, wie zum Beispiel die Musiksteine, der Strauch oder die Felsen im Hintergrund. Eine Scene kann also ein Level abbilden oder auch das Hauptmenü eines Spiels sein (Unity3D, o. D.).

Ein weiterer wichtiger Teil von Unity sind *Prefabs*. GameObjects können mit all ihren Components, Variablenwerten und Kind-GameObjects als Prefabs gespeichert

werden, die somit wiederverwertbar gemacht werden. Diese Prefabs können dann in anderen Scenes einfach platziert werden. So lassen sich Objekte, deren Funktionalität sehr oft benötigt wird, ganz einfach kopieren und wiederverwenden.

Allgemein werden alle Objekte, die in Projekten benutzt werden können, *Assets* genannt. Diese sind auch als ganze Pakete über den *Asset Store* von Unity zu unterschiedlichen Preisen beziehbar – viele Assets sind auch gratis erhältlich. Verschiedene Entwickler*innen bieten dort Grafiken, Sounds, 2D- oder 3D-Modelle und weitere Assets an (Unity3D, o. D.). Mit den Standard-Assets von Unity und einigen Paketen vom Asset Store wird der Prototyp des Spiels derzeit entwickelt und hat viele Kern-Funktionalitäten bereits implementiert.

4.2.2 MIDI-Player

Für das Abspielen von Tönen in Sonority wird ein externes Asset namens *Midi Player Tool Kit Pro (MPTK)* aus dem Asset Store benutzt. Das MPTK erlaubt es, Musik von MIDI-Dateien in Unity abzuspielen (Bachmann, 2019). Das bedeutet, dass sich eine MIDI-Datei, die ein Musikstück im MIDI-Format enthält, in Unity geladen und abgespielt werden kann. Die Klänge kommen jeweils von so genannten *SoundFonts*. Eine SoundFont ist vergleichbar mit einer virtuellen Bibliothek in Form einer Datei, die viele Samples von verschiedenen Musikinstrumenten enthält. Es können auch eigene SoundFonts erstellt und mit dem MPTK benutzt werden. Was jedoch im vorliegenden Projekt eher gebraucht wird, ist folgende Funktionalität: das Toolkit erlaubt auch direktes Spielen im Spiel, in dem es MIDI-Events liest, die von einem Skript gesteuert werden.

„Tabelle“ 4 zeigt einen kleinen Ausschnitt aus dem Programmcode der Klasse `NotePlayer`, und zwar die Methode `PlayNote()`. Die Klasse `NotePlayer` ist allgemein zuständig zum Abspielen von Tönen anhand von MIDI-Befehlen. Die Methode nimmt als ersten Parameter einen `enum`-Wert vom Typ `NoteEvent.Note`, als zweiten Parameter einen Integer-Wert namens `octave`. In `NoteEvent.Note` sind alle Töne von C bis C2 als Zahlenwerte von 1-8 definiert. Wenn die Oktave drunter oder drüber gespielt werden soll, kann hier für `octave` entweder -1 oder 1 übergeben werden; bei 0 wird keine Oktave gespielt. Der Übersichtlichkeit halber wurde kurz nach der Methodensignatur ein Stück Code weggelassen und mit „[…]“ abgekürzt.

Tabelle 4: Programmcode: Methode zum Abspielen einer Note mit dem MPTK (Hinweis: der Einfachheit halber wird Programmcode als Tabelle dargestellt, damit dieser auch im Tabellenverzeichnis gelistet wird.)

```
    /**
     * Plays given note with optional octave.
     * Also takes beat, bpm and instrument as parameters.
     *
     * */
    public void PlayNote(NoteEvent.Note note, int octave, NoteEvent.Beat beat,
float beatsPerMinute, MidiInstrument instrument)
    {

        […]

        int octaveFactor = 12 * octave;

        float duration = ((60000f / beatsPerMinute) * 4) / ((int)beat);
        noteEvent = new MPTKEvent()
        {
            Command = MPTKCommand.NoteOn,
            Value = 59 + octaveFactor + NoteEvent.GetSemiTone(note),
            Channel = 0,
            Duration = (long)duration,
            Velocity = 100
        };
        midiPlayer.MPTK_PlayEvent(noteEvent);
    }
```

Die Klasse `MPTKEvent` ist vom MPTK und wird mithilfe einer anonymen Klasse initialisiert, worin die gewünschten Werte direkt gesetzt werden können. Gleich am Anfang ist der entsprechende MIDI-Befehl zu sehen: `Command = MPTKCommand.NoteOn`. Dies entspricht einem *Note On* MIDI-Befehl, das den Start einer Note veranlasst. Dazu wird der Wert in `Value` übergeben: die Zahl 59 kommt deshalb zustande, weil der Wert 60 einem mittleren C entspricht – wenn nämlich ein C übergeben wird, das dem `enum`-Wert 1 (vom Typ `NoteEvent.Note`) entspricht, wird hier 59 + 1 = 60 gerechnet. Falls ein Oktavwert mitgegeben wurde, wird dieser hier ebenfalls dazu addiert. Der letzte Methodenaufruf `midiPlayer.MPTK_PlayEvent(noteEvent);` schickt diese Werte dann an den `midiPlayer`. Diese Klasse stammt vom MPTK und ist zuständig für das tatsächliche Abspielen der Töne innerhalb der Unity-Umgebung.

4.3 Steuerung

Die spielbaren Töne sind ähnlich wie bei Wandersong (Kapitel 3.2) in einer Art Ringmenü angeordnet, die über den rechten Analogstick ausgewählt werden können. Das Ringmenü erscheint erst durch Betätigen des Analogsticks in eine Richtung. Durch Drücken der RB-Taste (bzw. R1-Taste) ertönt der ausgewählte Ton aus der Panflöte (*Hinweis: Hier wird ein Xbox Controller verwendet*). In Abbildung 28 ist das Ringmenü als blauer Kreis dargestellt, das aus acht gleichgroßen Teilen besteht. Jedem Ausschnitt ist ein Ton zugeordnet.

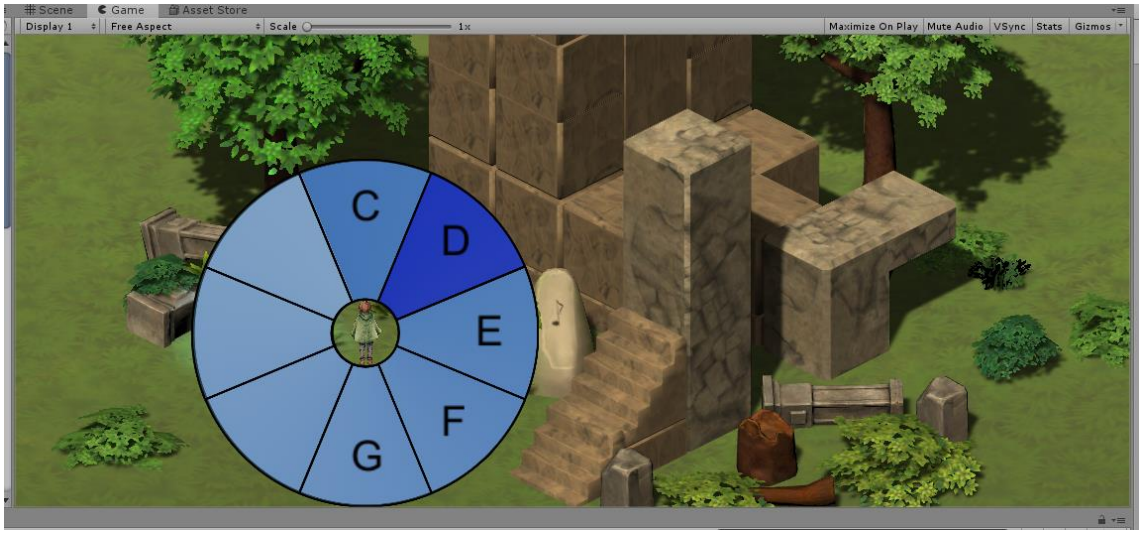

Abbildung 28: Das Ringmenü zum Auswählen der Töne, ähnlich dem vom Wandersong, Kapitel 3.2 (Screenshot aus Unity, Prototyp von Sonority)

Am PC lässt sich das Spiel ebenso mit der Tastatur steuern. Hier werden die Töne durch die alphanumerischen Tasten *1* bis *8* ausgelöst. Diese Steuerung ist vorerst als Fallback-Mechanismus gedacht und soll außerdem zum unkomplizierten Debuggen (Fehlersuche) dienen – wenn also kein Controller zur Hand ist.

Die Steuerung deckt sich etwas mit jener von Wandersong, allerdings ist bei Sonority ein zusätzlicher Schritt nötig, bis der Ton erklingt: zuerst wird er Ton durch über das Ringmenü ausgewählt, dann durch Drücken der RB-Taste (R1 auf dem PlayStation Controller) abgespielt. Bei Wandersong konnte der Ton direkt durch das Steuern des rechten Analogsticks angespielt werden.

4.3.1 Implementierung

Tabelle 5 zeigt die Methode, die die Steuerung des rechten Analogsticks abfragt und daraus berechnet, welcher Ton ausgewählt wurde. Diese Methode wird innerhalb der `Update()`-Methode aufgerufen, die in Unity standardmäßig jeden Frame ausgeführt

wird. (Hinweis: auch hier wurden Teile des Codes der Übersichtlichkeit halber mit „[…]“ gekürzt.) Zunächst werden sowohl die Werte der X-Achse mit `Input.GetAxis("Vertical Right");` als auch die der Y-Achse mit `Input.GetAxis("Horizontal Right");` extrahiert und respektive in den Variablen `x` und `y` gespeichert. Damit wird dann die Länge des Ausschlags des Sticks berechnet mit dem Satz des Pythagoras:

$$x^2 + y^2 = \text{length}$$

Der Ausschlag wird deshalb berechnet, weil sonst bei der kleinsten Bewegung des rechten Sticks die Funktion ausgelöst wird. Deshalb gibt es die statische Instanzvariable `MIN_AXIS_DISTANCE`, die in diesem Fall `0.3f` beträgt (Hinweis: der Wert des Ausschlags kann zwischen 0 und 1 liegen). Wenn der Ausschlag also über 0,3 ist, wird zunächst der Winkel des Ausschlags mithilfe der Funktion `VectorAngle(x, y);` berechnet, die einen Wert zwischen 0 und 360 ausgibt.

Tabelle 5: Programmcode: Abspielen von Tönen mit einem Analogstick

```
/**
 * Notes are mapped to one analog stick,
 * visualized in a circle with 8 segments,
 * one for each note.
 *
 **/
private void PlayNotesWithOneStick()
{
    float x = Input.GetAxis("Vertical Right");
    float y = Input.GetAxis("Horizontal Right");
    […]
    float length = Mathf.Sqrt((Mathf.Pow(x, 2) + Mathf.Pow(y, 2)));

    if (length > MIN_AXIS_DISTANCE)
    {
        […]
        float degree = VectorAngle(x, y);
        int panelNumber = getMappedDirection(degree);

        if (playableNotes[panelNumber] == true)
        {
            […]
            if (Input.GetKeyDown("joystick button 5"))
            {
                timePassed = 0;
                NoteEvent.Note note = (NoteEvent.Note)panelNumber + 1;
                EventManager.Instance().TriggerNoteEvent(note);
                notePlayer.PlayEndlessNote(note, midiPatch, noteSustain);
            }
        }
    }
[…]
}
```

Anhand des Winkels lässt sich mit der Funktion `getMappedDirection(degree)` ausrechnen, in welchem Segment sich der Ausschlag des Sticks befindet. Wenn das Ergebnis ein Wert zwischen 0 und 7 hat, wird mit `Input.GetKeyDown("joystick button 5")` abgefragt, ob die RB-Taste gedrückt wurde. Anschließend wird ein `enum` vom Typ `NoteEvent.Note` mit der um 1 erhöhten Variable `panelNumber` initialisiert – diese hat ja derzeit den Wertebereich 0 bis 7, doch das `enum` erwartet Werte zwischen 1 und 8. Das `enum` wird dann an den `EventManager` weitergereicht, der dafür zuständig ist, den gespielten Ton an Objekte weiterzureichen, die sich für diese Events *registriert* haben – zum Beispiel die Musiksteine. Schließlich wird mit der Instanz des `NotePlayers` der Ton abgespielt.

4.3.2 Evaluation

In der Arbeit von Schorrig wurde eine Evaluation anhand einer frühen Version des Prototyps durchgeführt, bei der die Tester*innen ebenfalls zu ihrer Erfahrung beim Spielen von Tönen gefragt wurde. Konkret wurde gefragt, ob die Spieler*innen das Gefühl hatten, „während de[m] Spiele[n] selbst zu musizieren“ (Schorrig, 2019, S. 77). Schorrigs Hypothese war dabei, dass das „direkte Bewegen von Objekten“ sich mehr nach Musizieren anfühle als das bloße Zuweisen von Tönen (Schorrig, 2019, S. 77). Unter den Tester*innen befanden sich sowohl Musiker*innen und Nicht-Musiker*innen als auch Gamer*innen und Nicht-Gamer*innen (Schorrig, 2019, S. 96). Für den Test wurden insgesamt zwölf Level prototypisch erstellt, die in ihrer Schwierigkeit aufsteigen und möglichst isoliert einen zuvor als Kernfrage formulierten Aspekt behandeln (Schorrig, 2019, S. 78).

*Abbildung 29: Bewertung der Tester*innen zur Möglichkeit, Töne selbst zu spielen (Schorrig, 2019, S.99)*

Zunächst fanden die Tester*innen die Möglichkeit, Töne selbst zu spielen, überwiegend positiv, wie in Abbildung 29 zu sehen ist. Abbildung 30 zeigt jedoch diesmal ein divergierendes Meinungsbild: insgesamt neun Personen empfanden die Steuerung eher intuitiv, zehn hingegen eher wenig intuitiv.

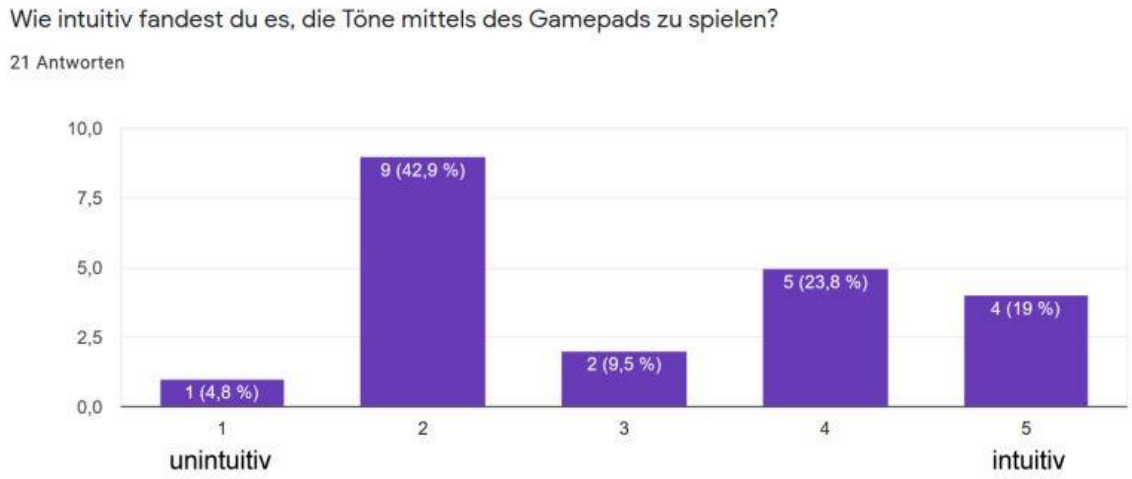

*Abbildung 30: Bewertung der Tester*innen zur Eingabe der Töne mittels des Gamepads (Schorrig, 2019, S. 99)*

In der Auswertung der Ergebnisse im Hinblick auf die musikalische Aktivität und Vorerfahrung der Testpersonen kommt Schorrig zu folgendem Schluss: „Wer oft selbst musiziert, hat überhaupt nicht das Gefühl, dies auch beim direkten Spielen von Tönen im Spiel zu tun“ (Schorrig, 2019, S. 108). Im Verlauf der Evaluation merkte eine Testperson an, wieso sie nicht das Gefühl hatte, zu muszieren: die Lösungsmelodien seien vorgegeben. Musizieren sei für die Person damit verbunden, sich „selbst kreativ entfalten zu können“ (Schorrig, 2019, S. 110). Die zuvor gewünschte, als Hypothese geäußerte Aesthetic im Kontext des MDA-Frameworks wurde also nur unzureichend bei den Rezipienten erreicht. Ausgehend von dieser Bemerkung, die mitten in der Evaluationsphase kam, wurden die darauffolgenden Testpersonen ebenfalls nach ihrer Einschätzung diesbezüglich gefragt; die meisten stimmten dieser Aussage zu. Auf Basis dieser Beobachtungen sollen nun im Folgenden alternative Steuerungskonzepte erarbeitet werden.

4.4 Konzeption neuer Steuerungsmechaniken

Bei der aktuellen Steuerung mit dem Ringmenü fällt beim Spielen auf, dass die einzelnen Segmente des Kreises teilweise schwer ansteuerbar sind. Töne, die genau in den Himmelsrichtungen liegen, sind sehr leicht auswählbar; jene zwischen den Himmelrichtungen sind jedoch aufgrund der Steuerung mit dem Analogstick etwas unpräziser zu erreichen. Aufgrund dieser Problematik entstand zunächst die Idee, die acht Töne nicht auf einen Stick, sondern auf beide Sticks zu legen.

4.4.1 Töne auf zwei Analogsticks

Durch die Aufteilung von acht Tönen auf zwei Analogsticks erhält man pro Auswahlkreis vier Segmente, die jeweils größer als die Segmente in einem Kreis sind. Damit ist es einfach, mit den Analogsticks in eins der Segmente zu *zielen*. Außerdem soll der Ton direkt getriggert werden und nicht durch zusätzliches Drücken einer Taste.

4.4.1.1 Implementierung

In Tabelle 6 ist der dazugehörige Code der Methode dargelegt. Ganz oben ist eine grobe visuelle Repräsentation der Verteilung der Segmente als Code-Kommentar dargestellt. Der Code ist ähnlich aufgebaut wie jener von der vorherigen Methode, mit Unterschied, dass diesmal die Berechnungen auch für den linken Analogstick durchgeführt werden. Zudem wurde das Triggern der Note über den `EventManager` und `NotePlayer` in der Methode `PlayNoteFromStick(currentNote)` ausgelagert, damit kein doppelter Code entsteht.

Tabelle 6: Programmcode: Triggern der Töne mit zwei Analogsticks

```
/** Notes are mapped to both analog sticks:
    * (currently)
    *     C               G
    * F       D       C2      A
    *     E               H
    */
private void PlayNotesWith2Sticks()
{
    float xL = Input.GetAxis("Vertical");
    float yL = Input.GetAxis("Horizontal");
    float xR = Input.GetAxis("Vertical Right");
    float yR = Input.GetAxis("Horizontal Right");

    float lengthL = Mathf.Sqrt((Mathf.Pow(xL, 2) + Mathf.Pow(yL, 2)));
    float lengthR = Mathf.Sqrt((Mathf.Pow(xR, 2) + Mathf.Pow(yR, 2)));

    if (lengthL > MIN_AXIS_DISTANCE || lengthR > MIN_AXIS_DISTANCE)
    {
        float degreeL = VectorAngle(xL, yL);
        float degreeR = VectorAngle(xR, yR);

        // which quarter
        int panelNumberL = getMappedDirectionQuarter(degreeL);
        int panelNumberR = getMappedDirectionQuarter(degreeR);

        if (playableNotes[panelNumberL] == true || playableNotes[panelNumberR] ==
true)
        {
            if (lengthL > MIN_AXIS_DISTANCE)
            {
                PlayNoteFromStick(panelNumberL + 1);
            }
            else if (lengthR > MIN_AXIS_DISTANCE)
            {
                PlayNoteFromStick(panelNumberR + 5);
            }
        }
     } else
     {
        lastPlayedNote = 0;
     }
}
```

Da diesmal der Ton direkt durch das Aussteuern des Sticks getriggert wird und nicht durch eine extra Taste, ist hier ein weiterer Mechanismus nötig, damit die Töne nicht pausenlos klingen. Dazu wird die zuletzt gespielte Note in der Variable `lastPlayedNote` abgespeichert. Wenn also eine Note gespielt wird, wird diese Variable mit der aktuell gespielten Note besetzt; in der Methode `PlayNoteFromStick(currentNote)` wird vor dem tatsächlichen Abspielen abgefragt, ob die aktuelle Note ungleich der `lastPlayedNote` ist. Erst wenn dies zutrifft, wird der Ton über den `NotePlayer` abgespielt. Werden die Sticks nicht mehr betätigt – ist also

der Ausschlag der Sticks unter der `MIN_AXIS_DISTANCE` – wird `lastPlayedNote` auf 0 gesetzt.

4.4.1.2 Bewertung

Beim Testen dieser Steuerung fällt direkt auf, dass zwangsläufig zusätzlich eine Art *Musikmodus* zusätzlich nötig ist, da der linke Analogstick eigentlich zur Steuerung der Protagonistin benutzt wird. Es müsste also immer zwischen Spielen von Tönen und Bewegung des Charakters hin und her gewechselt werden, was wiederum eine zusätzliche Hürde beim Spielen darstellt und den Spielfluss auch unterbricht. Das Spielen von Tönen an sich jedoch funktioniert besser als bei der vorherigen Steuerung mit einem Analogstick, da jedes Segment größer ist und damit besser ansteuerbar. Denkbar wäre hier auch eine gleichzeitige Steuerung der Velocity eines Tons durch die Stärke des Ausschlags des Sticks. Töne könnten dann durch leichtes Betätigen des Sticks leise, bei stärkerem Betätigen lauter gespielt werden, um so eine Dynamik ins Spiel zu bringen.

4.4.2 Ein Button pro Ton

Eine weitere Überlegung fußt auf der Idee, Töne ohne Analogsticks und damit nur mit Buttons zu spielen. Alle acht Töne der C-Durtonleiter bekommen jeweils ihren eigenen Button auf dem Controller zugewiesen. Die Idee lässt sich auf die Verteilung der Töne auf einem Klavier zurückführen (s. Abbildung 4). Nach etwas Ausprobieren stellte sich folgende Tastenbelegung als zunächst Sinnvollste heraus:

Tabelle 7: Tastenbelegung der einzelnen Töne

C	D	E	F	G	A	H	C2
LT (Left Trigger)	LB (Left Bumper)	D-Pad L (Directional Pad Left)	D-Pad Down (Directional Pad Down)	A	B	RB (Right Bumper)	RT (Right Trigger)

Zur besseren Veranschaulichung ist in Abbildung 31 ein Xbox 360 Controller abgebildet, der von einem halbkreisförmig verlaufenden Pfeil durchzogen ist. Dies soll die Reihenfolge der Töne untermauern, damit sie sich leichter gemerkt werden kann.

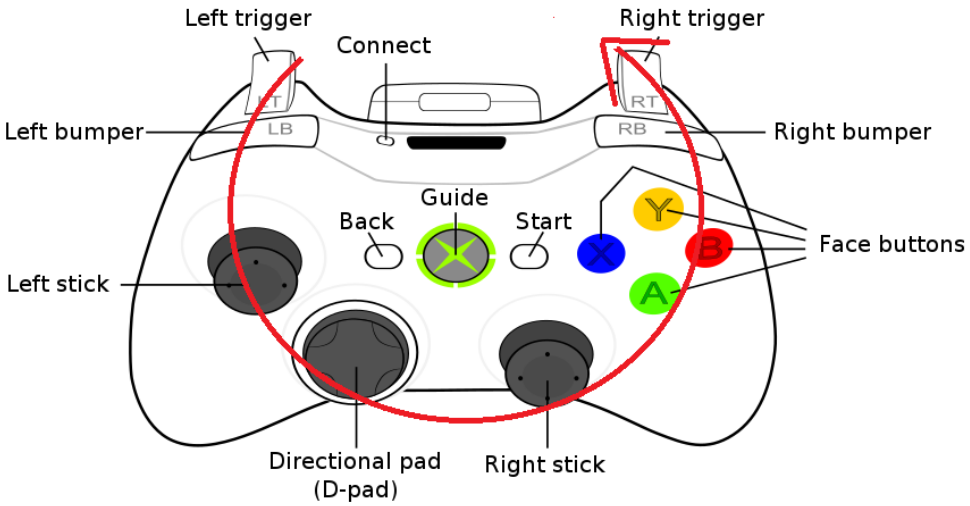

Abbildung 31: Tastenbelegung auf dem Xbox 360 Controller (eigene Darstellung, in Anlehnung an: Alphaton, Wikimedia Commons, 2010)

4.4.2.1 Implementierung

Die Implementierung dieser Steuerung ist im Vergleich zu den vorigen Steuerungen etwas anders. Tabelle 8 zeigt die Methode `PlayNotesWithButtons()`. Zunächst wird wieder ein `NoteEvent.Note` definiert und mit 0 initialisiert. Anschließend wird für jeden Button abgefragt, ob dieser gedrückt wurde und falls ja, ihre zugewiesene Note in der zuvor deklarierten Variable `note` abgespeichert. Auch hier wurde aus Gründen der Übersichtlichkeit einiges an sich wiederholendem Code weggelassen, um den Rahmen nicht zu sprengen. Bei einzelnen Buttons wie A, B, X, Y, LB und RB konnte über die Methode `Input.GetKeyDown(key)` abgefragt werden, ob diese gedrückt werden oder nicht. Anders ist dies bei den Tasten RT, LT, D-Pad Links und D-Pad Down. Diese sind per Definition keine Buttons, sondern werden über eine Achse gemessen.

Daher wird hier die Methode `Input.GetAxis(axis)` benutzt, die wie bei den Analogsticks einen Wert zwischen 0 und 1 liefert. Um das Drücken des D-Pads in die linke Richtung festzustellen, wird also mit `if (Input.GetAxis("DPad X") == -1 && !isDPadLeft)` gefragt, ob das D-Pad in der X-Achse nach ganz links ausgefahren ist, also den Wert -1 liefert. Zusätzlich soll die Abfrage nur dann erfolgreich sein, wenn das D-Pad aktuell nicht nach links gedrückt wird. Dies verhindert wiederum ein ständiges Auslösen der Note. Sind beide Bedingungen erfüllt, wird anschließend die entsprechende Note zugewiesen und die Variable `isDPadLeft` auf `true` gesetzt.

Tabelle 8: Programmcode: Ein Button pro Ton

```
private void PlayNotesWithButtons()
{
    NoteEvent.Note note = 0;
    if(Input.GetKeyDown("joystick button 0")) // A
    {
        note = NoteEvent.Note.G;
    }
    […]
    else if (Input.GetAxis("DPad X") == -1 && !isDPadLeft) // D-Pad Left
    {
        note = NoteEvent.Note.E;
        isDPadLeft = true;
    }
    […]
    else if (Input.GetAxis("Left Trigger") < -0.3 && !isTriggerLeft) // LT
    {
        note = NoteEvent.Note.C;
        isTriggerLeft = true;
    }
    […]
    if (note != 0)
    {
        lastPlayedNote = note;
    }
    // reset controls and note if it's lastPlayedNote
    if (Input.GetAxis("Left Trigger") > -0.3)
    {
        isTriggerLeft = false;
        if (lastPlayedNote == NoteEvent.Note.C)
        {
            notePlayer.StopEndlessNote();
        }
    }
    […]
    if (Input.GetAxis("DPad X") == 0)
    {
        isDPadLeft = false;
        if (lastPlayedNote == NoteEvent.Note.E)
        {
            notePlayer.StopEndlessNote();
        }
    }
    […]
    if (Input.GetKeyUp("joystick button 0") && lastPlayedNote == NoteEvent.Note.G)
    {
        notePlayer.StopEndlessNote();
    }
    […]
    if (note != 0)
    {
        EventManager.Instance().TriggerNoteEvent(note);
        PlayPlayerNote(note, noteSustain);
    }
}
```

Da die Trigger ebenfalls über Achsen verlaufen, wird hier ebenfalls über die `GetAxis`-Methode der Wert erfragt. Hier hat sich der Wert 0,3 als Schwellenwert als praktikabel herausgestellt; bei 0,1 hatte man nicht das Gefühl, eine Taste gedrückt zu haben, da bei der kleinsten Bewegung des Triggers der Ton sofort erklungen ist. Bei -1 müsste

man den Trigger erst ganz durchdrücken, bis der Ton erklingt, was sich nicht als besonders praktikabel erwiesen hat. Damit sich die Trigger so wie die Tasten des D-Pad ebenfalls wie Buttons verhalten, wurde auch hier eine Boolean-Variable eingeführt, um zu verhindern, dass beim Drücken der Ton ständig getriggert wird.

Wurden alle Tasten abgefragt und eine Note erfolgreich in `note` gespeichert, wird diese Referenz zunächst in `lastPlayedNote` abgespeichert. Anschließend folgt eine Reihe von `if`-Statements, die abfragen, ob die Buttons jeweils wieder losgelassen wurden. Bei den Triggern und dem D-Pad bedeutet dies, ob die Werte jeweils unter 0,3 respektive bei 0 liegen. Die dazugehörigen Boolean-Variablen werden auf `false` gesetzt. Falls die `lastPlayedNote` dabei dem jeweiligen Button zugewiesenen Ton entspricht, soll die aktuell gespielte Note mit `notePlayer.StopEndlessNote();` gestoppt werden, damit der Ton nicht ewig erklingt. Das Ganze wird auch für die einzelnen Buttons wiederholt. Zum Schluss wird, falls der Wert in `note` nicht 0 ist, der Ton wie üblich abgespielt.

4.4.2.2 Bewertung

Diese Anordnung fühlte sich beim Testen etwas intuitiver an und ist näher an einem Musikinstrument. Dies liegt vermutlich daran, dass jedem Ton dediziert eine Taste zugewiesen ist, ähnlich wie bei einem Piano. Während der Implementierung und Ausprobieren wurden erste kurze, informelle Tests mit insgesamt fünf anwesenden Personen durchgeführt. Alle Personen präferierten das Steuerungskonzept mit der hier genannten Tastenbelegung.

Im Vergleich zur vorherigen Steuerung ist bei diesem Konzept zunächst keine Dynamik möglich; beziehungsweise nur bei den Triggern, da diese eine gewisse Spannweite haben. Das D-Pad ist zwar als zwei Achsen definiert, allerdings ist der Spielraum zwischen 0 und 1 so klein, dass kaum eine feine, kleinschrittige Bewegung möglich ist - sie wirken eher wie Buttons, die zwei Status haben. Dadurch, dass die Töne auf Buttons liegen, sind jedoch beide Sticks wieder frei zur Verfügung. Der linke Stick wird also weiterhin für die Bewegung der Spielfigur benutzt. Der rechte Analogstick könnte nun dafür benutzt werden, die Velocity der Töne zu steuern. Denkbar wäre die Möglichkeit, dass Töne erst relativ leise anfangen und erst durch Betätigen des rechten Sticks lauter werden in Relation zum Ausschlag des Sticks.

Während dem Spielen ergab sich das Problem, dass beim Drücken des D-Pads in die untere Richtung hin und wieder die Taste in die linke Richtung ebenfalls getriggert

wurde. Dies kann aber auch an dem Xbox Controller liegen – eventuell ist dies bei Controllern anderer Hersteller durch ihre individuelle Bauweise nicht so.

4.5 Freiform-Spiel

Um die neu konzipierten Steuerungen zu testen und auszuprobieren, wurde beispielhaft eine neue *Scene* in Unity gebaut, die ein Freiform-Spiel ermöglichen soll, wie es bereits in Kapitel 3 erläutert wurde. Damit die grundlegenden Spielmechaniken vorhanden, bedarf es einigen Objekten im Spiel.

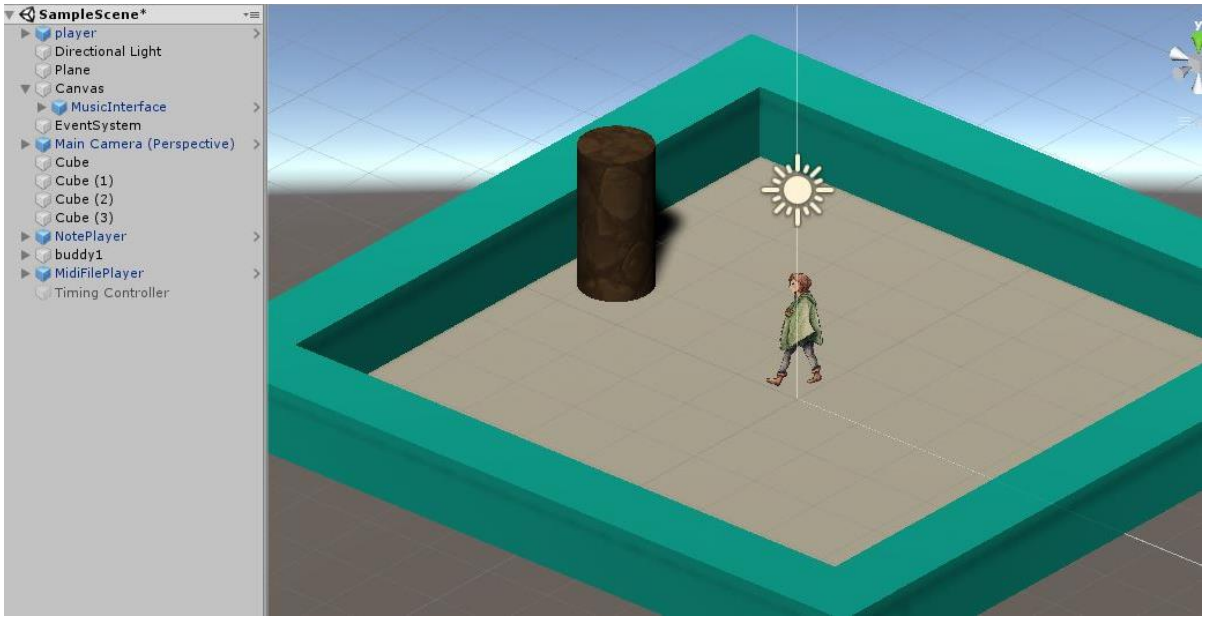

Abbildung 32: SampleScene in Unity zum Testen der neuen Steuerungen

Abbildung 32 zeigt die Scene im Unity-Editor. Aus Zeitgründen wurde hier auf eine aufwendige Ausgestaltung des Levels verzichtet, um sich auf die Funktionalitäten zu konzentrieren. Links sind alle in der Scene enthaltenen GameObjects aufgelistet. Blau gefärbte Listeneinträge sind dabei Prefabs. Alle sichtbaren Objekte bewegen sich auf einer *Plane*, also einer flachen Oberfläche, die den Boden abbildet. Darauf ist der *Player Prefab* abgestellt, aktuell noch als 2D-Konzeptgrafik, da sich die grafische Realisierung der Spielfigur noch in Arbeit befindet.

4.5.1 Auswahl der Steuerungskonzepte

Abbildung 33: Das `MusicInterface`-Component mit der Auswahlmöglichkeit der Steuerung

Die Klasse `MusicInterface` (Hinweis: aktuell heißt die Klasse fälschlicherweise noch `MusikInterface`) ist hier als Component eingebunden. Der Wert der öffentlichen Instanzvariable `selectedControllerMode` kann hier über eine Liste angepasst werden. Zur Auswahl stehen die im `enum ControllerMode` definierten Werte: `OneStick, TwoStick, OnlyButtons.` Der Code in Tabelle 9 zeigt die dazugehörige Abfrage in der `Update`-Methode der Klasse. `CheckKeyboardInput()` kümmert sich um die Eingaben über die Tastatur – diese Funktionalität ist, wie weiter oben schon erwähnt, immer gegeben, um einfach und schnell debuggen zu können.

Tabelle 9: Programmcode: Update-Methode der Klasse MusicInterface:

```
// Update is called once per frame
void Update()
{
    […]

    // Debug: play C to C2 via Number Keys 1-8:
    CheckKeyboardInput();

    switch (selectedControllerMode)
    {
        case ControllerMode.OneStick:
            PlayNotesWithOneStick();
            break;
        case ControllerMode.TwoStick:
            PlayNotesWith2Sticks();
            break;
        case ControllerMode.OnlyButtons:
            PlayNotesWithButtons();
            break;
        default:
            PlayNotesWithOneStick();
            break;
    }
}
```

Die restlichen Eingabemöglichkeiten wurden hier über ein `switch-case`-Statement abgefragt. Als `default`-Fall ist hier die bisher eingesetzte Steuerung mit einem Stick deklariert, um keine potenziellen Fehler im bisherigen Code zu auszulösen.

4.5.2 Buddy-Stein

Im Zuge der Umsetzung der Steuerungskonzepte wird zusätzlich eine weitere, bisher nur theoretisch angedachte Funktionalität des Spiels betrachtet und zumindest rudimentär umgesetzt: so genannte *Buddy-Steine* (Buddy = Kumpel, Freund). Sie ähneln den bereits bekannten Musiksteinen, sollen jedoch personifizierter sein, da sie eine Art Gesicht besitzen und sogar vielleicht sprechen können sollen. Die Steine sind immer wieder mal in Levels anzutreffen und sollen Esther auf ihrer Reise auf verschiedene Art und Weise unterstützen. Die Protagonistin kann mit den Steinen interagieren, indem sie in ihrer Nähe Töne spielt. In der Scene in Abbildung 32 ist ein Buddy-Stein aktuell als einfacher Zylinder mit dunkler Textur dargestellt und wird im Laufe der Entwicklung angepasst.

Abbildung 34: Buddy-Stein Component mit auswählbarem Modus

Zum Buddy-Stein gehört eine Klasse namens `BuddyStone`, die dessen Funktionalität enthält. In Abbildung 34 ist das dazugehörige Component zu sehen; der Buddy-Stein hat einen `buddyMode`, mit dem bestimmt werden kann, wie genau der Stein auf Töne reagiert. Die exemplarisch umgesetzten Modi der Buddy-Steine sind folgende:

- `LowOctave`: Der Buddy-Stein reagiert mit der tiefen Oktave.
- `PerfectFifth`: Der Buddy-Stein spielt die Quinte.
- `DiatonicThird`: Der Buddy-Stein spielt die diatonische Terz. Diatonisch deshalb, da zum aktuellen Zeitpunkt nur die C-Durtonleiter spielbar ist und die chromatische Terz bei bestimmten Tönen dissonant klingt.

Damit der Stein registriert, dass die Spielfigur einen Ton gespielt hat, muss sich dieser Stein dafür am in Kapitel 4.3.1 erwähnten `EventManager` anmelden. Dies wird in der `Start()`-Methode der Klasse `BuddyStone` mit dem Aufruf von `EventManager.Instance().RegisterForNoteEvent(NotePlayed);` durchgeführt.

Dabei ist zu beachten, dass `NotePlayed` keine Variable ist, sondern ein Zeiger zur Methode `NotePlayed(note)`, die in Tabelle 10 dargestellt ist. In dieser Methode wird zunächst abgefragt, ob die Spielfigur in der Nähe ist. Die Boolean-Variable `isPlayerNear` wird dann auf `true` gesetzt, wenn die Spielfigur einen so genannten Collider trifft, der um den Buddy-Stein herum platziert ist. Collider liegen meistens an Objekten an, um zu verhindern, dass die Spielfigur durch sie durchlaufen kann, indem die Kollision abgefragt wird. In diesem Fall fungiert der Collider nur als Trigger, um festzustellen, dass die Spielfigur einen bestimmten Abstand zum Stein erreicht hat.

Tabelle 10: Programmcode: Methode NotePlayed(note) der Klasse BuddyStone

```
private void NotePlayed(NoteEvent.Note note)
{
    if (isPlayerNear)
    {
        int octave = 0;
        int harmony = 0;
        if (buddyMode.Equals(BuddyMode.PerfectFifth))
        {
            harmony = (int)note + 5;
            if (harmony > (int)NoteEvent.Note.C2)
            {
                harmony = (int)note - 4;
               octave = 1;
            }
        }
        else if (buddyMode.Equals(BuddyMode.LowOctave))
        {
            harmony = (int)note;
            octave = -1;

        }
        else if (buddyMode.Equals(BuddyMode.DiatonicThird))
        {
            harmony = (int)note + 2;
            if (harmony > (int)NoteEvent.Note.C2)
            {
                harmony = (int)note - 5;
                octave = 1;
            }
        }
        NoteEvent.Note harmonyNote = (NoteEvent.Note)harmony;
        StartCoroutine(PlayDelayedNote(harmonyNote, octave));
    }
}
```

Die Variable `harmony` ist der Platzhalter für den zu spielenden Ton des Buddy-Steins und wird anfangs mit 0 initialisiert. Anschließend wird der aktuelle Wert der Variable `buddyMode` abgefragt. Um eine Quinte zu erhalten, wird zur übergebenen Note 5 addiert. Für den Fall, dass die Note dann über dem C2 (= 8) landet, wird von der Note wiederum 4 subtrahiert und dafür die Variable `octave` auf 1 gesetzt. Soll die tiefere Oktave gebildet werden, wird die Note übernommen und `octave` auf -1 gesetzt. Für die diatonische Terz werden zunächst zum bisherigen Ton 2 addiert. Liegt der Ton über C2, werden diesmal 5 abgezogen und wiederum die `octave` auf 1 gesetzt, da der zu spielende Ton im nächsten Register liegt.

Nun wird aus dem `int`-Wert in `harmony` das bereits bekannte `NoteEvent.Note` erstellt. Beim Testen dieser Funktionalität ist jedoch aufgefallen, dass der Ton vom Buddy-Stein oftmals kaum zu hören war, da er zeitgleich zum gespielten Ton der Spielfigur erklang. Um dies zu umgehen und den Ton des Steins wahrnehmbarer zu machen, wird der Ton einige Millisekunden später abgespielt. In Unity wird das durch eine so genannte `Coroutine` realisiert: dies ist eine ausgelagerte Methode mit dem Schlüsselwort `IEnumerator` vor dem Methodennamen. Der Code in Tabelle 11 zeigt die Methode `PlayDelayedNote(note, octave)`.

Tabelle 11: Programmcode: Ausgelagerte Methode PlayDelayedNote(note, octave)

```
public IEnumerator PlayDelayedNote(NoteEvent.Note note, int octave)
{
    yield return new WaitForSeconds(0.05f);
    notePlayer.PlayNote(note, octave, NoteEvent.Beat.Quarter, 100,
NotePlayer.MidiInstrument.Marimba);
}
```

Zuerst wird hier eine gewisse Zeit gewartet: ermöglicht wird dies durch den Befehl `yield return new WaitForSeconds(0.05f);` - `0.05f` entsprechen hierbei 50 Millisekunden. Dies lässt den aktuell laufenden Thread für die genannte Zeit *warten*, bevor dann der darauffolgende Code ausgeführt wird. Nun soll der `notePlayer` die übergebene Note abspielen. Neu ist hierbei, dass ein anderes Instrument gewählt wurde, um einen Kontrast zur Panflöte zu schaffen.

4.5.3 Zu MIDI-Dateien spielen

Eine weitere Idee für das Spiel, die im Zusammenhang mit den Buddy-Steinen aufkam, beinhaltet Musik, die von den Steinen ertönt.

Abbildung 35: Midi File Player zum Abspielen von MIDI-Dateien

Um dies exemplarisch unter Berücksichtigung des Aspekts des Freiform-Spiels umzusetzen, wurde die Funktionalität des MPTKs benutzt, MIDI-Dateien abzuspielen. Dafür lässt sich das entsprechende Prefab namens `MidiFilePlayer` in die Unity Scene integrieren. In Abbildung 35 ist das dazugehörige Script-Component mit den verschiedenen Einstellmöglichkeiten zu sehen. Bei Klick auf `Select Midi` öffnet sich eine kleine Liste mit verschiedenen MIDI-Dateien, die allesamt für MIDI transkribierte Songs von bekannten Künstler*innen sind; diese Dateien waren schon im MTPK Asset enthalten. Zudem lassen sich diverse Eigenschaften verändern, wie zum Beispiel die Lautstärke, die Tonart und weitere Parameter. Zum Testen wurde das *Präludium in C-Dur (Prelude in C Major)* von Johann Sebastian Bach gewählt, da dieses Stück, wie im Titel schon steht, in C-Dur ist. Durch den Parameter `Play At Startup` kann sichergestellt werden, dass das Stück abgespielt wird, sobald die Scene gestartet wird. Zum laufenden Stück können nun alle Töne dazu gespielt werden – und steht man in der Nähe des Buddy-Steins, harmoniert dieser mit den gespielten Tönen, je nach eingestelltem Wert der Variable `buddyMode`.

4.5.4 Bewertung

Insgesamt betrachtet funktioniert die Steuerung `OnlyButtons` besser im Vergleich mit den Steuerungen, die Analogsticks für das Auswählen von Tönen benutzt. Es ist durch das direkte Abspielen von Tönen näher an einem Musikinstrument und dadurch intuitiver. In einem kurzen Test mit ca. fünf anwesenden Studenten wurde diese Steuerung im Vergleich zu den anderen vorgezogen. Die Tester*innen hatten Spaß daran, frei und ohne bestimmte Vorgaben Töne spielen und sich ausprobieren zu können. Die Interaktion mit dem Buddy-Stein bewerten alle Tester*innen positiv – auch die Möglichkeit, zu einem laufenden Musikstück mitzuspielen, hatte den Testpersonen viel Freude bereitet.

In dieser kurzen Testphase hat sich herausgestellt, dass eine gewisse Latenz zwischen Drücken des Buttons und Abspielen des Tons herrschte. Dies führt leider dazu, dass die Töne nicht perfekt auf den Schlag im Takt gespielt werden können. Es war nicht so viel, dass das Spielen unmöglich gemacht wird, aber eine rhythmisch anspruchsvolle Darbietung wäre aktuell nicht einfach zu gewährleisten.

5 Fazit und Ausblick

In der vorliegenden Arbeit wurde untersucht, inwiefern ein Gamepad dazu dienen kann, als Musikinstrument eingesetzt werden zu können und eingesetzt wird. Dazu wurden zunächst die Grundlagen von Musik, Musikinstrumenten und Gamepads dargelegt, um ein Fundament für die Analyse von Spielen und deren musikalischen Elemente zu schaffen. Spiele wie Wandersong, Guitar Hero und Rocksmith hatten alle Musik als zentrales Gameplay-Element, doch jedes Spiel auf verschiedene Art und Weise.

Wandersong bietet die Möglichkeit, Töne direkt über das Gamepad zu spielen, jedoch werden die Töne nur abstrahiert ohne Notennamen dargestellt. Somit ist nur für Musiker, die die Töne kennen, eine Transferleistung in die reale Welt möglich. Guitar Hero ist technisch gesehen kein reines Musikspiel, da durch die Buttons keine richtigen Töne gespielt werden können. Rocksmith hingegen ist ein Spiel, das Musik in ihrer Reinform bietet, da mit Gitarre und Bass-Gitarre richtige Musikinstrumente als Gamepad dienen. Dies erwies sich jedoch nicht als besonders einsteigerfreundlich, da nicht jeder Zugriff auf solche Instrumente hat.

Ausgehend von der Analyse dieser Spiele wurde im Rahmen eines Prototyps neue Steuerungskonzepte konzipiert und entwickelt. Zunächst wurde der Prototyp von Sonority, ein musikalisches Rätselspiel, auf dessen bisher implementierte Steuerung untersucht und eine schon durchgeführte Evaluation in die Betrachtung miteinbezogen. Aufgrund der Ergebnisse der Evaluation wurden alternative Konzepte zur Eingabe von Tönen implementiert und in kurzen on-the-fly Tests mit anwesenden Personen durchgeführt. Hier bedarfs es einer größer angelegten Nutzerstudie, um die Ergebnisse dieser Arbeit besser bewerten zu können.

Schlussendlich wurde die Tastenbelegung, bei der jeder Ton einer Taste zugewiesen wird, als am intuitivsten befunden. Auch diese unmittelbare Wahrnehmung gilt es, in größeren Tests zu evaluieren. Für diese Tests müsste das prototypisch gebaute Level um einige Elemente erweitert werden und idealerweise in die Welt und Handlung von Sonority integriert werden.

Das in Kapitel 4.5 gezeigte Testszenario eines Freiform-Spiels fiel besonders positiv aus – die Nähe zum richtigen Musizieren zeigte sich hier am besten. Denkbar wären hier längere musikalische Performances, die nur mit einem herkömmlichen Gamepad gespielt werden - vorausgesetzt, das Problem der Latenz lässt sich eindämmen.

Die Idee der Freiform-Level soll in den Prototyp integriert werden. Es soll die Möglichkeit gegeben werden, an bestimmten Punkten im Spiel auf derartige Umgebungen zu

stoßen, in denen keine Rätsel im bisher definierten Sinn gelöst werden müssen, sondern frei und kreativ mit Buddy-Steinen interagiert und gegebenenfalls zu eigens komponierten Musikstücken Songs mitgespielt werden kann.

Auf Basis dieser Arbeit können die neu gewonnen Erkenntnisse über die Eingabe von Tönen mit Gamepads in der zukünftigen Entwicklung des Spiels eingesetzt werden.

Anhang

Der beiliegende Datenträger enthält folgende Dateien:

- Vorliegende Arbeit in digitaler Version mit dazugehörigen Quellmaterialien und Bilder
- Das Unity Projekt mit Programmcode
- Eine auf Windows-PCs ausführbare Version des Prototyps von *Sonority* mit der bearbeiteten SampleScene

Quellenverzeichnis

Ackermann, Philipp (1991): *Computer und Musik.* Springer Verlag, Wien

Alphaton / Wikimedia Commons (2010): *Diagram of an Xbox 360 controller with button labels.* Abgerufen am 15.12.19, von https://commons.wikimedia.org/wiki/File:360_controller.svg [Grafik]

Bachmann, Thierry (2019): *Midi player Tool Kit for Unity.* Abgerufen am 12.12.19, von https://paxstellar.fr/

Bellosa, Gerhard (2000): *Die Sprache der Musik.* BoD – Books on Demand, Norderstedt

Bernstein, Herbert (2019): *Elektroakustik: Mikrofone, Klangstufen, Verstärker, Filterschaltungen und Lautsprecher.* Springer-Verlag, Wiesbaden

Brockhaus, Immanuel (2017): *Kultsounds: Die prägendsten Klänge der Popmusik 1960-2014.* transcript Verlag, Bielefeld

Brown, Mark [Game Makers's Toolkit] (2016, Mai 10): *Controllers Control Everything* [Youtube]. Abgerufen von https://www.youtube.com/watch?v=VJGKDyrR8qc

Deutscher Computerspielepreis (2019): *Strahlende Gewinner und gute Nachrichten beim Deutschen Computerspielepreis 2019 (Pressemitteilung).* Abgerufen am 20.12.19, von https://deutscher-computerspielpreis.de/presse/strahlende-gewinner-und-gute-nachrichten-beim-deutschen-computerspielpreis-2019

Dickreiter, Michael (1977): *Der Klang der Musikinstrumente.* TR-Verlagsunion GmbH, München

Görne, Thomas (2014): *Tontechnik: Hören, Schallwandler, Impulsantwort und Faltung, digitale Signale, Mehrkanaltechnik, tontechnische Praxis.* Carl Hanser Verlag GmbH Co KG, München

Grabner, Hermann (2015): *Allgemeine Musiklehre.* 26. Auflage, Bärenreiter-Verlag Karl Vötterle GmbH & Co. KG., Kassel

Hardy, James (2014): *The Beats to Beat: A History of Guitar Hero.* Abgerufen am 19.11.19, von https://historycooperative.org/the-beats-to-beat-a-history-of-guitar-hero/

Heatherly, Ben & Howard, Logan (2014): *Video Game Controllers.* Abgerufen am 03.11.19, von http://andrewd.ces.clemson.edu/courses/cpsc414/spring14/papers/group2.pdf

Helmholtz, Hermann von (1913): *Die Lehre von den Tonempfindungen als Physiologische Grundlage für die Theorie der Musik.* 6. Auflage, Vieweg+Teubner Verlag, Wiesbaden

HowStuffWorks (o. D.): *How Guitar Hero Works.* Abgerufen am 19.11.19, von https://cdn.hswstatic.com/gif/guitar-hero-5.jpg [Grafik]

Hunicke, Robin / Marc LeBlanc / Robert Zubek (2004): *MDA: A Formal Approach to Game Design and Game Research.* Abgerufen am 05.11.19, von https://users.cs.northwestern.edu/~hunicke/MDA.pdf

Huppertz, Michael (2003): *Musik und Gefühl.* In: Holtmeier, Ludwig; Klein, Richard & Mahnkopf, Claus-Steffen: *Musik & Ästhetik*, Klett-Cotta Verlag, Stuttgart

Kayali, Fares & Pichlmair, Martin (2007): *Levels of Sound: On the Principles of Interactivity in Music Video Games.* Erschienen in: *DiGRA 07 – Proceedings of the 2007 DiGRA International Conference: Situated Play.* Abgerufen am 11.11.19, von https://publik.tuwien.ac.at/files/pub-inf_4783.pdf

Krash / Wikimedia Commons (2005): *Minimoog synthesizer.* Abgerufen am 01.11.2019, von https://de.wikipedia.org/wiki/Moog_Minimoog#/media/Datei:Minimoog.JPG (Grafik)

Lobanov, Greg (2018): *Wandersong Factsheet.* Abgerufen am 01.12.19, von http://greg.style/press/sheet.php?p=wandersong

Lu, William (2003): *Evolution of Video Game Controllers: How Simple Switches Lead to the Development of the Joystick and the Directional Pad.* Abgerufen am 02.11.2019, von https://web.stanford.edu/group/htgg/cgi-bin/drupal/sites/default/files2/wlu_2003_1.pdf

Margel, Michael (o. D.): *Literature Review of Video Game Input Devices.* Abgerufen am 01.11.19, von http://margel.ca/files/papers/428.pdf

Moser, Hans Joachim (2013): *Allgemeine Musiklehre*. 3. Auflage, Walter de Gruyter GmbH, Berlin

Pierini, David / Appleman (2016): *It's all in the fingers for this ‚sick' iPad drummer.* Abgerufen am 05.11.19, von https://www.cultofmac.com/417578/its-all-in-the-fingers-for-this-sick-ipad-drummer/

Petkovic, John (2008): *Guitar Hero co-founders turned a bright idea into $100 million.* Abgerufen am 30.11.19, von https://www.cleveland.com/pdextra/2008/03/guitar_hero_cofounders_turned.html

Qpaly / Wikimedia Commons (2004): *Schematische Darstellung der Klaviatur von Tasteninstrumenten.* Abgerufen am 20.10.2019, von https://de.wiktionary.org/wiki/Datei:Klaviatur_(Tasten).png [Grafik]

Redaktion Duden Learnattack GmbH (2019): *Systematische Klassifizierung der Musikinstrumente.* Abgerufen am 29.10.2019, von https://www.lernhelfer.de/schuelerlexikon/musik/artikel/systematische-klassifizierung-der-musikinstrumente

Reinaldo Mendoza, Madeline / Elisa Schorrig / Willi Schorrig (2019): *Spiele-Konzept für Sonority, Einreichung zum Deutschen Computerspielpreis (DCP) 2019*

Schorrig, Willi (2019): *Musik als Spielmechanik. Eine Analyse der Regeln und Wirkung von Musik aus Game-Design-Perspektive und die prototypische Implementierung eines Musik-Rätselspiels mit anschließender Nutzerstudie.* Unveröffentlichte Masterarbeit (Stand: 08.12.19), Hochschule der Medien, Stuttgart

Spitzer, Manfred (2002): *Musik im Kopf*. 1. Auflage, Schattauer Verlag, Stuttgart

Stange-Elbe, Joachim (2015): *Computer und Musik: Grundlagen, Technologien und Produktionsumgebungen der digitalen Musik.* Walter de Gruyter GmbH & Co. KG, Berlin

Statistisches Bundesamt (2019): *Prognose der Umsätze im Markt für Videospiele in Deutschland von 2010 bis 2023 nach Segment.* Zitiert nach de.statista.com. Abgerufen am 20.12.19, von https://de.statista.com/statistik/daten/studie/3886/umfrage/umsaetze-im-markt-fuer-games-nach-segment/

Taibzadeh, Golnar & Moezzi, Fariba (2014): *Ney*. Abgerufen am 02.11.19, von https://theforgotten2014.wordpress.com/about/6-the-story-of-the-moaning-ney/ney/

Ubisoft (2017): *Offizielle Website von Rocksmith.* Abgerufen am 03.12.19, von https://www.ubisoft.com/de-de/game/rocksmith/#223165769

Ubisoft (2017): *Billy Talent „Devil in a Midnight Mass".* Abgerufen am 03.12.19, von https://ubistatic19-a.akamaihd.net/ubicomstatic/de-DE/global/media/RS2014_screenshots_BillyTalent_original_165618.jpg [Grafik]

Unity3D (o. D.): *Unity User Manual (2019.2).* Online-Dokumentation der Unity-Engine. Abgerufen am 06.12.19, von https://docs.unity3d.com/Manual/UnityOverview.html

WikiHero (o. D.): *Guitar Hero.* Abgerufen am 30.11.19, von https://guitarhero.fandom.com/wiki/Guitar_Hero

WikiHero / Gta-mysteries (o. D.): *Star Power gems as they appear in GH2. The star power meter is seen above the rock meter.* Abgerufen am 30.11.19, von https://vignette.wikia.nocookie.net/guitarhero/images/7/72/StarPower-GH2-gems.jpg/revision/latest?cb=20101018050627 [Grafik]

Wolf, Rebecca (2018): *Musikinstrumente.* In: Morat D., Ziemer H. (eds) *Handbuch Sound.* J.B. Metzler, Stuttgart

Y2kcrazyjoker4 / Wikimedia Commons (o. D.): *Photo of the guitar controller for the Playstation 2 game „Guitar Hero“.* Abgerufen am 19.11.19, von https://commons.wikimedia.org/wiki/File:Guitarhero-controller.jpg [Grafik]