Spielerisches Lernen
"Entwicklung eines Programmes zum spielerischen Erlernen der Mundharmonika"

Abschlussarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

an der

Hochschule für Technik und Wirtschaft Berlin
Fachbereich Wirtschaftswissenschaften II
Studiengang Angewandte Informatik

1. Prüfer: xxxxxxxxxxxxxxxxxxxxxxx
2. Prüfer: xxxxxxxxxxxxxxxxxxxxxxxxx

Eingereicht von Tom Bleek
13. März 2014

Inhaltsverzeichnis

Abkürzungsverzeichnis....4
Abbildungsverzeichnis....5
Tabellenverzeichnis....7
1. Von der Kunst zum Spiel....8
2. Grundlagen....10
2.1 Was man wissen sollte....10
2.2 Musik auf dem Papier....10
2.2.1 Musik besteht aus Tönen....10
2.2.2 Bitte nach Vorschrift spielen....13
2.3 Die Mundharmonika....18
2.3.1 Eine kurze Geschichte der Mundharmonika....18
2.3.2 Aufbau der Mundharmonika....19
2.3.3 Die Stimmung macht die Musik....20
2.3.4 Mundraummusik....22
2.4 Vertreter der Musikspiele....23
2.4.1 Von Karaoke und Plastikgitarren....23
2.4.2 Singstar und Ultrastar....23
2.4.3 Guitar Hero und Rockband....25
2.4.4 Rocksmith....26
2.4.5 Synthesia....27
2.5 Wie man eine Frequenz erhält....28
2.5.1 Ein Signal ist die Summe seiner Teile....28
2.5.1.1 Die Sinusfunktion....28
2.5.1.2 Mehr als nur ein einzelner Ton....30
2.5.2 Die Spektralanalyse....31
2.5.2.1 Die Fourierreihe....31
2.5.2.2 Die Fouriertransformation....35
2.5.2.3 Die Diskrete Fourier-Transformation (DFT)....36
2.5.2.4 Die Schnelle Fourier-Transformation (FFT)....41
2.6 Wie lernt es sich am besten?....43
2.6.1 Was Lernen für das Gehirn ist....43
2.6.2 Wie oft gelernt werden sollte....44
2.6.3 Voraussetzungen für gutes Lernen....45
3. Die Software....48
3.1 Ziel im Rahmen der Arbeit....48
3.2 Struktur und Aufbau der Implementierung....49
3.2.1 Auswahl des Frameworks....49
3.2.2 Vorgehensweise der Tonerkennung....53
3.2.2.1 Trennung von Spiel und Tonanalyse....53
3.2.2.2 Rauschunterdrückung....54
3.2.2.3 Einlesen mit Hilfe eines Ringspeichers....55
3.2.2.4 Anwendung der FFT....57
3.2.2.5 Konvertierung in angespielten Ton....58
3.2.2.6 Knapp daneben ist nicht so schlimm....60

3.2.3 Das Dateiformat der Musikstücke....62
3.2.3.1 Schnell, simpel und unbekannt....62
3.2.3.2 MIDI (Musical Instrument Digital Interface)....65
3.3 Aufbau für den Nutzer....67
3.3.1 Usability....67
3.3.2 Vom Start zum Spiel....69
3.3.3 Helfende Funktionen für den Nutzer....72
3.3.3.1 Achievements....72
3.3.3.2 Komfort in der Titelwahl....74
3.3.3.3 Unterstützung beim Bending....75
3.3.3.4 Immer zugängliche Optionen....76
3.3.4 Wie man neue Lieder bekommt....77
3.3.4.1 Die rechtliche Lage....77
3.3.4.2 Bezugsquellen neuer Lieder....78
3.4 Der Nutzer als Testroutine....79
3.5.1 Maschinelles Testen....79
3.5.2 Usability Expert Evaluation....79
3.5.3 Usability Tests mit Google Analytics....80
3.5.4 Usability mittels Feedback von Nutzern....81
4. Spielen nicht lesen!....83
4.1 Mehr als die Summe ihrer Teile....83
4.2 Die Früchte dieser Arbeit....84
4.3 Danksagungen....85
5. Anhang....86
5.1 UML Diagramme....86
5.2 Code Dokumentation (CD-ROM)....89
6. Quellenverzeichnis....90
6.1 Offlinequellen....90
6.2 Onlinequellen....90
6.2.1 Erläuterungen....90
6.2.2 Informatik Allgemein....90
6.2.3 Musiklehre und Akustik....90
6.2.4 Mundharmonika....92
6.2.5 Frequenzanalyse....93
6.2.6 Benutzerfreundlichkeit....94
6.2.7 Lernen....95
6.2.8 Videospiele....95

Abkürzungsverzeichnis

API Application Programming Interface, die nach außen sichtbaren Funktionen und Befehle einer Programmierbibliothek

DFT Diskrete Fourier-Transformation

FFT Fast Fourier-Transformation

Hz Hertz

kHz Kilohertz (1 kHZ entspricht 1.000 Hz)

MIDI Musical Instrument Digital Interface

π Pi = 3,14159... Die Kreiszahl, gibt das Verhältnis von Umfang zu Radius an.

Abbildungsverzeichnis

1. Hörempfinden des Menschen....12
2. Ausschnitt der Klaviatur....15
3. Aufbau der Mundharmonika....19
4. Richterstimmung in C....22
5. Erreichbare Noten der Richterstimmung in C....22
6. Bidlschrimfoto – Singstar....24
7. Bildschirmfoto – Guitar Hero....25
8. Bildschirmfoto – Rocksmith....26
9. Bildschirmfoto – Synthesia....27
10. Sinusveranschaulichung....28
11. Der Kammerton A4 (440 Hz)....29
12. Sprachaufnahme....30
13. Rechtecksignal....31
14. Rechtecksignal, Aproximationen....32
15. Amplituden des Rechecksingals....34
16. Nichtperiodisches Rechtecksignal....35
17 Frequenzspektrum des Rechtecksignals....36
18. Sinus mit einfacher und doppelter Frequenz....37
19. Abtastung eines Signals....38
20. Abgetastetes Mikrophonsignal....39
21. Frequenzspektrum mittels DFT....40
22. Vergleich der DFT und FFT....41
23. FFT, einfaches Schema....42
24. Basiskomponenten eines Computerspiels....49
25. Ringspeicher....56
26. Debug-Ansicht des Prototypen....58
27. Ablauf des Erststartes....69
28. Spielansicht des Prototypen....70
29. Normaler Ablauf des Programmes....71
30. Bildschrimfoto – Halo: Reach....73
31. Konzept des Titelauswahlbildschirmes....74

32. Konzept des Techniken-üben-Bildschrimes.....75
33. Icon der app Harpo.....84
34. Klassendiagramm View-Hierarchie.....86
35. Klassendiagramm Audioanalyse.....87
36. Paketgiagramm des Prototypen.....88

Tabellenverzeichnis

1. Vergleich zwischen verschiedenen Spiele Frameworks.....51

1. Von der Kunst zum Spiel

Seit frühester Zeit widmet sich der Mensch den musikalischen Künsten. So gehen die ältesten bekannten Funde eines Musikinstrumentes auf etwa 45.000 Jahren vor unserer modernen Zeitrechnung zurück.[1] Und bis heute ist die Musik ein steter Begleiter wohl eines jeden Menschen.

Wie bei vielen Medien ist das Verhältnis von Produzenten und Konsumenten stark einseitig. So wird Musik von einigen wenigen für sehr viele andere produziert. Mit etwas Glück darf, oder muss, man im Kindesalter ein Instrument erlernen. Meist zum Leidwesen des Kindes, welches die oft erwähnte Blockflöte eher zu hassen als zu lieben lernt.
So kommt es das ein oder andere Mal vor, dass das Kind der Schulzeit entwachsen ist und langsam Interesse an der selbsterschaffenen Musik entwickelt.

Ab und an fällt die Wahl des neu zu erlenenden Musikinstrumentes auf die Mundharmonika. Denn kaum ein anderes Instrument ist so günstig. Sie kann schon oft für wenige Euro erstanden werden.

Ziel dieser Arbeit ist es, diesen Menschen auf ihrem, nicht immer einfachen, Weg des Lernens, unter die Arme zu greifen.

Dafür soll eine Software entstehen, die es möglich macht, vordefinierte aber auch selbstgeschriebene Lieder zu üben, sie zu wiederholen und nach und nach losgelöst von der Software spielen zu können.
Die Software soll nicht dazu dienen, das Spiel der Mundharmonika bis zur Perfektion beherrschen zu lernen, sondern lediglich Musikstücke üben zu können. So hat gerade dieses Blasinstrument einige Ecken und Kanten in

1 Eiszeitkunst im suddeutsch-schweizerischen Jura. Anfänge der Kunst – Claus-Stephan Holdermann, Hansjürgen Müller-Beck, Ulrich Simon

der Spielweise, die der junge Spieler zwar leicht versteht, deren Umsetzung aber nur durch langes Üben und/oder durch persönliche Betreuung eines erfahrenen Spielers erlernbar ist.

Um ein möglichst zufriedenstellendes Produkt entwickeln zu können, ist es unabdingbar, sich mit den Grundlagen der westlichen Musiktheorie vertraut zu machen. Deshalb beschäftigt sich die Arbeit zu Beginn unter anderem mit einigen Fragestellungen, wie zum Beispiel, was sind überhaupt Töne und wie stehen diese zu einander? Weiterhin ist es wichtig zu wissen, wie man ein analoges Signal in ein digitales umwandelt und für die Erkennung der gespielten Töne verwenden kann.

Im Rahmen dieser Arbeit soll nicht nur eine Software sondern ein Spiel entstehen, welches dem Mundharmonikaanfänger das Erlernen neuer Lieder einfacher macht. Daher ist es auch wichtig zu wissen, ob es denn bereits andere Vertreter dieser Art von Software gibt und wie diese aufgebaut sind. Dadurch wird es möglich, den spielenden und lernenden Nutzer auf dem Niveau abzuholen auf dem er bereits einmal war und ihm einen Weg zu zeigen der ihn buchstäblich auf das nächste Level führt.

Ich wünsche Ihnen als Leser und vielleicht auch als potentieller Nutzer des Programms viel Vergnügen beim Lesen dieser Arbeit. Denn wer versteht, was er tut, tut es umso lieber.

Tom Bleek

2. Grundlagen

2.1 Was man wissen sollte

Wie jedes Instrument ist die Mundharmonika einzigartig. Einzigartig in ihrer Bauweise, ihrem Klang und wie sie sich musikalisch einsetzen lässt.

Für das Verständnis des Hauptteiles der Arbeit (Kapitel 3) ist es wichtig, vorher einige Informationen bereitzustellen. So, dass der Leser einen leichteren Einstieg in das Thema erhält.

2.2 Musik auf dem Papier

2.2.1 Musik besteht aus Tönen

Musik ist allgemein bekannt. Und doch ist Musik für jeden etwas anderes. Was der Enkel heute hört, ist für seine Großmutter oftmals nur Krach. Was die Musik vom Krach unterscheidet, ist zunächst einmal die Empfindung des einzelnen Menschen. Denn Musik ist immer subjektiv. Objektiv hingegen unterscheidet Musik sich vom Krach in ihrer Struktur. Musik besteht aus Tönen, die in der Regel gut definiert und arrangiert sind.

Aber was ist ein Ton genau und wie wird er erzeugt? Physikalisch ist ein Ton eine Schwingung bzw. eine Ansammlung von Schwingungen. Ein Pendel einer großen Standuhr schwingt auch. Doch ist das bereits ein Ton? Nein. Denn es schwingt viel zu langsam. Entscheidend für die Hörbarkeit des Tones sind die Frequenz und die Schwingungsamplitude. Also wie oft bzw. wie schnell und mit welcher Kraft ein Objekt schwingt.

Die Frequenz wird dabei in der Einheit Hertz (Hz) gemessen. Damit werden die Anzahl der Schwingungen pro Sekunde angegeben. Schwing das Pendel innerhalb von zwei Sekunden hin und wieder zurück so hat es eine Frequenz von 0,5 Hz.

Das schwingende Objekt braucht allerdings noch einen Informationsträger. Schwingt die Seite eines Klaviers, so lässt sie auch die Luft um sie herum schwingen. Das erzeugt eine Welle, oder auch Kette, aus schwingender Luft. Das ist eine Schallwelle. Ganz genau so, als ob man ein Stein in einen Teich wirft, breitet sich diese Welle aus. Ist nun ein Ohr in direkter Reichweite einer Schallwelle, so kann man den Ton auch hören. Denn im Grunde macht nicht der Ton die Musik, sondern das Gehirn, welches aus der schwingenden Luft, mittels dem Ohr und seines komplexen Aufbaues, Töne erkennt.

Das Pendel der Uhr erzeugt also einen Ton, den man aber nicht hören kann. Der Grund hierfür ist, dass das menschliche Ohr nur Frequenzen in einem bestimmten Bereich wahrnehmen kann. Und auch innerhalb dieses Bereiches sind nicht alle Frequenzen gleich gut hörbar. Die folgende Grafik verdeutlicht dies:

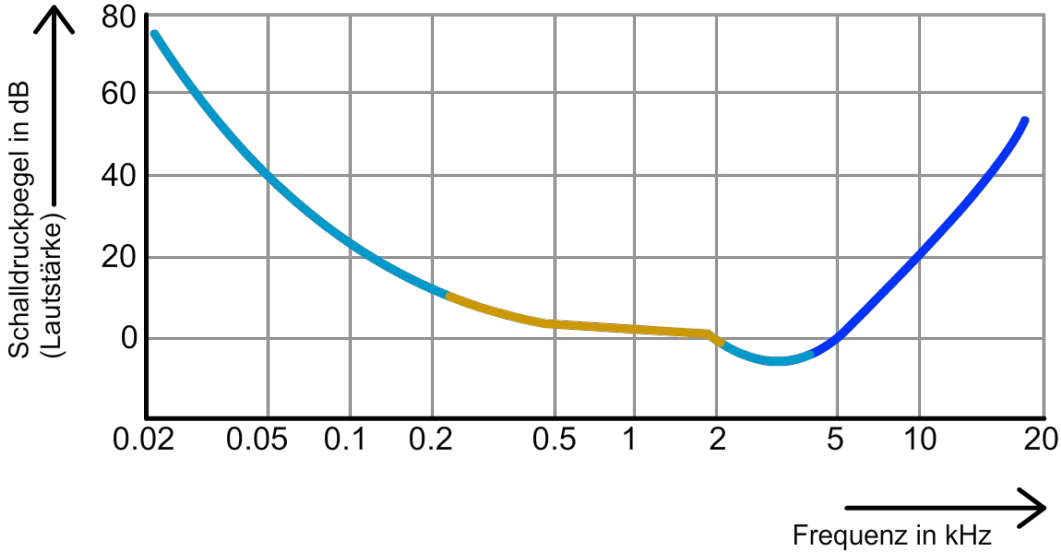

Abbildung 1: Hörempfinden des Menschen mit eingezeichneten Klavier und Mundharmonikafrequenzen. Angelehnt an: siehe Fußnote[2]

Je niedriger die Kurve, desto empfindlicher ist unser Gehör für eine bestimmte Frequenz. So ist hier dargestellt, dass das menschliche Gehör im Bereich von 2 bis 5 Kilohertz (kHz) am empfindlichsten reagiert. Wenn

man einem Klavierstück zuhört, hört man Töne in einem Bereich von ~ 16 Hz bis ~ 4.186 Hz. Dieser ist in der Grafik dargestellt durch den hellblauen Bereich der Kurve. Innerhalb dieses Bereiches ist der Frequenzbereich für eine Richter-Mundharmonika in der Tonart C eingezeichnet.[2,]

Jedes Instrument klingt anders. So klingt die Note C auf einer Geige anders als auf einem Klavier oder der Mundharmonika, obwohl der gleiche Ton mit der gleichen Frequenz gespielt wird. Was hierbei zum Tragen kommt, sind die Obertöne. Spielt man ein A4, diese Note ist festgelegt mit 440 Hz, so schwingt bei einigen Instrumenten auch das nächste A, also A5, mit. Die Note A5 ist mit einer Frequenz von 880Hz festgelegt. Es schwingen also auch Vielfache der eigentlichen Grundschwingung mit. Und diese Kombination von Grund- und Obertönen machen die Klangfarbe eines Instrumentes aus. Wobei ein Oberton immer nur ein Vielfaches des Grundtones ist, nicht aber ein Teiler. Wird A mit 440Hz gespielt, schwingt 880Hz mit, andersherum ist das aber nicht der Fall.

2.2.2 Bitte nach Vorschrift spielen

Da sich die Musik auf dem gesamten Globus zeitgleich entwickelte, entwickelten sich auch diverse Vorschriften wie bestimmte Instrumente gestimmt werden sollen. So klingt orientalische Musik anders als westliche.

Musik aus anderen Teilen der Welt kommt dem Europäer oft leiernd oder quengelig vor. Das liegt daran, dass Musik etwas ist, was wir lernen zu hören. So nutzen wir für alle Instrumente der westlichen Welt die gleichen Frequenzen und das Gehirn hat eben diese Töne verinnerlicht.

Bis 1917 gab es auch in Europa diverse Arten, um ein Instrument zu stimmen. So klang das Klavier, auf dem Bach spielte, damals anders als

2 http://www.dasp.uni-wuppertal.de/ars_auditus/psychoak/psychoak4.htm (Grafik modifiziert)

heute. Außer man stimmt sein Klavier nach seinen Vorgaben.[3] Heute wird fast ausschließlich die sogenannte "Gleichstufige Stimmung" verwendet. Das bedeutet, dass die Töne alle das gleiche Verhältnis zu einander haben. Somit wurden Instrumente vielseitiger, aber auch ungenauer. Das fällt aber nur sehr wenigen Kennern auf, denn viele Musiker kennen oftmals gar keine anderen Stimmungen. Dabei sind weit mehr als 50 verschiedene Stimmungen dokumentiert worden.[4]

So nimmt man heute zum Stimmen eines Instrumentes einen Grundton, der auch Kammerton genannt wird, und richtet die restlichen Töne danach aus. Der Kammerton ist oft ein Ton mit einer Frequenz von 440 Hz (U.S.) oder 441 Hz (Europa, hier teilweise auch höher).[5] Für die meisten Menschen ist der Unterschied aber fast nicht wahrnehmbar.

Für den weiteren Verlauf der Arbeit wird ein Kammerton von 440 Hz angenommen. Anhand der Klaviatur wird gezeigt, wie man für eine bestimmte Taste die einzustellende Frequenz berechnet. Weiterhin wird gezeigt, wie man aus einer erkannten Frequenz die dazugehörige Tastennummer erhält.

Berechnung der Frequenz für eine bestimmte Taste:[6]

$$f(n) = 440\left(\sqrt[12]{2}\right)^{n-49}$$

"n-49" kommt zustande, da der Kammerton in diesem Beispiel auf der Taste 49 Liegt. Im nächsten Abschnitt wird die Frequenz für eben diesen Kammerton berechnet. Zusätzlich wird noch der namentlich wohl bekannte Ton "Das mittlere C" berechnet:

3 http://kilchb.de/rein_duffin.html
4 http://www.groenewald-berlin.de/Inhaltsverzeichnis.html
5 http://www.tedrounds.com/Can_you_hear_the_difference.htm
6 http://www.sengpielaudio.com/Rechner-notennamen.htm

$$f(n)=440(\sqrt[12]{2})^{49-49}$$

$$f(n)=440(\sqrt[12]{2})^{0}$$

$$f(n)=440*1$$

$$f(n)=440\text{Hz}$$

Wie zu erwarten, berechnet man hier den Ton, von dem alle anderen Tone ausgehen. Sieht man sich die Berechnung für das "Mittlere C" an, entsteht eine auf den ersten Blick nicht sehr harmonische Schwingungsanweisung:

$$f(n)=440(\sqrt[12]{2})^{40-49}$$

$$f(n)=440(\sqrt[12]{2})^{-9}$$

$$f(n)=440(1{,}05946309436)^{-9}$$

$$f(n)=440*0.5946035575$$

$$f(n)=261.625565301\text{Hz}$$

Wieso ist das so? Das Geheimnis liegt in der Oktave und der damit verbundenen zwölften Wurzel aus 2. Die Tasten des Klaviers haben alle einen Namen:

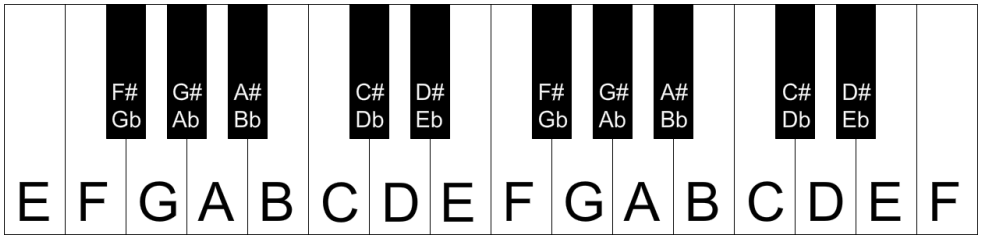

Abbildung 2: Ausschnitt der Klaviatur mit Notennamen modifiziert[7]

Die weißen Tasten werden durchgehend von A bis G bezeichnet. Wobei in Deutschland das B einem H entspricht. Die schwarzen Tasten sind die Halbtöne. Da diese zwischen den weißen liegen, haben diese jeweils zwei Namen. Das F# (Fis) bedeutet, dass das F um einen Halbton erhöht ist. Die Taste heißt aber auch Gb (Ges), weil das G um einen Halbton vermindert ist. Das heißt aber nicht, dass ein F# in der Berechnung mit x + ½ berechnet wird! Die schwarzen tasten sind vollwertige Ganze in der Berechnung. Denn der relative Abstand von F zu F# ist der gleiche wie von B zu C.

So gibt es zwölf verschiedene Tasten. Danach wiederholt sich das Muster. Für den Kammerton A (genauer A4 oder A') wurde oben eine Schwingung von 440 Hz berechnet. Berechnet man nun die nötige Frequenz für das nächste A, so erhält man eine vorgeschriebene Schwingung von 880 Hz. Es ist also festgelegt, dass eine Oktave immer die doppelte Frequenz der vorangegangen hat.[8]

Mit diesen zwölf sich wiederholenden Tasten lassen sich diverse Vorschriften zum Spielen erzeugen. Die wohl bekanntesten Tonarten sind Dur und Moll. Diese sind Vorschriften, mit welchen Tasten gespielt wird und in welchen Abständen diese zu einander stehen.

Dur: Grundton + Ganzton + Ganzton + Halbton + Ganzton + Ganzton + Ganzton + Halbton

Der Grundton beschreibt dabei die Taste, mit der die dadurch entstandene Tonleiter beginnt.[9] Ist ein Stück in C-Dur geschrieben, nutzt man nur die weißen Tasten des Klaviers. Daher ist dies auch für viele Anfänger die

7 http://www.sengpielaudio.com/Rechner-notennamen.htm (2014-02-06)
8 http://www.sengpielaudio.com/Rechner-centfrequenz.htm (2014-01-22)
9 http://www.harmonielehre.info/tonarten-und-tonleiter.html (2014-01-22)

Tonart, mit der das Lernen begonnen wird. So ist auch bei der Mundharmonika die Tonart C-Dur am häufigsten vertreten.

Für das Projekt dieser Arbeit ist es aber nicht nur wichtig zu wissen, auf welche Frequenzen die Software reagieren muss, sondern auch der umgekehrte Fall ist wichtig. Der Nutzer soll einen Ton abgeben und dieser muss vom Programm interpretiert werden.

Dafür brauchen wir den Kehrwert der oben aufgeführten Formel:

$$Nr = 12 * log2(f/440) + 49$$

Als Beispiele betrachtet man hier auch wieder den Kammerton mit 440 Hz und das „Mittlere C“ mit rund 262 Hz:

$$Nr = 12 * log2(440/440) + 49$$

$$Nr = 12 * log2(1) + 49$$

$$Nr = 12 * 0 + 49$$

$$Nr = 49$$

Wie zu erwarten ist, sollte der Spieler die Taste 49 gedrückt haben. Beim „Mittleren C“ nutzen wir einen gerundeten Wert:

$$Nr = 12 * log2(262/440) + 49$$

$$Nr = 12 * log2(0.59545454545) + 49$$

$$Nr = 12 * (-0.74793671198) + 49$$

$$Nr = -8.97524054385 + 49$$

$$Nr = 40.0247$$

$$Nr \approx 40$$

Nachdem das Ergebnis gerundet wurde, erhält man auch hier die richtige Taste. Trotz dem ein Instrument nicht perfekt gestimmt sein kann, ist es so möglich, die richtige Taste und somit auch den richtigen Ton zu bestimmen. Und da eine Mundharmonika auch in einer nach diesen Formeln gestimmten Tonart gestimmt ist, lässt sich dieses Prinzip sehr gut darauf übertragen und somit herausfinden welche Kanzelle (das sind die Aushöhlungen) der Mundharmonika angespielt wurde.

2.3 Die Mundharmonika

2.3.1 Eine kurze Geschichte der Mundharmonika

Wie auch das Klavier veränderte sich die Mundharmonika im Laufe der Geschichte. Bereits rund 3.000 Jahre vor Christus wurde im alten China der erste Vertreter der Durchschlagzungen-Instrumente entwickelt. Rund 4.400 Jahre später wurde dieses als Sheng benannte Instrument auch in Europa bekannt. Von da an begannen die europäischen Instrumentenbauer mit dem Prinzip der Durchschlagzunge zu experimentieren. So wurde 1816 das Terpodion von Johann Buschmann erfunden. Dabei handelt es sich um ein klavierartiges Instrument, welches

sich wie eine heutige Mundharmonika anhört.[10] Bereits nur wenige Jahre später hielten die Menschen ein Instrument in der Hand, welches sich seitdem im Grunde nicht verändert hat. Denn 1825 baute Herr Richter von Böhmen die uns heute bekannte 10-Lochmundharmonika.

Seither hat sich, außer der Bauweise, nur die Stimmungsart der Zungen geändert. Die bekannteste geht dabei auf Herrn Richter von Böhmen selbst zurück und nennt sich dementsprechend Richter-Stimmung.

Seit die Mundharmonika 1829 in Massenproduktion gegangen ist, verbreitete sie sich über den gesamten Globus. So war sie wegen ihres geringen Preises und der handlichen Größe für viele Musiker zum Begleiter geworden.

Besonders für die schwarze Bevölkerung war sie ein erschwingliches Instrument. So wurde die Mundharmonika ein fester Bestandteil im Bereich Jazzmusik und Blues. Deshalb wird die 10-Lochmundharmonika umgangssprachlich auch Blues Harp genannt.

Heute ist die Mundharmonika selten als Soloinstrument anzutreffen. Vielmehr stellt sie einen Hintergrundbegleiter dar, der einem Musikstück eine typische Note hinzufügt.[11,12]

10 http://commons.wikimedia.org/wiki/File%3ATerpodion2.ogv
11 http://www.klausrohwer.de/privat/hobbies/muha/geschi.htm
12 http://www.bluessource.de/blues-harp-die-geschichte-der-mundharmonika/

2.3.2 Aufbau der Mundharmonika

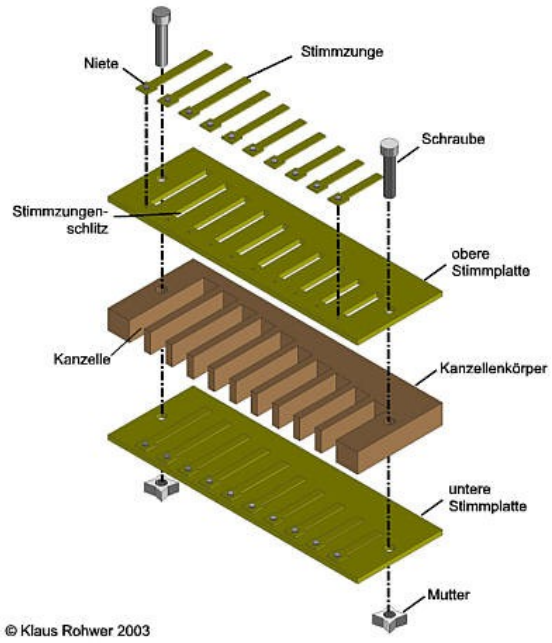

Abbildung 3: Konzeptioneller Aufbau der Mundharmonika (ohne Schutzdeckel)[13]

Es gibt viele verschiedene Typen von Mundharmonikas. Ihr Aufbau hingegen ist in ihren Grundzügen immer gleich. So besteht der Kanzellenkörper, der das Mittelteil der Mundharmonika darstellt, in der Regel aus Kunststoff, Holz oder Metall. In das Mittelstück (Kanzellenkörper) sind zehn quaderförmige Aussparungen eingelassen, die als Kanzellen bezeichnet werden. Diese bilden mit den ober- und unterhalb montierten Stimmplatten die zehn charakteristischen Löcher der Mundharmonika.

Die beiden Stimmplatten sind aus Metall und dienen als Träger für die ebenfalls aus Metall bestehenden Stimmzungen. Von den Stimmzungen gibt es pro Stimmplatte zehn. Diese legen durch ihre Länge den Ton fest, den diese beim Schwingen erzeugen. Die Stimmzungen sind dabei so positioniert, dass ein Ton beim Blasen in die Mundharmonika erzeugt wird und ein anderer Tom beim Ziehen entsteht. Somit sind theoretisch 20 verschiedene Töne vordefiniert.

Damit die Stimmzungen nicht beim Spielen beschädigt werden, sind

zusätzlich Schutzdeckel ober- und unterhalb der Mundharmonika angebracht. Erst dadurch lässt sie sich gut in den Händen halten, denn die Deckel geben dem Instrument seine endgültige Form.

Zusätzlich gibt es bei einigen Mundharmonikas einen Schieberegler. Mit diesem ist es möglich, auch Noten zu spielen, die in den Stimmzungen nicht voreingestellt sind. Weiterhin besitzen diese Mundharmonikas Ventile, die verhindern, dass der Luftstrom beim Spielen nur eine der beiden anliegenden Stimmzungen zur Verfügung steht. Diese Sonderform nennt sich chromatische Mundharmonika. Die eher gebräuchliche Richter-Mundharmonika mit ihrem einfachen Aufbau wird diatonische Mundharmonika genannt.[13]

2.3.3 Die Stimmung macht die Musik

Wie auch das Klavier oder die Flöte ist eine Mundharmonika gestimmt. Das bedeutet, dass das Anspielen (hinein Pusten oder Ziehen), einer bestimmten Kanzelle einen vordefinierten Ton erzeugt. Von einem Grundton aus werden alle anderen Töne nach dem gleichen Muster gestimmt. Somit bleibt das Verhältnis der Töne untereinander immer gleich, aber der grundlegende Klang wird höher oder niedriger angesetzt.[14]

Da eine Mundharmonika im Gegensatz zum Klavier erheblich weniger potentielle Töne zur Verfügung hat, ist man hier auf ein eingeengtes Spektrum begrenzt. Hat man bei einem Klavier in der Regel 88 Unterschiedliche Töne, so sind es auf einer diatonischen Mundharmonika lediglich 20.

Es gibt viele verschiedene Arten, eine Mundharmonika zu stimmen. Zum Beispiel „Country Tuning“, die „Melody Maker™“ oder die „Richterstimmung“.[15] Im Rahmen dieser Arbeit wird ausschließlich die Richterstimmung verwendet, da sie die gebräuchlichste ist und in vielen Anfängermodellen Verwendung findet, da diese für Volksmusik gestimmt

13 http://www.klausrohwer.de/privat/hobbies/muha/muhapubl/muhakunde.htm (2013-01-22)
14 http://kilchb.de/axiomensystem.html (2014-01-31)
15 http://www.angelfire.com/tx/myquill/SpecialTunings.html (2014-01-31)

ist und sich damit auch ohne Vorkenntnisse gut klingende Ergebnisse erzielen lassen.[16]

In der Richterstimmung gibt es 19 verschiedene Töne, obwohl die Mundharmonika 20 anspielbare Kanzellen besitzt. Das liegt daran, dass der Zugton der zweiten Kanzelle genau dem gleichen Ton entspricht, wie der Blaston der dritten.[17]

16 http://www.harmonikaverband.at/muha/mundharmonika_richter.htm (2014-02-06)
17 http://www.klausrohwer.de/privat/hobbies/muha/muhapubl/muhakundeabb/abb02_gesamt.png (31-01-2014)

2.3.4 Mundraummusik

Wer als Anfänger soweit ist, einzelne Töne der Mundharmonika anspielen zu können, kann sich anschließend an einfache Lieder wagen. Allerdings wird er schnell merken, dass es oft vorkommt, dass Noten zu spielen sind, die gar nicht eingestimmt wurden. Die Töne F4, A4 und B6 liegen zwar innerhalb der Grenzen einer auf C gestimmten Richtermundharmonika, lassen sich aber nicht auf den eingestimmten Kanzellen finden:

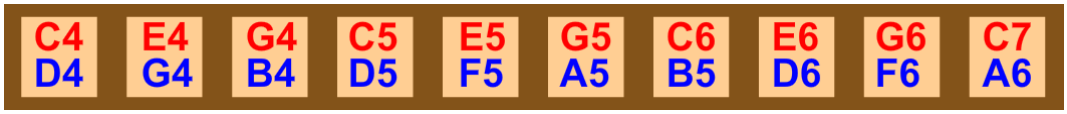

Abbildung 4: Die Richterstimmung in C

Es gibt aber Techniken, die es ermöglichen, alle fehlenden 18 Töne zu spielen. Nicht nur die drei Töne, die der Grundstimmung fehlen, sondern auch all die, die in anderen Stimmungen auftretenden Halbtöne wie C# oder F#. Auf die Funktionsweise der Techniken wird hier nicht weiter eingegangen. Allen ist jedoch gemein, dass sie die fehlenden Töne mit Hilfe einer veränderten Luftsäule im Mund-Rachenraum erzeugen. Wobei einige Techniken schwerer zu erlernen sind als andere. Hier eine schematische Darstellung aller erreichbaren Töne einer in C Gestimmten Richtermundharmonika:

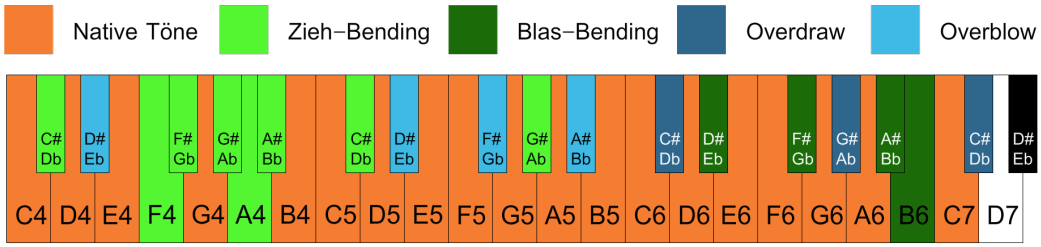

Abbildung 5: Alle erreichbaren Noten der Richterstimmung in C

2.4 Vertreter der Musikspiele

2.4.1 Von Karaoke und Plastikgitarren

Singen unter der Dusche macht Spaß, auch wenn man nicht wirklich singen kann. So verhält es sich auch bei diversen Videospielen. Seit Singstar im Jahre 2004[18] auf den Markt gekommen ist, entwickelte sich ein millionenschweres Business rund um Party- und Musikspiele. Immer wieder wurden Spiele entwickelt, die echte Instrumente wie Gitarren (Guitar Hero), Schlagzeug (Rock Band) oder sogar einen DJ Plattenteller (DJ Hero) verwenden und diese Instrumente dann stark abstrahiert haben, um sie so als Steuerungseinheiten für Musikspiele zu nutzen. Besondere und wichtige Vertreter sollen nun folgend vorgestellt werden.

2.4.2 Singstar[19] und Ultrastar[20]

2004 erschien für die PlayStation2 das Spiel Singstar.[21] In diesem Spiel geht es darum, so gut wie möglich ein ausgewähltes Lied gleichzeitig mit dem Original zu singen. Anders als im klassischen Karaoke hört man hier also nicht nur die Melodie des Stückes, sondern auch den originalen Gesang, während im Hintergrund das dazugehörige Musikvideo läuft.

Dem Spiel liegen ein oder zwei Mikrophone bei, die den Gesang aufnehmen. Die aufgenommenen Töne werden auf dem Bildschirm in eine Balkengrafik umgewandelt. So ist es das Ziel, die vordefinierten Töne zu treffen.

Um das Spiel für hohe wie auch tiefe Stimmen gleichermaßen spielbar zu halten, ist es nicht wichtig, den exakten Ton zu treffen, sondern nur den richtigen Ton innerhalb einer Oktave. So wird ein C4 genauso als richtig erkannt, wie ein doppelt so hohes C5 als allgemeines C erkannt wird.

18 http://dhost.info/ultrastar/faq_allgemein.html (2014-01-22)
19 http://www.singstar.com/de_LU/ (2014-01-22)
20 http://sourceforge.net/projects/ultrastardx/files/ (2014-01-22)
21 http://www.mobygames.com/game/singstar (2014-01-22)

Singstar achtet dabei nicht auf das gesungene Wort, sondern lediglich auf die Frequenz des Tones. Auch kann der Schwierigkeitsgrad eingestellt werden. Ist dieser auf „Leicht“ gestellt, muss man den Ton nicht ganz so exakt treffen, um Punkte zu bekommen.

Abbildung 6: Singstar (PlayStation3) "Auch im Regen" von Rosenstolz.[22]

Je mehr Töne der Spieler trifft, desto mehr Punkte erhält dieser. Das ist ein klassisches Spielelement des High Scores. Den besonderen Reiz des Spieles spürt man oft erst, wenn man mit einem zweiten Mikrophon im Duett oder im Duell singt. So wird dem Spaß gleichzeitig noch eine Note des Rangkampfes hinzugefügt. Dies ist der Grund, der Singstar zu einem oft angetroffenen Partyspiel macht.

Da das Prinzip nicht lange auf sich warten ließ, um kopiert zu werden, wurde 2007 das Spiel Ultrastar im Internet veröffentlicht. Dabei handelt es sich um eine Open Source Version. Das heißt Ultrastar kann kostenlos heruntergeladen und geändert werden. Allerdings kommt Ultrastar ohne vorinstallierte Lieder daher. Diese werden von Fans eigenhändig editiert und auf verschiedenen Internetseiten bereitgestellt.

22 http://www.computerbild.de/artikel/cbs-Tests-PS3-Singstar-2153339.html (2014-01-22)

2.4.3 Guitar Hero und Rockband

Als Guitar Hero 2005 für die PlayStation2 erschien, war es dem Spieler möglich, E-Gitarre zu spielen, ohne je E-Gitarre spielen gelernt zu haben. Der Kontroller der PlayStation2 wurde so modifiziert, dass er wie eine E-Gitarre aussieht. Betrachtet man den Ablauf des Spieles, wird klar, dass es sich um ein ähnliches Prinzip wie auch schon in Singstar handelt. Und zwar geht es auch hier darum, die richtigen Töne zur richtigen Zeit zu treffen.

Nur sind es hier nicht gesungene Töne, sondern Tasten, die die Seiten einer echten E-Gitarre repräsentieren. Diese müssen gedrückt gehalten werden. Wird zusätzlich der Wippschalter am Körper des Gitarrennachbaus betätigt, wird die "Note" als gespielt eingegeben.

Abbildung 7: Guitarhero: Warriors of Rock mit Gitarrenkontroller[23]

Da nach und nach die Absatzzahlen für das Franchise zurückgingen wurde die Serie 2011 eingestellt.

23 Bilder: http://theherofeed.com/4403/july-guitar-hero-dlc-sale-announced/ (2014-01-22) und http://upload.wikimedia.org/wikipedia/commons/thumb/8/80/Guitar_Hero_series_controllers.jpg/800px-Guitar_Hero_series_controllers.jpg (2014-01-22)

Das 2007 erschienene Spiel Rock Band[24] (PlayStation3/XBox360) erweiterte dieses Prinzip um ein Schlagzeug und verband so Gesang, E-Gitarre, Bass und Schlagzeug zu einem Spiel.

2.4.4 Rocksmith[25]

Im September 2012 erschien Rocksmith für die PlayStation3 und XBox360. Das besondere bei diesem Spiel ist, dass der Spieler eine echte E-Gitarre bzw. einen E-Bass nutzt.

So handelt es sich bei diesem Spiel nicht um Entertainment sondern um Edutainment.[26] Das Spiel ermöglicht es dem Spieler das Spielen echter Lieder auf einem echten Instrument. Dabei wird er mit einem automatisch angepassten Schwierigkeitsgrad und durch spielerisches Lernen an das Gitarrespielen herangeführt. Ein Spiel, bei dem nach einiger Zeit klar wird, dass es trotz Spaß immer noch Arbeit ist und viel Zeit und Geduld braucht, ein Instrument zu erlernen.[27]

Abbildung 8: Im Gegensatz zu Guitar Hero gibt es hier mehr als nur 5 Knöpfe.[28]

Im Oktober 2013 veröffentlichte Ubisoft (Herausgeber der Software) die neueste Version der Software unter dem Titel „Rocksmith 2014“, in dem der Schwerpunkt mehr auf das Erlernen als auf das noch in der

24 http://www.rockband.com (2014-01-22)
25 http://rocksmith.ubi.com (2014-01-22)
26 http://wirtschaftslexikon.gabler.de/Definition/edutainment.html (2014-01-22)
27 http://www.youtube.com/watch?v=eGp2p_mAXjo (2014-01-22)
28 http://www.joystiq.com/2011/10/18/rocksmith-review/ (Bild, 2014-01-22)

Vorgängerversion vorhandene Element des Storymodes gelegt wurde.[29]

2.4.5 Synthesia[30]

Wenn es darum geht, autonom Klavier bzw. Keyboard zu lernen, ist Synthesia für den PC das Mittel der Wahl. Die Software bietet die Möglichkeiten eigene MIDI-Dateien (Musical Instrument Digital Interface, Steueranweisungen für ein elektronisches Instrument) in das Programm zu laden, die Stücke in einer beliebigen Geschwindigkeit abspielen zu lassen oder es wartet so lange, bis der Spieler die richtigen Tasten gedrückt hat. Verbunden wird das Keyboard dabei per MIDI-Adapter, was ein Keyboard so zum 88-knöpfigen Kontroller werden lässt.

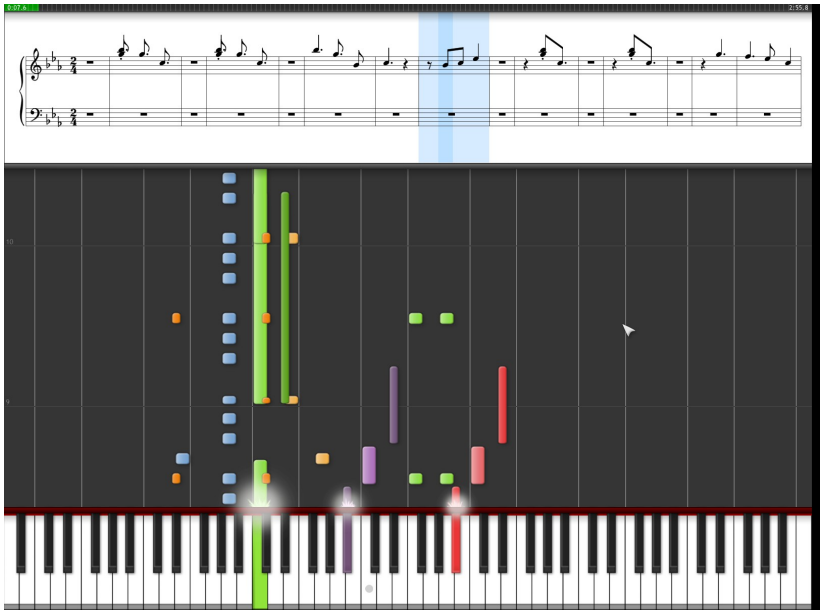

Abbildung 9: Synthesia Screenshot

Synthesia verzichtet hierbei auf Spielerische Elemente wie in den vorherigen aufgezählten Musikspielen. Lediglich ein High Score wurde eingebaut. Synthesia kann somit als reine Lernsoftware betrachtet werden.

29 http://www.youtube.com/watch?v=-aT0C-Etf04 (2014-01-22)
30 http://www.synthesiagame.com/ (2014-01-22)

2.5 Wie man eine Frequenz erhält

2.5.1 Ein Signal ist die Summe seiner Teile

2.5.1.1 Die Sinusfunktion

Wie schon in Kapitel 2.2.1 (Musik besteht aus Tönen) erklärt wurde, ist ein Ton im Grunde nur schwingende Luft, die das Trommelfell im Ohr bewegt. Diese Bewegung des Trommelfells wird dann im Gehirn als Ton interpretiert.[31] Ein Ton und somit seine Schwingung lässt sich grafisch darstellen. Dies geschieht mittels einer Sinuskurve auf einem kartesischen Koordinatensystem. Der Sinus ist der Graph, den man erhält, wenn man die Bahn eines Kreises nachzeichnet und gleichzeitig die Position des aktuellen Punktes auf ein Koordinatensystem abzeichnet. Die folgende Darstellung verdeutlicht dies weitaus besser als die textliche Beschreibung:

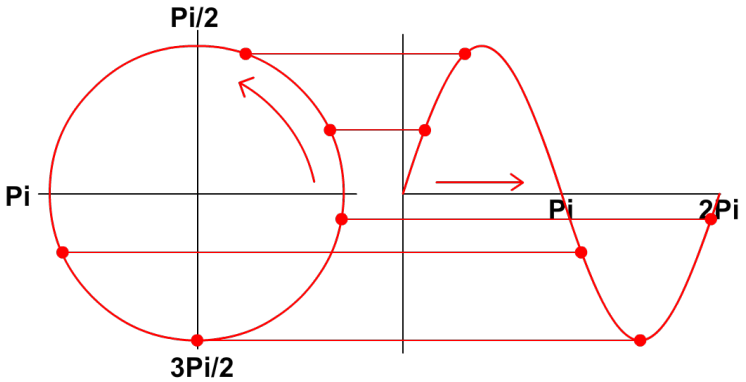

Abbildung 10: Kreisabbildung auf ein Kartesisches Koordinatensystem[32]

Da ein Kreis den Umfang von 2π besitzt, ist die Periode der Sinusfunktion auch 2π lang. Das wiederum heißt, dass sich nach 2π das Muster der Sinusfunktion wiederholt. In einer Formel ausgerückt wäre das:

$$f(x)=\sin(x)$$

31 http://dasgehirn.info/wahrnehmen/hoeren/was-hoert-man-eigentlich-128/ (2014-02-07)
32 http://www.visionlearning.com/en/library/Math-in-Science/62/Wave-Mathematics/131 (2014-02-06) Abbildung abgeändert

Mittels Hz beschreibt man die Frequenz in Schwingungen pro Sekunde. Würde man für die Umrundung des Kreises genau 2π Sekunden benötigen, würde das rund 0,318 Hz bedeuten. Das ist ein für den Menschen nicht hörbarer Ton. Möchte man allerdings einen hörbaren Ton, wie zum Beispiel den Kammerton A4 mit einer Frequenz von 440 Hz, abbilden, so müsste man die Periode eines Kreisumlaufes zeitlich verkürzen.

$$f(x)=\sin(x*2*\pi*Hz)$$
$$f(x)=\sin(x*2\pi*440)$$
$$f(x)=\sin(880\pi x)$$

Als Graph würde dies nun folgendermaßen aussehen:

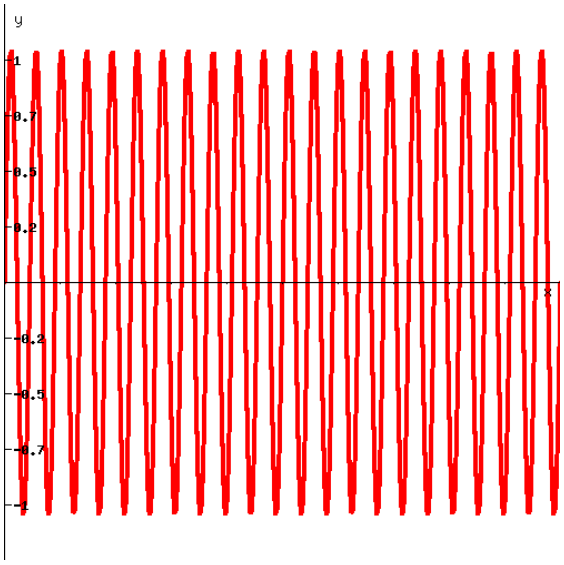

Abbildung 11: Der Ton A4 (440 Hz) die x-Achse geht von 0s bis 0,05s

Da 440 Schwingungen in der Sekunde bei einem Bereich von 0 bis 1 Sekunde nur zu einem einzigen Roten Balken verschmelzen würden, wurde in der Abbildung auf der x-Achse der Bereich von 0 bis 0,05 Sekunden abgedeckt.

Nimmt man mit einer Audiosoftware den Ton auf, würde bei einer genügenden Zoomstufe genau das gleiche Bild entstehen.

2.5.1.2 Mehr als nur ein einzelner Ton

In der Realität gibt es nicht immer nur einen Ton zur gleichen Zeit. Es schwingen die unterschiedlichsten Töne im selben Augenblick an das Ohr oder das Mikrophon. Demzufolge sieht ein reales aufgenommenes Audiosignal auch nicht wie eine wohlgeformte Sinuskurve aus, sondern eher so wie auf der folgenden Abbildung:

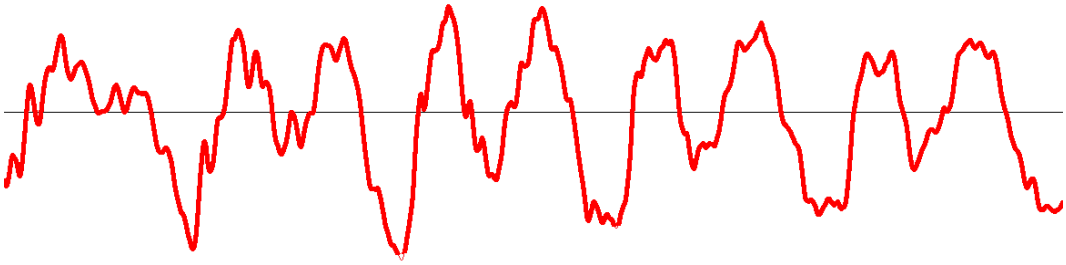

Abbildung 12: Ausschnitt einer Sprachaufnahme aus Audacity. modifiziert

Auch hier wurde der Ausschnitt wieder sehr stark vergrößert, sodass hier nur ein kleiner Teil einer Sekunde zu sehen ist. Und man sieht sehr deutlich, dass hier keine wohlgeformte Sinusfunktion zu sehen ist. Welcher Ton ist das hier also? Die Antwort lautet: Es sind viele verschiedene Töne.

Wie Wellen im Wasser können sich auch Schallwellen addieren und gegenseitig aufheben. Und wie auch im Wasser kann das ein scheinbares Durcheinander geben. Und obwohl es nach einem Durcheinander aussieht, sind es doch alles im Prinzip die gleichen sinusförmigen Wellen mit ihren unterschiedlichen Eigenschaften, die als Ganzes ein Bild ergeben.

2.5.2 Die Spektralanalyse

2.5.2.1 Die Fourierreihe

Baron Joseph Fourier, geboren am 21. März 1768[33] fand heraus, dass sich jede beliebige periodische, also sich wiederholende, mathematische Funktion als eine Reihe von unendlich vielen Sinus- und Cosinusfunktionen darstellen lässt. Die Cosinusfunktion ist die Sinusfunktion um π/4 nach rechts versetzt.

Dabei wird von einer Grundfrequenz ausgegangen, die so lang ist wie eine Periode des endgültigen Signals. Die anderen dazugehörigen Schwingungen sind ein Vielfaches der Frequenz. Sie sind also doppelt, dreifach bzw. x-fach so schnell wie die Grundschwingung. Hier lassen sich Ähnlichkeiten zu der oben erwähnten Klangfarbe erkennen.

So lässt sich auch das folgende Rechtecksignal mittels einer Fourierreihe beschreiben.

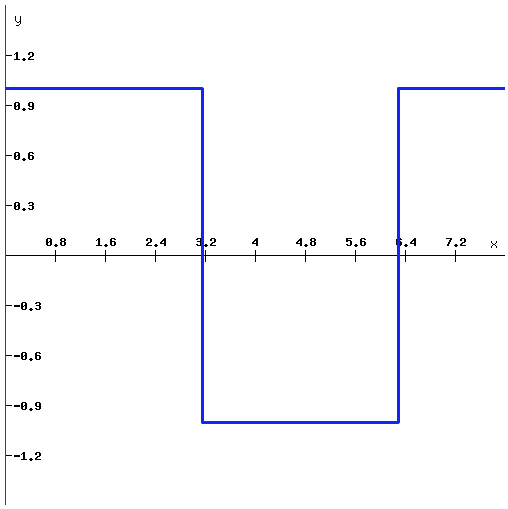

Abbildung 13: Ein Rechtecksignal[34]

33 http://www.britannica.com/EBchecked/topic/215097/Joseph-Baron-Fourier (2014-02-06)

34 Alle Signalabblidungen sind ähnlichen Abbildungen aus dem Skript "Einführung in die Theorie der Fourierreihen und Fouriertransormation" von Prof. Dr. A. Speanhauer und Prof. Dr. M. Steiner-Curtis nachempfunden.

Theoretisch lässt sich dieses Signal mit der folgenden Formel beschreiben:

$$f_n(t)=\frac{4}{\pi}\sum_{k=1}^{n}\frac{1}{2k-1}\sin((2k-1)t)$$

Setzt man für n ∞ (unendlich) ein, würde man theoretisch das Rechtecksignal erhalten. In der Realität kann man n zwar nicht ∞ setzen, dafür aber sehr groß. Je größer n gewählt wird, desto näher kommt man dem endgültigen Rechtecksignal. In den folgenden Abbildungen wurde für n=1, 2, 3, und 15 gewählt.

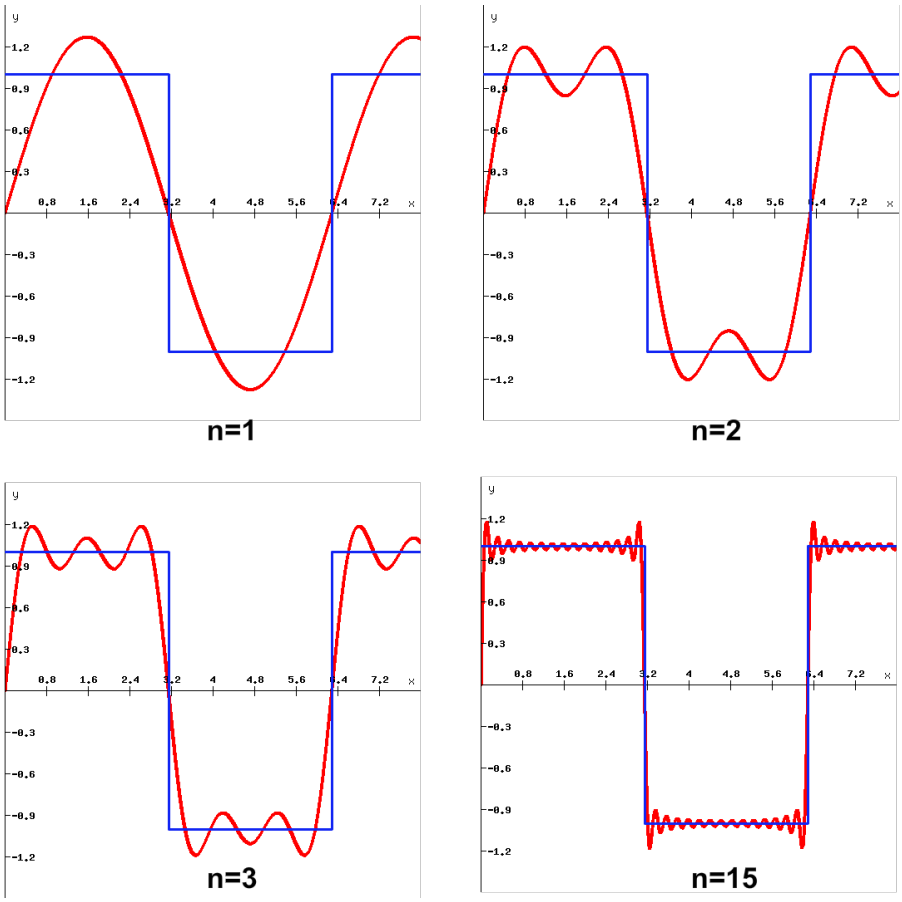

Abbildung 14: Aproximationen an das Rechtecksignal

Der Vollständigkeit halber soll hier die allgemeine Formel für Fourierreihen angegeben werden, auf die aber nicht weiter eingegangen wird, da es den Rahmen der Arbeit überschreiten würde. Genauere mathematische Erläuterungen findet man in der Literatur[35].

$$f(t)=\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos(n\omega_0 t)+b_n\sin(n\omega_0 t)\right)$$

In dieser Formel sind besonders a_n und b_n interessant, denn sie geben an, wie stark die Sinus- bzw. Cosinusfunktion mit gegebener Geschwindigkeit schwingt.

Da das oben aufgezeigte Rechtecksignal punktsymetrisch ist, genau wie die Sinusfunktion, werden auch nur Sinusfunktionen in der Berechnung auftauchen. Würde das Signal sich an der Y-Achse spiegeln lassen, so würde es nur Cosinusfunktionen geben. Möchte man in diesem Beispiel nun herausfinden, wie stark die einzelnen Vielfachen der Grundschwingung an dem Signal beteilig sind, berechnet man dies anhand der folgenden Formel:

$$b_n=\frac{2}{T}\int_0^T f(t)\sin(n\omega_0 t)\,dt$$

35 "Einführung in die Theorie der Fourierreihen und Fouriertransormation" von Prof. Dr. A. Speanhauer und Prof. Dr. M. Steiner-Curtis 2013 (Online Vorlesungsskript) http://web.fhnw.ch/personenseiten/marcel.steiner/Unterricht/Datenanalyse/Fourier-Skript-Marcel-Steiner.pdf

Setzt man dort die Funktion des Rechtecksignals ein und integriert, erhält man dies:

$$b_n = \frac{2}{n\pi}(1-\cos(n\pi))$$

Setzt man hier n=1, so erhält man die Stärke der Grundschwingung des Signals. Setzt man n=3, so erhält man die Stärke der Schwingung, die drei mal so schnell ist, wie die Grundschwingung.

Das die Grundschwingung am stärksten an diesem Signal beteiligt ist, verdeutlicht folgende Grafik:

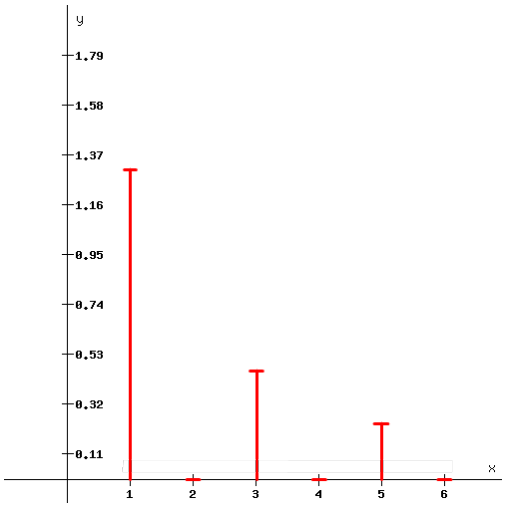

Abbildung 15: Amplituden der beteiligten Sinusfunktionen

Daraus wird deutlich, das nur jeder zweite Oberschwingung am Signal beteiligt ist.

Man erkennt hier sehr schön, dass die Grundschwingung am Stärksten vertreten ist und alle Weiteren enthaltenen Schwingungen mit abnehmender Stärke vorhanden sind.

2.5.2.2 Die Fouriertransformation

Hat man eine periodische Funktion, so lassen sich die einzelnen Schwingungskomponenten mit der Fourierreihe bilden.

Was aber wenn ein Signal nicht periodisch ist, wie zum Beispiel das Signal in der nächsten Abbildung?

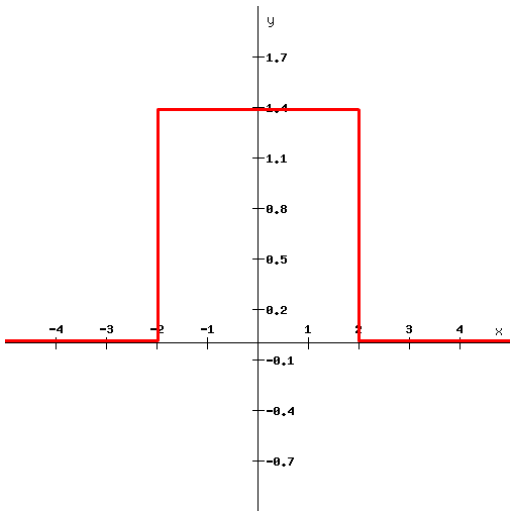

Abbildung 16: nicht periodisches Rechtecksignal

Der Trick, den man hier anwendet, ist, die Periodendauer des Signals unendlich werden zu lassen. Dadurch wird das abgebildete Spektrum immer feiner, bis letztendlich der Abstand zwischen den einzelnen Koeffizienten null ist. Das erinnert an die Integralrechnung, in der die Fläche unter einer Funktion mittels unendlich kleiner Abstände gemessen wird. Was daraus resultiert, ist keine Formel mehr, mit der einzelne Frequenzen in ihrer Amplitude bestimmt werden, sondern eine fließende Kurve, die ein ganzes Frequenzspektrum darstellt.

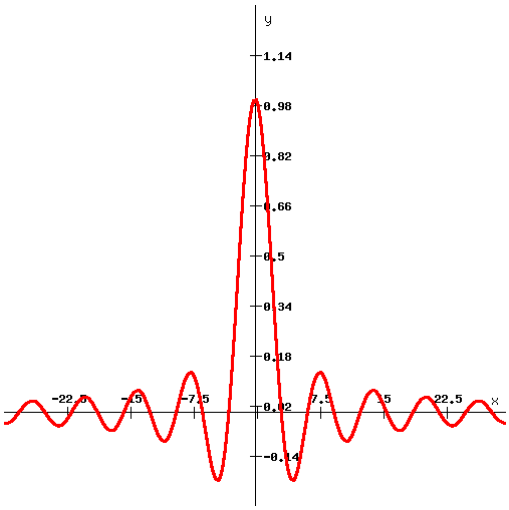

Abbildung 17: Frequenzspecktrum des Rechtecksignals

Dies ist die Fouriertransformation und diese lässt sich wieder in das ursprüngliche Signal umrechnen. Man verändert also nicht das Signal, man ändert nur die Betrachtungsweise.

2.5.2.3 Die Diskrete Fourier-Transformation (DFT)

Mathematische Funktionen zu analysieren, ist eine tolle Sache! Leider sieht die Realität etwas anders aus als die Theorie. Denn die Signale, die ein Mikrophon an den Rechner sendet, sind keine mathematischen Funktionen.

Es verhält sich so, dass in regelmäßigen Abständen überprüft wird, wie weit die Membran des Mikrophons ausschwingt. Was man so erhält, sind einzelne Werte in einem bestimmten Zeitraum. Was zwischen diesen Werten geschieht, ist unbestimmt. Um trotzdem heraus zu bekommen, welche einzelnen Schwingungskomponenten enthalten sind, gibt es die Diskrete Fourier-Transformation (DFT) mit folgender Formel:

$$F(k)=\sum_{n=0}^{N-1} f(n)*e^{-i\frac{2\pi kn}{N}}$$

N entspricht dabei der Gesamtanzahl der Werte. Die jeweiligen Einzelwerte werden durch n angegeben und k sind die Vielfache der

Grundschwingung. Wobei die Grundschwingung immer N entspricht. Warum ist das so?

Die Fouriertransformation kann nur auf periodische Signale angewandt werden da Sinus und Cosinus auch periodisch sind und nicht einfach plötzlich aufhören können. Man nimmt daher die aufgenommenen Werte und geht davon aus, dass sich diese Werte periodisch wiederholen, obwohl das evtl. nicht der Fall ist. Daher ist eine Periode genau N Messungen lang und demzufolge ist auch die Grundschwingung genau N lang. Alle weiteren Schwingungen sind nun das zwei-, drei- oder x-fache der Grundschwingung, nicht aber 0,7 mal, denn dann wäre die Grundschwingung keine Grundschwingung, sondern die Schwingung, die 0,7 mal in die Grundschwingung passt.

Wie oft muss ein Signal abgetastet werden? Abtasten bedeutet, man bekommt einen Wert pro festgelegtem Zeitintervall. Zum Beispiel den Ausschlag der Membran des Mikrophons. Ziel ist es, aus diesen Werten die beteiligten Frequenzen herauszufiltern. Wie sieht es aber hier aus:

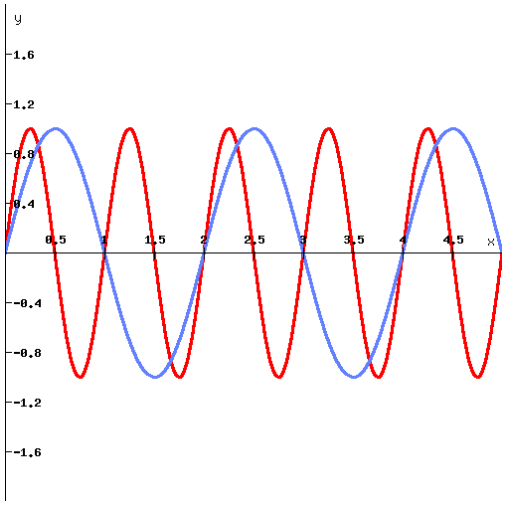

Abbildung 18: Sinus mit einfacher und doppelter Frequenz

Die rote Sinuswelle ist das echte Signal. Würde man nun einmal in der Sekunde dieses Signal abtasten, wären alle Werte gleich 0. Man weiß also nicht, ob es nun das rote Signal ist oder ob vielleicht gar kein Signal

vorhanden ist. Auch kann es sein, dass es das blaue Signal ist welches entdeckt wird.

Wenn man das Signal doppelt so oft abtastet, erhält man auch immer wieder die 0 als Wert. Da das blaue Signal aber bei 0,5 Sekunden 1 ist, und nicht wie gemessen 0, kann das blaue Signal auf keinen Fall in dem abgetasteten enthalten sein. Allerdings weiß man nun immer noch nicht, ob es überhaupt ein Signal gibt. Würde man nun davon ausgehen, dass ein Signal da ist, dann würde es reichen, die Abtastung zweimal in der Sekunde auszuführen. Möchte man sicher sein, so setzt man die Abtastfrequenz höher. Zum Beispiel dreimal in der Sekunde:

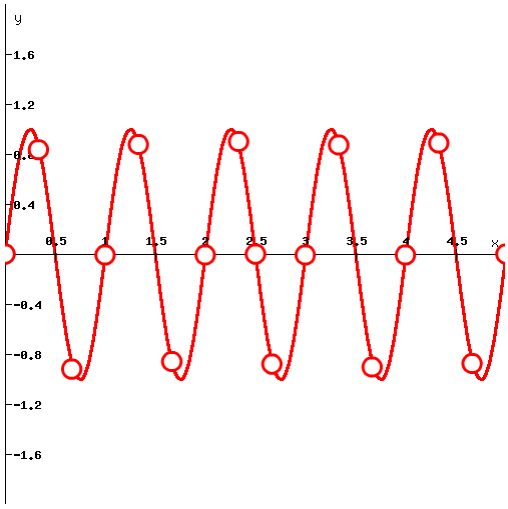

Abbildung 19: Signal wird drei Mal in der Sekunde abgetastet

Nun sieht man, dass es auf jeden Fall ein Signal gibt, denn die Messwerte sind nicht überall 0. Was trotzdem passieren kann, ist, dass es Frequenzen gibt, die höher sind als diese Kurve. Man dreht sich also im Kreis.

Was daraus folgt, ist das Nyquist-Shannon Abtasttheorem.[36] Dieses besagt, dass die Abtastung eines Signals mindestens doppelt so hoch sein muss wie die höchste zu erfassende Frequenz. Möchte man also die Stärke aller auf dem Klavier spielbaren Töne analysieren, muss man

36 http://www.uni-protokolle.de/Lexikon/Nyquist-Shannon_Abtasttheorem.html (2014-02-08)

mindestens 8.372,02 mal in der Sekunde Werte vom Mikrophon bekommen. Das hört sich sehr viel an, ist es aber im Verhältnis zur Abtastrade von CDs nicht. Der Standard mit dem CDs gepresst, werden liegt bei 44.100 Hz. Weiter oben wurde gezeigt, dass der Mensch bis zu einer Frequenz von 20.000 Hz hören kann. Auf der CD wurde dies großzügig beherzigt.

Auf eine ausführliche Rechnung eines Beispiels soll hier verzichtet werden. Stattdessen werden nur die Ausgangslage und das Ergebnis dargestellt:

Nimmt man an, man tastet ein Signal ab und erhält folgende Messungen:

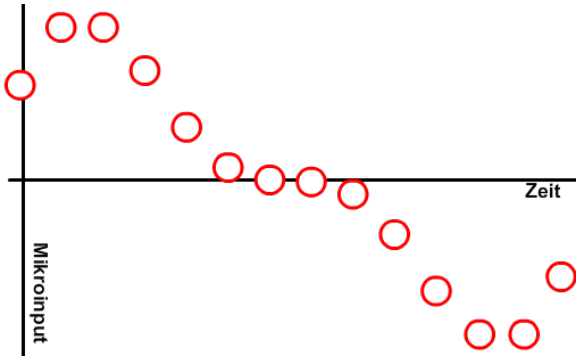

Abbildung 20: Vom Mikrophon abgegriffene Werte über einen Zeitraum.

Ein realer Mikrophone Input wäre, wie oben erwähnt, nicht so schön anzusehen und gliche eher einem Durcheinander. In diesem Falle wurde folgende Funktion zum Abtasten verwendet welche 14 mal pro Sekunde abgetastet wurde:

$$f(x)=2*\sin(x)+\sin(2\mathrm{x})$$

Erwartungsgemäß sollten sich also diese beiden Schwingungen in der DFT abzeichnen.

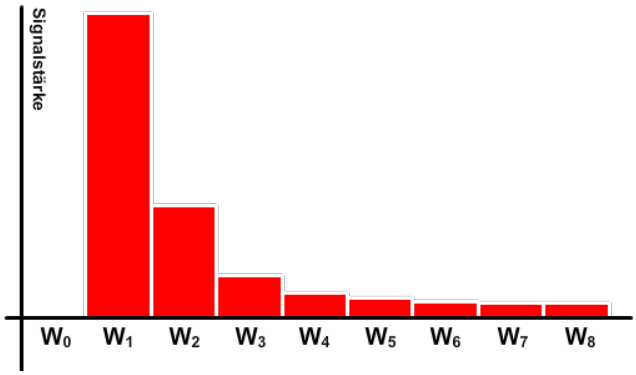

Abbildung 21: Frequenzspektrum der DFT

Der ausschlaggebendste Wert ist W_1, der die Grundschwingung repräsentiert. An Platz zwei befindet sich das Doppelte der Grundschwingung. W_0 ist entspricht der Signalstärke 0, da keine Konstante in der Funktion enthalten ist.

Was nicht zu erwarten war, sind die Werte ab W_3, denn diese dürften gar nicht enthalten sein. Das ist der Leck-Effekt.[37] Er entsteht, weil ein Datenbereich analysiert wurde, der nicht periodisch ist. Auch wenn die diesem Beispiel zugrunde liegende Formel sehr wohl periodisch ist, so sind es die Messwerte nicht. Der Effekt wird umso mehr unterdrückt, je öfter die zu untersuchenden Messwerte künstlich periodisiert werden bzw. das Zeitfenster der Abtastung erweitert wird.

37 http://www.uni-giessen.de/~gd1186/F-Prak/node9.html (2014-02-11)

2.5.2.4 Die Schnelle Fourier-Transformation (FFT)

Auf Grund der vielen komplexen Multiplikationen und Additionen wird die normale DFT mit zunehmenden Werten, die analysiert werden müssen, schnell sehr teuer. Das heißt, bei z.B. vier Messwerten müssen bereits 16 komplexe Multiplikationen ausgeführt werden. Bei zehn Messwerten sind es dann schon 100.

Die reine Diskrete Fourier Transformation hat eine Komplexität von $O(N^2)$. Dass es aber auch effizienter geht, zeigten 1965 John W. Tukey und James Cooley, die mit Hilfe einiger Grundlagen von Carl Friedrich Gauß aus dem Jahre 1805 die Schnelle Fourier-Transformation (Englisch: Fast Fourier Transform, FFT) entwickelten. Mit ihr ist es möglich, den Aufwand auf $O(N \log N)$ zu beschränken.[38]

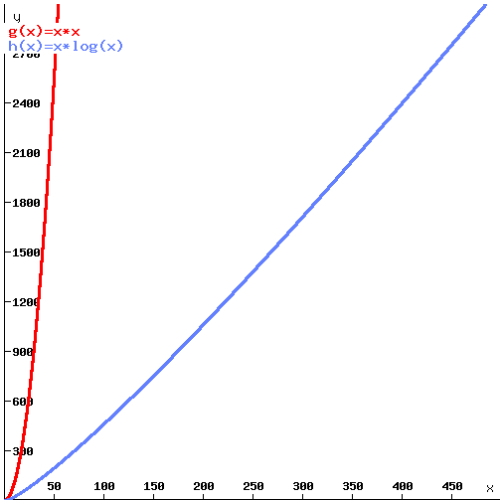

Abbildung 22: In Rot der Aufwand der DFT und in Blau der Auswand der FFT mit zunehmenden Messwerten auf der X-Achse

Die FFT von Cooley und Tukey ist darauf ausgelegt, 2^n Messwerte entgegenzunehmen. Diese Sammlung (wahrscheinlich ein Array) von Werten wird dann in zwei weitere der halben Größe aufgespalten. In eines werden alle Werte mit einem geraden Index abgelegt und die restlichen,

38 http://pauli.uni-muenster.de/tp/fileadmin/lehre/NumMethoden/SoSe2009/Vortrag_FFT_220409.pdf (2014-02-11)

mit den ungeraden Indizes kommen in das verbliebene Array. Danach wird die FFT auf diese beiden Arrays angewandt.

Dieser rekursive Algorithmus wird solange ausgeführt, bis nur noch Einzelwerte vorhanden sind. Da die DFT von einem einzelnen Wert der Wert selbst ist, kann dieser ohne Berechnung zurückgegeben werden. Der zurückgegebene Wert wird nun mit seinem ungeraden Bruder kombiniert und ebenfalls zurückgegeben.

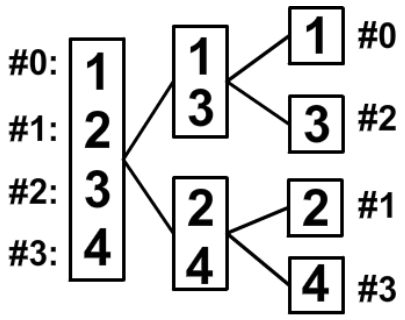

Abbildung 23: Rekursiver Aufruf der FFT

In der Grafik ist der rekursive Aufruf der FFT dargestellt. Jeder Block stellt einen Aufruf dar. Ist die gesamte Aufrufhierarchie abgearbeitet, ist der Algorithmus beendet und gibt die berechnete Frequenzanalyse zurück.[39]

Die rekursive Variante der FFT-Implementierung ist nicht die Einzige. Es gibt eine Vielzahl an Varianten. Einige sind optimiert an bestimmte Rechnerarchitekturen, andere an die Anzahl der übergebenen Messwerte. Eine Bibliothek, die nicht nur eine, sondern viele Varianten umsetzt und die beste für den jeweiligen Anwendungsfall auswählt, ist die FFTW, „Fastest Fourier Transform in the West“.[40]

39 http://youtu.be/EsJGuI7e_ZQ (2014-02-11)
40 http://www.fftw.org/ (2014-02-11)

2.6 Wie lernt es sich am besten?

2.6.1 Was Lernen für das Gehirn ist

Nicht für die Schule lernen wir, nein, für das Leben! Ein Satz, den der Ein oder Andere sicherlich in seiner Schulzeit vom Lehrer gehört hat. Doch stimmt das nicht ganz, nicht für das Leben lernen wir sondern das Leben selbst heißt lernen. Der Mensch hört nie auf zu lernen. Denn lernen ist in erster Linie das Verhalten und Wissen anhand von Erfahrungen zu formen.[41]

Und diese Formung findet im menschlichen Gehirn statt. Beim Lernen werden beteiligte Neuronen (Nervenzellen) über neuronale Netze miteinander verbunden. Was dazu führt, dass das gemeinsame Schalten dieser neu verbunden Neuronen in Zukunft nicht nur von außen ausgelöst werden kann, sondern auch aktiv vom Gehirn selbst. Das ist das aktive Anwenden von Wissen. Dieses steigert sich mit dem Anwenden. Je öfter man etwas Gelerntes in der Praxis umsetzt oder in verschiedenen Zusammenhängen nutzt, desto stärker wird die Verbindung zwischen den betroffenen Neuronen.

So verhält sich das Gehirn genau wie ein Muskel. Nutzt man bestimmte Regionen bzw. wendet man bestimmtes Wissen nicht mehr an, ebben die Verbindungen zwischen den Neuronen ab und das Wissen verschwindet.[42] Das erklärt auch, warum viele nicht wissen, wie man die Fläche eines Kreises berechnen kann, aber trotzdem genau wissen, in welchem Regal im Supermarkt die Nudeln liegen.

Weiterhin werden verschiedene Inhalte auch verschieden gut aufgenommen. So werden beim Lernen verschiedene Hormone freigesetzt. Hat der Lernende Spaß an dem was er tut, schüttet sein Belohnungszentrum Dopamin aus. Spaß heißt in diesem Falle ein regelmäßiges Erfolgserlebnis vorweisen zu können.

41 http://www.lfm-nrw.de/fileadmin/lfm-nrw/Publikationen-Download/Doku41-Spielend-Lernen.pdf (2014-02-11)
42 Gehirngerecht lernen – Eine Lehranleitung auf neuropsychologischer Grundlage; Peter Gasser; 2010

Angst kann auch ein starker Antrieb für das Lernen werden. Das ist aber im Anbetracht der Situation, dass im Rahmen dieser Arbeit ein Spiel entstehen soll, eher hinderlich. Denn Spiele sollen ja Spaß machen.

2.6.2 Wie oft gelernt werden sollte

Wer das Lernen von etwas bestimmten zu seinem Hobby macht, braucht sich keine Gedanken zu machen wie lange er lernen soll. Denn wenn etwas Freude bereitet, tut man es so oft und so lange wie man möchte. Vorausgesetzt, die Zeit die man gern hätte, ist auch wirklich vorhanden. Die Frage nach dem "wie oft?" zeigt zwar den Willen etwas zu beherrschen, nicht aber die Passion oder die Geduld für die Sache.

Was man gern hätte, wäre ein Nürnberger Trichter. Der Nürnberger Trichter wird praktischer Weise an den Kopf gesetzt, dann man das nötige Wissen einfach nur oben hinein zu werfen und schon ist alles fertig gelernt im Gehirn.[43]

Leider gibt es diesen Trichter nicht, sondern ist eher symbolisch anzusehen, und dem Beherrschen oder Verstehen einer Sache geht oft ein langwieriger Prozess des Lernens voraus.

Wie oft nun wirklich gelernt oder sich mit der zu lernenden Sache beschäftigt werden muss, ist nicht zur Gänze erforscht. Ein weiterer Punkt ist, dass jeder Mensch anders ist. Nicht nur physisch, sondern auch psychisch. So hat jeder sein ganz individuelles Vorwissen oder Talent.

Vor allem die musikalische Früherziehung ist ausschlaggebend für das musikalische Talent des willigen Schülers. Bis zu dem Alter von Zehn Jahren ist der Mensch noch sehr aufnahmefähig für Musik und das Lernen fällt sehr viel leichter. So kann im fortgeschrittenen Alter ein höheres musikalisches Niveau erreicht werden.

43 Lernen – Gehirnforschung und die Schule des Lebens; Manfred Spitzer; 2006

Für diejenigen, die eine solche Früherziehung nicht hatten, sei folgende geflügelte Behauptung ein eher deprimierender als motivierender Gehilfe:

Rund 10.000 Stunden muss der Mensch in eine Sache investieren, bis er auf dem Niveau eines Profis ist.[44]

Bei einer halben Stunde des täglichen Übens wäre man also nach etwas über 57 Jahren am Ziel.

2.6.3 Voraussetzungen für gutes Lernen

Für das allgemeine Lernen gibt es viele Tipps und Tricks, die man anwenden kann. Ein Musikstück hingegen gleicht eher einem strickt vorgeschriebenen Ablauf. Note 2 kommt nach Note 1 und vor Note 3. Wann welche Note gespielt wird ist also reines Auswendiglernen. Aber auch das Auswendiglernen lässt sich effizienter gestalten, als das triste Nachspielen einer vorbestimmten Reihenfolge.

Versucht man ernsthaft etwas Neues zu lernen, ist es nicht von Vorteil, dies z.B. auf einem Open Air Festival zu versuchen. Zu viele Eindrücke überfordern das Gehirn und machen es so schwer, etwas Neues aufzunehmen. Sorgt man hingegen für Ruhe oder zumindest für einen neutralen Hintergrund, der nicht viel Neues bietet, ist das Gehirn fokussierter auf die gestellte Aufgabe. Auch Ablenkungen, die das Gehirn zum Multitasking zwingen, sind besser zu vermeiden. Möchte man etwas Neues lernen oder vertiefen, so sollte man sich nur auf diese Sache konzentrieren können.

Da das Gehirn am Tage bereit ist, neue Information aufzunehmen, aber den Schlaf braucht, um diese Informationen einzulagern, ist es ratsam, lieber täglich eine halbe Stunde zu üben, anstatt einmal in der Woche 3,5 Stunden sich der Sache zu widmen. Wer möchte und sich seiner Passion

44 http://www.zeit.de/2008/02/P-John-Sloboda (2014-02-17)

hingibt, kann natürlich auch 3,5 Stunden am Tag üben. Solange die eigene Motivation da ist, steht dem nichts im Wege.

Lernen funktioniert immer frei nach dem Motto: „Use it or loose it!“. Wenn man sich nicht oft genug dehnt, wird man immer unbeweglicher. Genauso verhält es auch mit dem Gelernten. Wendet man es nicht an oder frischt es auf, so verfällt es nach einer gewissen Zeit. Die neuronalen Netze lösen sich auf und werden so anders vielleicht besser verwertet. Möchte man also ein Instrument und bestimmte Lieder lernen, ist es ratsam diese, regelmäßig zu wiederholen. Auch wenn ein Lied Stück für Stück gelernt wird, sollte man zunächst mit dem beginnen, was man schon weiß. um dieses Wissen aktiv wieder zu aktivieren und somit stärker im Gehirn zu verankern.

Auch ist es nicht unbedingt ratsam, sich gleich zu Beginn das ganze zu lernende Musikstück merken zu wollen. Das Gehirn hat nur einen begrenzten Platz für neue Informationen. So soll man sich das zu Erlernende lieber einteilen und jedes Mal etwas weiter gehen als beim letzten Mal.

Das Gehirn „freut“ sich, wenn es weder unter noch überfordert ist. Mit einer leichten Erweiterung der Grenzen der eigenen Fähigkeiten können schnell neue und motivierende Erfolgserlebnisse geschaffen werden. Das ist ein Punkt, der auch für den Flow-Effekt zuständig ist. Dabei vergisst man die Welt um sich herum komplett und kann sich ganz der Aufgabe widmen. Unter Programmierern sagt man, man sei in einem Tunnel.[43]

Zusammenfassend kann man also folgendes über ein gutes und effektives Lernverhalten sagen:

- lieber mehrmals ein bisschen lernen, als zu viel auf einmal (besser täglich als wöchentlich)
- immer in einer reizarmen bzw. störungsfreien Umgebung lernen
- den Umfang des Lernmaterials nicht zu hoch ansetzen
- das bereits Gelernte immer wiederholen
- Erfolge stellen sich schneller ein, wenn man mit Freude lernt

3. Die Software

3.1 Ziel im Rahmen der Arbeit

Innerhalb weniger Wochen eine vollausgereifte und nutzerfreundliche Software zu entwickeln, ist in normalen Ausmaßen nicht möglich. So wird im Rahmen dieser Arbeit lediglich ein Prototyp eines spielerischen Lernprogrammes zum Nachspielen eines einfachen Liedes entwickelt, der die Kernfunktionen der Tonerkennung umsetzt, da dies der wissenschaftliche Schwerpunkt dieser Arbeit darstellt. Theoretisch wird hingegen das Thema des spielerischen Aufbaues der Software beschrieben. Sprich, welche grafischen Nutzeroberflächen in einer fertigen Version vorhanden sein sollten, wie diese verschachtelt sind und welchem Zweck diese dienen. Auch werden Funktionen, die nicht im Prototypen umgesetzt werden, erwähnt und in ihrem Umfang zumindest angerissen.

Am Ende dieses Kapitels wird auf eine besondere Art des Softwaretests eingegangen, die man in der Computerspieleindustrie am häufigsten antrifft. Dabei handelt es sich um den Nutzer als Testroutine bzw. Um viele Nutzer als Softwaretester.

3.2 Struktur und Aufbau der Implementierung

3.2.1 Auswahl des Frameworks

Ein Spiel von Grund auf entwickeln zu wollen, kann sich als aufwändiges Unterfangen herausstellen. Denn ein Spiel besteht in der Regel aus mehr als nur aus einer Kommandozeilenoberfläche oder ein paar Standardbuttons des Betriebssystems.

Ein Computerspiel besteht heutzutage in der Regel aus mindestens diesen fünf Komponenten[45]:

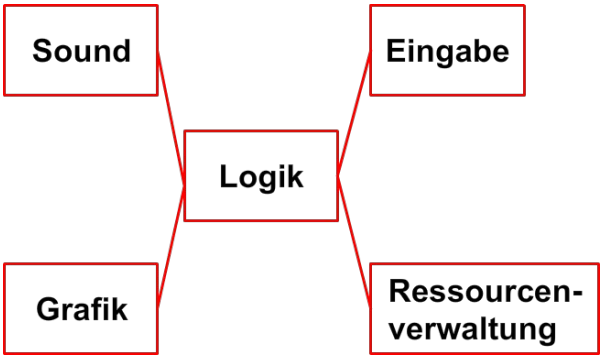

Abbildung 24: Grundlegende Komponennten eines Computerspiels

Die in der Grafik stark abstrahierten Komponenten setzen sich je nach Komplexität des umzusetzenden Projektes aus vielen Unterkomponenten zusammen. Nach einem Beispiel in dem Buch „Game Engine Architecture“[46] aus mehr als 120 Teilsystemen. Diese von Grund auf selbst zu entwickeln, ist nicht nur sehr aufwändig und erfordert Expertise in diversen Fachbereichen, es ist dementsprechend auch teuer. Darum greift man oft auf fertige Spiele-Frameworks zurück, welche dementsprechende Bibliotheken bereitstellen. Diese Frameworks können sehr rudimentär sein und dem Entwickler trotz allem noch viel eigene Arbeit übertragen (Pixi.js[46]) oder sie sind sehr speziell und können den Entwickler sogar auf

45 Abbildung Stark modifiziert nach: Game Engine Architecture; Jeff Lander und Jason Gregory von Taylor & Francis Ltd. (2009)

46 http://www.pixijs.com/ (2014-02-17)

ein bestimmtes Genre festlegen (RPG Maker[47]).

Die Wahl des Frameworks hängt von vielen Faktoren ab. Welche das in dem Fall für das Spiel dieser Arbeit ist, wird im Folgenden dargestellt. Dafür ist es zunächst notwendig, einige Begriffe zu definieren.

Implementierungsaufwand
Wie schnell lassen sich ausreichend gute und sichtbare Ergebnisse produzieren?

Vorerfahrung des Entwicklers
Hat der Programmierer bereits Erfahrung mit dem Framework oder der zugrunde liegenden Programmiersprache?

Portabilität
Da das Spiel auch für mobile Geräte erhältlich sein soll ist eine möglichst schnelle und reibungslose Portierung zumindest auf die Betriebssysteme Android und iOS wichtig.

Kosten
Diese sollen möglichst niedrig sein. Optimal sind gar keine Kosten für Nutzung und Publizierung der erstellten Software.

Vermarktungsrechte
Ist das Framework darauf ausgelegt, damit auch Projekte umzusetzen, welche eine kommerzielle Nutzung möglich machen? Muss das Framework genannt werden oder muss man sogar den Source Code offenlegen?

In folgender Tabelle sollen einige vorausgewählte Frameworks verglichen werden.

47 http://www.rpgmakerweb.com/ (2014-02-17)

Framework	Implementierungsaufwand	Vorerfahrungdes Entwicklers	Portabilität	Kosten	Vermarktungsrechte
Pixi.js	- da Mikrophoneingabe in HTML5 nicht vollständig unterstütz wird, ist implementierung nur aufwändig für Android und iOS möglich	- Programmiersprache: JavaScript (HTML5) - kleinere Spieleprojekte bereits umgesetzt. (50h+)	- mittels zusätzlichen Frameworks portabel zu Android und iOS	- keine Kosten	- Pixi.js ist komplett Open Source und unterliegt keinen Auflagen - Freie Nutzung ohne Erwähnung des Frameworks ist möglich.
Starling[48]	- volle Unterstützung von Grafiken durch Verwandtschaft zu Adobe Flash - prototyping geht innerhalb weniger Stunden dank einfacher API (Schnittstellen, siehe Abkürzungsverzeichnis)	- Programmiersprache: ActionScript3 - mittelgroße Ein- und Zweimannprojekte umgesetzt, rund sechs monatige Erfahrung	- portabel auf Web, Android und iOS ohne größere Probleme	- Entwicklungsumgebung "Adobe Flash Builder" kostet 61,49 € im Monat. - Sonst keine weiteren Kosten	- Starling ist komplett Open Source und unterliegt keinen Auflagen. - Freie Nutzung ohne Erwähnung des Frameworks ist möglich - Eine Adobesignatur ist allerdings zwingend
libGDX[49]	- erste Tests ergaben sehr schnelles Erstellen eines Prototyps. Binnen weniger Stunden (unter 4) waren erste Tonerkennungen, -analysen und grafische Repräsentationen möglich	- Programmiersprache: Java - keine Vorerfahrung mit dem Framework allerdings mit Java	- bringt von Haus aus sehr einfache Portabilität mit dank eigenem Projektgenerator. Android und iOS Versionen werden sozusagen gleich mit gebaut	- keine Kosten	- libGDX ist komplett Open Source und unterliegt keinen Auflagen - Freie Nutzung ohne Erwähnung des Frameworks ist möglich
Cocos2-x[50]	- sehr fehleranfälliges Framework. Was auf der einen Plattform ohne Probleme läuft, lässt eine andere abstürzen	- Programmiersprache: C++ - ein mittelgroßes Projekt umgesetzt (150h+)	- von Grund auf darauf ausgelegt Desktop, Android, iOS, Windows 8 und HMTL5 zu unterstützen	- keine Kosten	- Cocos2d-x ist komplett Open Source und unterliegt keinen Auflagen - Freie Nutzung ohne Erwähnung des Frameworks ist möglich.

Tabelle 1: Vergleich verschiedener Spiele Frameworks

48 http://gamua.com/starling/ (2014-02-24)
49 http://libgdx.badlogicgames.com/ (2014-02-24)
50 http://www.cocos2d-x.org/ (2014-02-24)

Das Framework Pixi.js scheidet auf Grund von unzureichender Implementierung der Mikrophonfunktionalität von Browser aus. Auch beschränken sich die zur Verfügung stehenden Funktionalitäten ausschließlich auf Grafik und Usereingabe.

Mit Starling hat der Entwickler zwar viel Erfahrung und kann auf einen Fundus von fertigem Code zurückgreifen, doch steht dem Entwickeln der Preis von rund 60 € monatlich im Weg.

Eine Pein in der Einrichtung und Entwicklung ist das Framework Cocos2d-x. Zwar hat der Entwickler genug Erfahrung mit dem Umgang, jedoch ist es schwierig, dieses Framework kontinuierlich anzuwenden, da zu viele kryptische Fehlermeldungen auftauchen, die jedes Mal erst recherchiert werden müssen.

Obwohl keinerlei Erfahrung im Umgang mit libGDX vorhanden ist, erwies sich das erste Erstellen eines Prototyps als sehr effizient und ohne größere Fehleranfälligkeiten auf Windows und Android.

Demzufolge ist die Wahl des Frameworks zur Entwicklung der Mundharmonika-Lernsoftware auf libGDX gefallen.

3.2.2 Vorgehensweise der Tonerkennung

3.2.2.1 Trennung von Spiel und Tonanalyse

Das, was der Nutzer auf dem Bildschrim sieht, ist nicht immer genau das, was sich im Hintergrund abspielt. Das Einlesen von genügend Messwerten des Mikrophons braucht Zeit. Das ist Zeit, in der das Programm im wahrsten Sinne des Wortes wartet. Und wenn das Programm wartet, ist es nicht mehr möglich eine flüssige Animation von 60 Bildern pro Sekunde dem Nutzer zur Verfügung zu stellen. Und das würde ein vernünftiges Spielerlebnis nicht ermöglichen.

Daher sind die Logiken von Spielelementen, die auf dem Bildschirm angezeigt werden und das Einlesen und Analysieren der Messwerte des Mikrophons in einzelnen Threads verlagert.

Ein Problem ist, dass das Spiel, welches die analysierten Messwerte nutzt, jederzeit die Messwerte abrufen kann. So kann es passieren, dass Werte in der Analyse geschrieben sind, die aber noch nicht endgültig von dem Analysecode bearbeitet wurden. So würde man Wertearrays bekommen, die unfertige und falsche Werte enthielten.

Der Trick, dem sich hier bedient wird, ist der, dass man zwei Arrays mit fertigen Werten verwaltet und einen als Rückgabewert bereithält, während das andere Array bearbeitet wird. Ist das Zweite fertig, wird dieses bei Aufruf der entsprechenden Funktion zurückgegeben.

Dies ist eine Technik, die in den meisten Anwendungen mit dynamischen Grafiken ihren Platz findet. Dort besser bekannt unter dem Namen "Double Buffering".[51]

51 http://gameprogrammingpatterns.com/double-buffer.html (2014-02-22)

3.2.2.2 Rauschunterdrückung

Auch wenn die Umgebung des Nutzers ruhig ist, so würde man doch ein als Rauschen erkennbares Signal wahrnehmen, wenn man diese Stille mit einem Mikrophon aufnimmt. Dieses Rauschen ist von Mikrophon zu Mikrophon unterschiedlich stark ausgeprägt und kann die einwandfreie Erkennung eines gespielten Tones stören.

Auch können diverse Hintergrundgeräusche, wie eine Parkatmosphäre, die im Ganzen einen dem Rauschen sehr ähnlichen Zustand haben, störend auf die eigentliche Analyse wirken.

Daher ist es ratsam, den allgemeinen Störpegel, der zu dem Zeitpunkt der Benutzung vorherrscht, zu analysieren, um ihn im weiteren Verlauf heraus rechnen zu können.

Um das Rauschen zu filtern, muss allerdings ein Rauschprofil ermittelt werden. Dazu wurde ein gemittelter Wert mehrerer Spektralanalysen gewählt.

Das heißt, dass über einen gewissen Zeitraum die FFT auf das Signal angewendet und addiert wird. Anschließend wird der Mittelwert über alle Spektren gebildet. Daraus entsteht ein allgemeines Rauschmuster, das für jedes einzelne Frequenzband eine Unterdrückung zur Verfügung stellt.

Die Dauer der Analyse kann weit unter einer Sekunde liegen, um gute Ergebnisse zu erzielen. Es wird für den Nutzer aber anschaulicher sein, wenn dieser Vorgang mindestens eine Sekunde dauert. So hat er die Chance, der Kalibrierung mittels einer kleinen Meldung beizuwohnen.

Eine andere Möglichkeit wäre es gewesen, nicht das Spektrum zu analysieren sondern nur den Ausschlag des Mikrophonpegels. Das ist rechnerisch zwar einfacher umzusetzen, bringt aber auch schlechtere Erfolge, da so für alle Frequenzen die Empfindlichkeit in gleichem Umfang gemindert wird und so schwache Signale schwächer werden, obwohl dies gar nicht nötig wäre.

3.2.2.3 Einlesen mit Hilfe eines Ringspeichers

In Kapitel 2.5.2.3 (Die Diskrete Fourier-Transformation (DFT)) wurde dargelegt, dass die Abtastrate eines Signals mindestens doppelt so hoch sein muss wie die höchste zu erkennende Frequenz. Da dem grundliegendem Prinzip des Programmes, die Tonerkennung, eine gewisse Vielseitigkeit innewohnen soll, wird das Klavier als Referenz genutzt. Der höchste Ton, den ein gleichgestimmtes Klavier erzeugen kann, hat eine Frequenz von 4.186,01 Hz. Die Mindestfrequenz der Abtastung des Mikrophons läge damit bei 8.372,02 Hz. Auf dem Desktop ist eine Abtastrate von 8.500 Hz ohne weiteres möglich, auf Androidgeräten erweist sich das als schwierig. Es macht den Anschein, als ob verschiedene Geräte nur bestimmte Abtastraten zulassen.[52]

Die Standard-CD-Rate von 44.100 Hz wurde allerdings von allen Testgeräten anstandslos angenommen. Daher fällt die Wahl auf eben diese Frequenz.

Da in dem fertigen Programm eine Lernfunktion für das Bending (siehe Kapitel 2.3.4 Mundraummusik) bereitstellen soll, ist es wichtig eine gewisse Granularität gewährleisten zu können. Das heißt, dass zwischen den einzelnen Frequenzen der Töne auch die restlichen in einer gewissen Auflösung erkennbar sein müssen.

So sollen nicht nur die Frequenzen von zum Beispiel 261 Hz und 277 Hz erkannt werden sondern auch Teilschritte von der einen zur anderen Frequenz.

Aus diesem Grunde ist es nötig, einen Datenpuffer von 16.384 Werten zu halten. Bei dieser Anzahl an Messwerten können mindestens 5 Schritte zwischen zwei Tönen erkannt werden.

Den Puffer aufzufüllen, würde so rund 0,37 Sekunden dauern. Ein flüssiges Spielerlebnis wäre damit nicht mehr möglich. Die Aktualisierungsrate der Tonerkennung wäre so zu gering.

52 Getesten auf Google Nexus 7 und Samsung Ace II

Um die Aktualisierungsrate höher zu halten, wurde der Datenpuffer mittels eines Ringspeichers[53] realisiert, welcher eine bestimmte Anzahl an Segmenten zu je einer festgelegten Anzahl an Messwerten beinhaltet.

Mittels eines Ringspeichers ist es möglich, die ältesten Werte eines Datenblocks mit neuen Werten zu überschreiben und so immer einen Vorrat der letzten n aktuellen Werte bereitstehen zu haben.

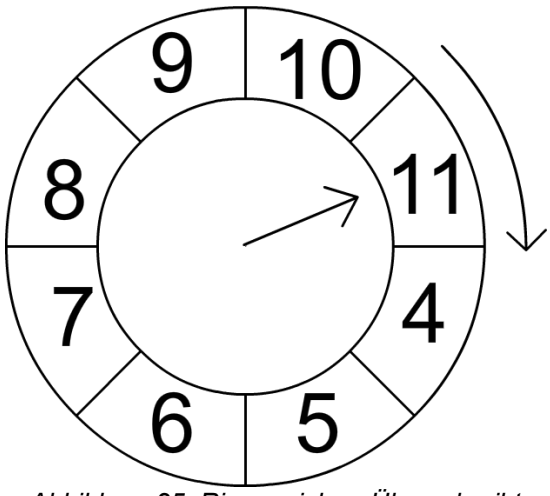

Abbildung 25: Ringspeicher: Überschreibt älteste Daten

Legt man so eine Segmentgröße auf 512 Werten fest, kann der analysierte Ton 32 mal öfter pro Sekunde aktualisiert werden.

Das Problem, welches sich dabei ergibt, sind erkannte Töne, die gar nicht mehr da sind. Da der älteste Datensatz in dem Ringspeicher rund eine drittel Sekunde alt ist.

Man muss sich also entscheiden, ob man eine sehr genaue Abstufung des Tons haben möchte oder eine geringe Länge der Phantomsignale.

Das Hauptprogramm arbeitet auch sehr korrekt mit einer weitaus geringeren Auflösung des Ringspeichers. So funktioniert die Erkennung mit 4.096 Messwerten hinreichend gut und das älteste Ringsegment ist

53 http://www.embedded.com/electronics-blogs/embedded-round-table/4419407/The-ring-buffer (2014-02-19)

damit nur rund eine zehntel Sekunde lang, was ein flüssiges Spielen ermöglicht.

So werden im fertigen Programm zwei Ringspeicherkonfigurationen genutzt. Diese sind abhängig von der Anwendung.

Das Üben von Bendingtechniken nutzt einen Ringspeicher von 16 Segmenten á 1.024 Messwerten.

Das Spielen von Liedern nutzt einen Ringspeicher von 4 Segmenten á 1.024 Messwerten.

3.2.2.4 Anwendung der FFT

Nachdem ein neues Segment des Ringspeichers mit Daten vom Mikrophon gefüllt ist, werden die Daten des gesamten Ringes mit Hilfe einer Implementierung der Fast Fourier-Transformation (FFT) in das Frequenzspektrum des Signales umgewandelt. Der im Abschnitt 2.5.2.4 (Die Schnelle Fourier-Transformation (FFT)) kurz beschriebene rekursive Algorithmus wird dabei nicht verwendet. Stattdessen wird eine iterative Implementierung genutzt. Das heißt, die Funktion ruft sich nicht selbst aus, sondern wird nur einmal aufgerufen. Eine rekursive Funktion ist im Allgemeinen hübscher anzusehen und kleiner, aber belastet den Stack des auszuführenden Rechners. Und kommt dieser an seine Grenzen, bricht das Programm ab und der Nutzer eines alten mobilen Android-Gerätes kann das Programm nicht absturzfrei nutzen.

Da das ausgesuchte Entwicklungs-Framework libGDX ist, welches die Programmiersprache Java nutzt, ist die ebenfalls in Kapitel 2.5.2.4 (Die Schnelle Fourier-Transformation (FFT)) erwähnte Bibliothek "fftw" hierfür nicht geeignet, da diese in C implementiert ist. Aus Gründen der höchst möglichen Portabilität wurde daher auf eine reine in Java geschriebene Variante der FFT von Damien Di Fede[54] genutzt.

54 https://github.com/libgdx/libgdx/blob/master/extensions/gdx-audio/src/com/badlogic/gdx/audio/analysis/FFT.java (2014-02-22)

3.2.2.5 Konvertierung in angespielten Ton

In den vorherigen Kapiteln wurde aufgezeigt, wie die Signale vom Mikrophon aufgenommen, gespeichert und in ihr Frequenzspektrum überführt werden. Diese Vorarbeit ist nötig, um das eigentliche Anliegen des hier entwickelten Programmes umsetzen zu können. Dabei geht es darum, herauszufinden, welcher Ton auf einer Mundharmonika gespielt wird. In folgender Abbildung wird das bereits fertig bearbeitete und dann in Frequenzspektrum umgewandelte Signal dargestellt.

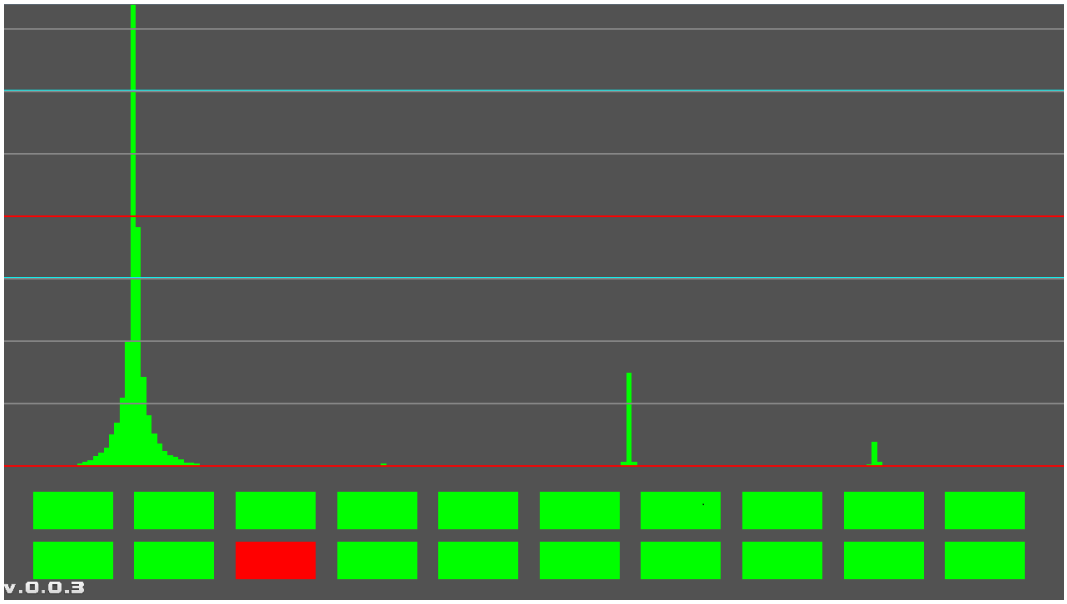

Abbildung 26: Debug-Ansicht des Prototypen

Diese Grafik zeigt die Debug-Ansicht für das Testen der Software. Unten im Bild sind 20 grüne Rechtecke platziert, welche die Kanzellen der Mundharmonika repräsentieren. Zu sehen ist, dass an der dritten Kanzelle gezogen wird (rotes Rechteck). Dies ist das Ergebnis der Analyse des Spektrums.

Zunächst wird das Spektrum von der FFT-Funktion entgegengenommen und daraufhin untersucht, ob einer der Werte des Spektrums den Schwellwert überschreitet. Wie hoch dieser ist, wird am Anfang des Programmes durch ein Testpusten in die Mundharmonika ermittelt.

Wird der Schwellwert überschritten, wird der höchste Wert des Spektrums

gesucht. Sobald dieser gefunden ist, werden alle Werte anhand dieses höchsten Wertes normiert. Das heißt, dass der höchste Wert das Maximum darstellt und demzufolge 1 ist. Alle anderen Werte werden darauf hin in Relation zum Maximalwert neu berechnet. Alle Werte sind danach also zwischen 0 und 1.

Das ist sehr nützlich, da man so immer ein genormtes Signal erhält auch wenn der Nutzer plötzlich lauter als sonst spielt (zum Beispiel in dem er näher an das Mikrophon gerückt ist oder stärker in die Mundharmonika pustet oder daran zieht).

Wie in der Abbildung oben zu sehen ist, gibt es drei Ausschläge (vier, wenn man den kleinen noch dazu zählt), das sind die Obertöne des ganz links stark ausschlagenden Grundtons. Einer dieser Ausschläge ist der wirklich gespielte Ton. Durch das Prinzip der Grund- und Obertöne muss der Ausschlag mit der niedrigsten Frequenz der Ton sein, der vom Nutzer gespielt wird. Denn Obertöne sind immer nur ein Vielfaches des Grundtones, nicht aber ein Teiler dessen.

Mittels eines zweiten Schwellwertes wird nun nach der ersten Frequenz gesucht, die diesen Schwellwert überschreitet. Dies ist dann der vom Programm als gespielt erkannte Ton. Der Schwellwert wird in der Abbildung als Zyan-farbene Linie dargestellt (zwischen den beiden roten Linien).

Meistens ist der lauteste Ton auch der Grundton und damit der von Nutzer gespielte. Manchmal kommt es aber vor, dass eine Oberschwingung lauter ist als die Grundschwingung. Dadurch kann es passieren, dass der eigentlich gespielte Ton unterhalb des zweiten Schwellwertes liegt und damit eine Oberschwingung als Grundschwingung identifiziert wird.

Dieses Problem kann mit einem niedrigen Schwellwert allerdings gut behoben werden.

Einzelne Ausreißer lassen sich aber ab und an nicht vermeiden.

Der "getroffene Ton" ist aber nur eine Frequenz. Genauer gesagt entspricht dieser Ton der mittleren Frequenz, je nachdem wie gut das Spektrum aufgelöst ist.

Diese Frequenz wird nun mit der eingestellten Mundharmonika abgeglichen. Da die Frequenzen allerdings leicht abweichen können, muss ein kleiner Zwischenschritt eingefügt werden.

In Kapitel 2.2.2 (Bitte nach Vorschrift spielen) wurden zwei Formeln vorgestellt, diese ermöglichen es, eine Frequenz in eine bestimmte Taste der Klaviatur umzuwandeln und für eine bestimmte der Klaviatur die Frequenz auszurechnen. Mit ihrer Hilfe ist es nun möglich, die gespielte Frequenz der Analyse zu nehmen und sie in die Taste der Klaviatur umzurechnen, die am ehesten angespielt wurde. Ist das Layout der Mundharmonika nun auch dementsprechend eingestellt, dass man es nicht mit den Frequenzen der spielbaren Töne, sondern mit den Tastennummern der Klaviatur versieht, lässt sich das ganze sehr schön und einfach abgleichen.

```
public class HarpLayouts {

    public static int[] RICHTER_STANDARD = {40, 44, 47, 52, 56, 59, 64, 68, 71, 76,
                                            42, 47, 51, 54, 57, 61, 63, 66, 69, 73};

}
```

Hier ist die Konfiguration der Richterstimmung in C angegeben. In der oberen Reihe finden sich die Tastennummern, die den Blastönen entsprechen und in der unteren Reihe die, die den Ziehtönen entsprechen.

3.2.2.6 Knapp daneben ist nicht so schlimm

In Kapitel 2.3.4 (Mundraummusik) wurde bereits über die Technik des Bending gesprochen. Dabei werden Töne zwischen dem Blaston und dem Ziehton der Mundharmonika durch die Modulation des Mundraumes

erschaffen, um so auch Noten spiele zu können, die nicht direkt auf der Stimmung des Instrumentes beruhen. Natürlich kann es auch vorkommen, dass ein ungeübter Spieler zufällig diese Zwischentöne spielt. In dem Fall hat der Nutzer zwar in die richtige Kanzelle geblasen oder gespielt, der richtige Ton wird aber nicht erkannt. Das ist zwar korrekt aber auch deprimierend und der Nutzer kann die Software als fehlerhaft beurteilen.

Damit dies nicht passiert, wird dem Nutzer einer von zwei Tonerkennungsmodi zur Wahl gestellt. Diese gleichen einem Schwierigkeitsgrad. In der schwierigen Einstellung muss der Ton möglichst genau getroffen werden. In der leichten Einstellung sucht die Software den Ton auf der Mundharmonika, der der erkannten Frequenz am nächsten ist.

Dafür wird mittels Satz des Pythagoras ($a^2 + b^2 = c^2$) die Differenz der einzelnen Frequenzen der Kanzellen mit der gespielten verglichen und die kleinste als vom Nutzer "gemeinten" Ton interpretiert.

3.2.3 Das Dateiformat der Musikstücke

3.2.3.1 Schnell, simpel und unbekannt

Es gibt verschiedene Arten Musik aufzuschreiben. Die bekannteste ist sicherlich die schriftliche Notenform. Doch versteht ein Rechner die Art der Darstellung nicht besonders gut, da sie für den Menschen konzipiert wurde. Für den Rechner ist eine Form in ASCII[55] oder Binärform deutlich einfacher zu lesen.

Da der Wunsch, Musik schriftlich festzuhalten, nicht erst mit dieser Arbeit entstanden ist, sondern von anderen Menschen bereits vorher, gibt es einige bereits definierte Formate zum digitalen abbilden von schriftlichen Noten.

Als Beispiele sind hier die beiden Textformate MusicXML[56] und ABC[57] zu nennen.

MusicXML ist, wie der Name bereits vermuten lässt, ein auf XML basierendes Format. Das heißt, dass Noten und ihre Eigenschaften innerhalb von sogenannten "Tags" eingeschlossen sind. Eine Note wird zum Beispiel so dargestellt:

```
<note>
  <pitch>
    <step>C</step>
    <octave>4</octave>
  </pitch>
  <duration>4</duration>
  <type>whole</type>
</note>
```

Hier wird die Note C4 als ganze Note im 4/4 Takt dargestellt. Was auffällt ist, ein sehr großer Overhead. Das heißt es sind Informationen vorhanden, die zusätzlich zur Kerninformation vorhanden sind.

55 http://www.ascii-code.com/ (2014-02-23)
56 http://www.musicxml.com (2014-02-23)
57 http://abc.sourceforge.net/ (2014-02-23)

Folgend soll nun die gleiche Note in dem Format ABC dargestellt werden:

```
c
```

Richtig, lediglich der Buchstabe "c" ist hier abgebildet. In diesem Falle gibt es gar keinen Overhead. Das liegt daran, dass in beiden Beispielen nur die reine Darstellung der Noten aufgezeigt wurde. Das Lied "Alle meine Entchen" würde hingegen in ABS folgendermaßen ausgedrückt werden.

```
T:Alle meine Entchen
M:2/4
L:1/8
K:C
C D E F | G2 G2 [|: A A A A | G4 :|]
F F F F | E2 E2 | G G G G | C4 |]
```
[58]

Man sieht hier bereits, dass etwas mehr dazugehört als nur die Namen der Noten. Auch werden einige Attribute festgelegt, wie Titel (T), Takt (M) oder das Vorkommen der meistvertretenden Notenlänge (L).

Nach einigen Überlegungen wurde das Format ABC in der vorliegenden Arbeit gewählt. Allerdings wurde dieses Format nicht im Original übernommen, sondern es wurde nur als Stütze gewählt, um ein einfacheres kleines Dateiformat zu entwerfen.

Der Grund dafür liegt darin, dass ABC trotz seiner Einfachheit eine Vielzahl an Symbolen unterstützt, die jeweils ausgewertet werden müssen. Daher wurde eine eingedampfte Version entwickelt, die es ermöglicht, schnell Lieder aufzuschreiben. Allerdings wurde sich hier die Simplizität wiederum durch einen etwas größeren Overhead erkauft. Das oben in

58 http://penzeng.de/Geige/Abc.htm (2014-02-23)

ABC dargestellte Lied würde in diesem neuen Format so aussehen:

```
T:Alle meine Entchen
C: Unbekannt
B:200
M: 2/4
D: 3

S: C58 D58 E58 F58 G54 G54
A58 A58 A58 A58 G2
A58 A58 A58 A58 G2

F58 F58 F58 F58 E54 E54
G58 G58 G58 G58 C2
```

Auf den ersten Blick sieht diese Variante vielleicht etwas unüberschaubarer für das menschliche Auge aus. Die Noten setzen sich folgendermaßen zusammen:

Der Notenname, zum Beispiel C5, direkt gefolgt von der Notendauer. Hierbei steht "1" für eine ganze Note und zum Beispiel 8 für eine Achtelnote. Zwischen den einzelnen Noten muss mindestens ein Whitespace Zeichen vorhanden sein. Folgend werden alle Attribute und ihre dazugehörige Verwendung aufgelistet.

Attribut	Bedeutung
T	Titel des Liedes
C	Autor des Liedes
B	Taktschläge pro Minute
M	Takt des Liedes
D	Wert zwischen 1 und 10, gibt die allgemeine Schwierigkeit des Liedes an
S	Muss das letzte Attribut sein. Nachfolgende Zeichen sind das eigentliche Lied
$	Alles hinter einem "$" wird als Kommentar gewertet

In der Notation ist es auch möglich, Pausen zu definieren. "P2" ist zum Beispiel eine Pause in der Länge einer halben Note.

Durch die einfach gehaltene Struktur des Formates ist es nicht möglich, jegliches Musikstück abzubilden. So fehlen unter anderem Triolen oder Haltebögen, mit denen es möglich ist, auch Noten zu verwenden, die länger als eine ganze Note sind.

Sich nur auf wenige grundlegende Eigenschaften zu beschränken, ist in Ordnung. Denn dieses Format dient lediglich der schnellen Implementierung und bietet dem Nutzer eine kleine Hilfe, um eigene Lieder in das Programm zu übertragen.

3.2.3.2 MIDI (Musical Instrument Digital Interface)

Im Jahre 1983[59] entwickelte man eine Schnittstellenspezifikation, die viele Jahre lang die digitale Musikindustrie formen sollte. "Das Musical Instrument Digital Interface" oder eher bekannt unter den Namen "MIDI".

MIDI-Dateien beinhalten, kurz gesagt Steuerbefehle für digitale Instrumente. Nur eben nicht wie ABC in lesbarer ASCII-Form, sondern rein binär. Dabei ist es möglich, auf verschiedenen Spuren verschiedene digitale Instrumente anzusteuern und so sogar komplexe Stücke abzubilden. Durch ihre geringe Größe wurden MIDI-Dateien lange Zeit in der Computerspieleindustrie verwendet, um Sound und Musik zu modulieren. Eigentlich kann man auf MIDI-Dateien keine Stimmen abbilden, sondern nur Instrumente und wie lange sie welchen Ton mit welcher Lautstärke spielen sollen. Künstlern ist dies allerding in einem gewissen Umfang trotzdem möglich.[60]

Desktop wie auch mobile Geräte unterstützen das MIDI in der Hinsicht, dass sie diese abspielen können. Es eignet sich hervorragend dazu, nicht nur ein Musikstück abzubilden, sondern dieses auch mit einfachen programmatischen Mitteln dem Nutzer hörbar zu machen.

59 http://unseretollepage.de/hosted/midiguide/vorwort.html (2014-02-23)
60 http://www.youtube.com/watch?v=muCPjK4nGY4 (2014-02-23)

Daher wäre ein über einen Protoytpen hinausgehendes Programm die Unterstützung von MIDI-Dateien eine willkommende Bereicherung.

Dabei ist es ratsam, bei MIDI-Dateien, welche mehrere Klangspuren beinhalten, soweit runter zu brechen, dass nur eine Spur herausgenommen und für das hier entstehende Programm genutzt wird. Denn hier geht es darum, jeweils nur einen einzelnen Ton zu spielen.

In Kombination mit der Software werden die meisten MIDI-Dateien also nicht viel Sinn ergeben. Von daher wäre es eine Idee, dass es auch einen eigenen Editor gäbe, der dem Nutzer auf eine bestimmte Weise strukturierte Dateien zur Verfügung stellen könnte.

Um den Durchschnittsnutzer aber nicht weitergehend zu verwirren, kann man die Dateiendung der MIDI-Datei, welche in der Regel ".mid" oder ".midi" ist, abändern. So denkt der Nutzer, es handelt sich um ein ganz eigenes Format und wird nicht so schnell auf die Idee kommen herkömmliche MIDI-Dateien in das Programm laden zu wollen.

3.3 Aufbau für den Nutzer

3.3.1 Usability

Usability lässt sich am besten mit dem Wort "Benutzerfreundlich" übersetzen. Das bedeutet, dass dem Nutzer die Interaktion mit dem Programm möglichst einfach gemacht wird.

Nach ISO 9241.11 (Ergonomie der Mensch-System-Interaktion) gelten folgende Anforderungen[61] an eine gute Benutzerfreundlichkeit für Softwaresysteme:

- leicht zu erlernen
- intuitiv zu benutzen
- geringe Fehlerrate
- zufriedenheit sicherstellen

Das bedeutet, das Spiel muss dem Nutzer auch zugänglich sein, wenn dieser noch gar keine Ahnung vom Spielen der Mundharmonika hat. Außerdem müssen die Menüs und Anweisungen so einfach und genau wie möglich sein. Je selbsterklärender etwas ist, desto weniger zusätzliche Beschreibung ist notwendig, um dem Nutzer dort hin zu bringen, wo er hinmöchte.

Weiterhin ist es wichtig, dass der Nutzer das bekommt, was er erwartet. Klickt er auf "Spielen", möchte er spielen nicht erst noch drei Konfigurationsbildschirme durchlaufen. In diesem Fall wäre die Beschriftung "Weiter" oder "Einstellungen" passender.

Am Ende ist es aber die allgemeine Zufriedenheit des Nutzers selbst, die den Wert der Software ausmacht.

61 http://www.handbuch-usability.de/iso-9241.html (2014-02-22)

Eine komplett perfekte Software die jeden Nutzer gleichermaßen zufrieden stellt, gibt es nicht. Gerade weil jeder Mensch etwas anders auf das vor sich Befindliche reagiert.

Die Aufgabe der Umsetzung der Usability ist es, dem Nutzer eine möglichst gute Erfahrung ermöglichen zu können.

Wie gut diese Umsetzung funktioniert, wird im Kapitel 3.4 (Der Nutzer als Testroutine) näher erläutert.

3.3.2 Vom Start zum Spiel

Das Wichtigste für eine angenehme Nutzbarkeit ist es dem Nutzer einen einfachen und kurzen Weg zur eigentlichen Funktion zu bieten. Das heißt, er soll schnell ein Lied nachspielen können.

Daher ist es von Bedeutung, dass vor allem der erste Start des Programmes mit wenig Interaktion direkt ins Spiel führt. Folgendes Diagramm stellt einen Erststart der Software dar.

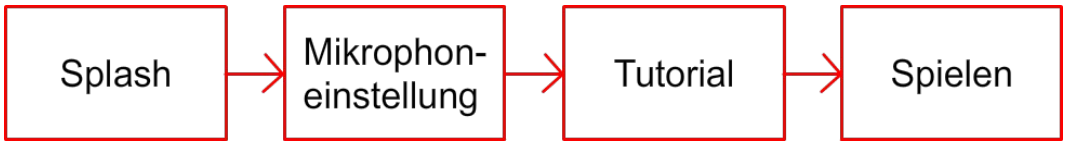

Abbildung 27: Ablauf des ersten Starts.

In dem Diagramm steht ein Rechteck für einen angezeigten Bildschirm. Zunächst wird dem Nutzer ein sogenannter Splashscreen gezeigt. Auf diesem wird, wie in einem Kinofilm, das Entwicklerlogo präsentiert, was in erster Linie zu Werbezwecken des Softwareentwicklers dient. Dieser sollte nicht länger als zwei bis drei Sekunden dauern.

Danach geht es zum Konfigurationsbildschirm. Hier wird dem Nutzer mitgeteilt, dass er eine ruhige Umgebung aufsuchen soll und dann die Umgebungsanalyse mittels des Berühren des Bildschirmes (Tippen auf einen Kreis) durchführt. Danach wird der Nutzer aufgefordert, in seine Mundharmonika zu pusten, um die Lautstärke zu messen. Anschließend wird der Nutzer bei dem ersten Start des Programmes nach einem weiteren Tippen zum Tutorialbildschirm geleitet.

Mittels des Tutorialbildschirmes wird dem Nutzer erklärt, was die Symbole im anschließenden Spiel bedeuten und wie er darauf zu reagieren hat. Ist der Nutzer der Meinung, die Anweisungen verstanden zu haben, kommt er wieder mittels Tippen auf einen Button mit der Beschriftung "Spielen" in den Lernmodus für ein einfaches Lied.

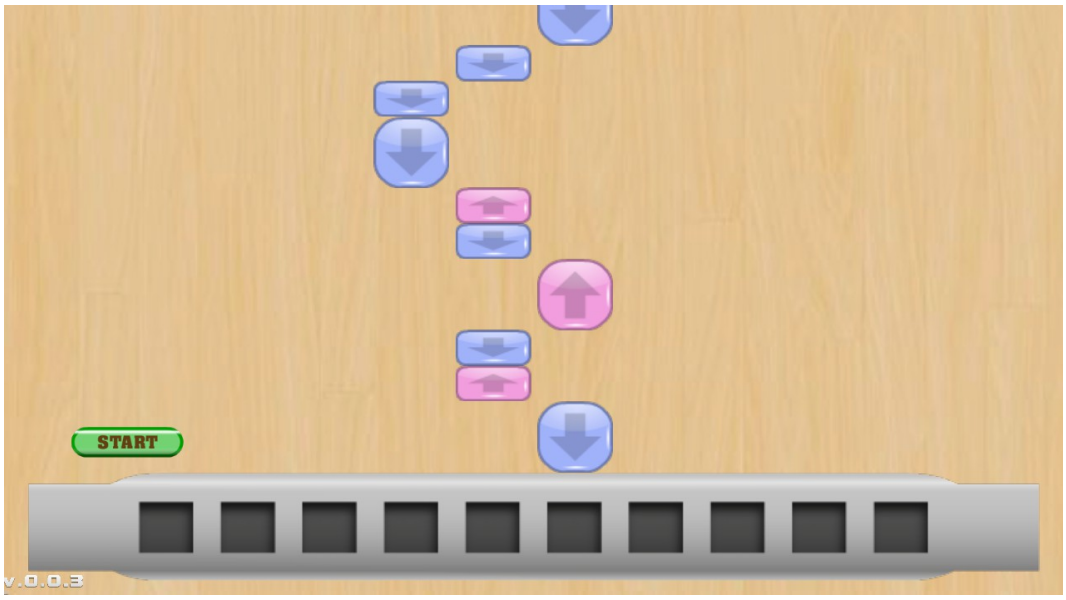

Abbildung 28: Die blauen Felder bedeuten Ziehen und die roten bedeuten Pusten.

Hier in der Abbildung 28 wird der Spielbildschirm des implementierten Prototypens dargestellt. Der Nutzer sieht direkt, wie er in welche Kanzelle seiner Mundharmonika pusten oder an ihr ziehen muss. Im Lernmodus wartet das Programm auf den richtigen Ton, wird dieser erkannt, gleiten die abstrahierten Noten auf dem Bildschirm nach unten und der Nutzer muss die nun an der Mundharmonikagrafik anliegende Note spielen.

Hat der Nutzer das Einführungslied durchgespielt, ist es ihm möglich, mittels eines Buttons in das eigentliche Hauptmenü des Spieles zu gelangen.

Folglich soll der normale Ablauf für den Nutzer vom Start des Programmes dargestellt werden.

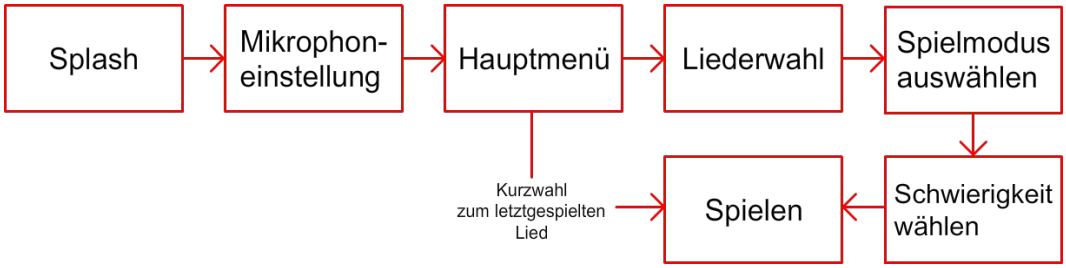

Abbildung 29: Normaler Ablauf vom Start der software bis zum Spielen eines Liedes.

Wie in der Abbildung zu sehen, ist der Ablauf nun etwas komplexer. Der Splash und die Mikrophonkonfiguration bleiben gleich. Danach ändert sich das Verhalten der Software. Anstatt einer Erklärung wird nun ein Hauptmenü angezeigt. Von diesem aus kann der Nutzer unter anderem das letzte von ihm gespielte Lied, inklusive der dafür eingestellten Parameter, sofort ansteuern oder ein neues Lied auswählen und einstellen.

Durch die Kurzwahl hat der Nutzer eine schnelle und praktische Möglichkeit dort weiterzumachen, wo er beim letzten Mal der Benutzung des Programmes aufgehört hat. Und der Ablauf erinnert an den des ersten Programmstates.

Wenn der Nutzer ein neues Lied wählt, hat er anschließend die Wahl zwischen zwei Spielmodi:

Das Lied lernen

Dieser gleicht dem Modus aus dem Tutorial, das Programm wartet darauf, das der Nutzer den richtigen Ton spielt und fährt dann weiter.

Das Lied mitspielen

Hierbei fahren die Noten automatisiert durch das Bild und der Nutzer muss im richtigen Moment den richtigen Ton treffen.

Nachdem der Nutzer den Spielmodus gewählt hat, wird er vor eine weitere Wahl gestellt. Er muss die Schwierigkeit des Liedes wählen.

Folgende Schwierigkeitsstufen stehen ihm dabei zur Auswahl:

Leicht

Dem Nutzer werden alle Noten an der richtigen Position angezeigt. Auch wird ihm gezeigt, ob er in die Mundharmonika pusten oder an ihr ziehen soll.

Mittel

Dem Nutzer wird zwar angezeigt, welche Kanzelle er nutzen muss, aber nicht, ob er in sie hinein pusten oder an ihr ziehen muss.

Schwer

Der Nutzer erhält die Information, ob er pusten oder ziehen muss, nicht aber an welcher Kanzelle der Mundharmonika er dies tun soll.

Nachdem diese Einstellung getroffen wurde, gelangt der Nutzer zum bekannten Spielebildschirm, in welchem er nun das Lied üben oder spielen kann.

Diese Einstellungen sind nun gespeichert und können vom Hauptmenü aus direkt angewählt werden, um so viele Klicks bis zum Kern, das Spielen selbst, zu gelangen.

3.3.3 Helfende Funktionen für den Nutzer

3.3.3.1 Achievements

Mit den oben beschriebenen Funktionen, wäre die Software vollständig. Zumindest würde sie so den Grundgedanken des Programmes umsetzen.

Trotzdem ist es nicht verkehrt, ein paar Anreize und Hilfestellungen zu schaffen, die das Lernen oder die Motivation dabei noch ein bisschen unterstützen könnten.

Beim Spielen eines Videospieles geht es oft um Erfolge. Den Endboss besiegen, die Prinzessin retten oder einfach nur Erster zu werden. Dieser Erfolg entspricht einer Belohnung. In Form einer Siegesanimation, des Spieleabspanns oder der simplen Gewissheit, besser als andere zu sein.

Mit dem Aufkommen der sogenannten Casual Games, der Gelegenheitsspiele wie Bejeweld Blitz, welche oft kein echtes Ende haben und sehr repetitiv sind und gleichzeitig mit immer komplexeren Spielen für die es oft über 40 Stunden dauerte, bis man sie durchgespielt hatte, tauchten immer häufiger sogenannte Achievements auf.[62]

Dabei handelt es sich um kleine Zwischenerfolge, die zwar keine direkte Auswirkung auf das eigentliche Spiel haben, aber der allgemeinen Motivation dienen. Auch gibt es Achievements, welche nur sehr schwer zu bekommen sind. Dabei verhält es sich ähnlich den diversen Orden beim Militär. Man kann sie zur Schau stellen.

Abbildung 30: Achievement in dem Spiel "Halo: Reach" (Benachrichtigung unten-mittig)[63]

Eine einfache Form von Achievements kann auch in dem hier entstehenden Programm eingesetzt werden. Es handelt sich dabei um einfache Medaillen der Stufen Bronze, Silber und Gold.

62 Wann und wo Achivements ihren Ursprung hatten war leider nicht genau herauszu finden. Es ist aber anzumerken, dass mit der Downloadplattformm "Steam" von Valve, der PlayStation3 von Sony und der X-Box 360 Achievements einen festen Platz in der Spieleindustrie einnahmen. Also lässt sich sagen, dass Achievements ab mitte 2006 zum Quasi-Standard der Spieleindustrie geworden sind.

63 http://www.gamesradar.com/halo-reach-achievement-guide/ (2014-02-23)

Da es drei verschiedene Schwierigkeitsstufen gibt, erhält man beim fehlerlosen Absolvieren der Stufe eine entsprechende Medaille (Einfach: Bronze; Mittel: Silber; Schwierig: Gold). Da es zwei Spielmodi gibt, kann man pro Lied insgesamt sechs Medaillen erlangen.

3.3.3.2 Komfort in der Titelwahl

Auch bei der Auswahl bzw. Übersicht aller zur Verfügung stehenden Lieder zum Lernen kann man den Nutzer auf vielfältige Weise und mit sinnvollen Informationen unterstützen.

Zunächst eine Skizze des Titelauswahlbildschirmes:

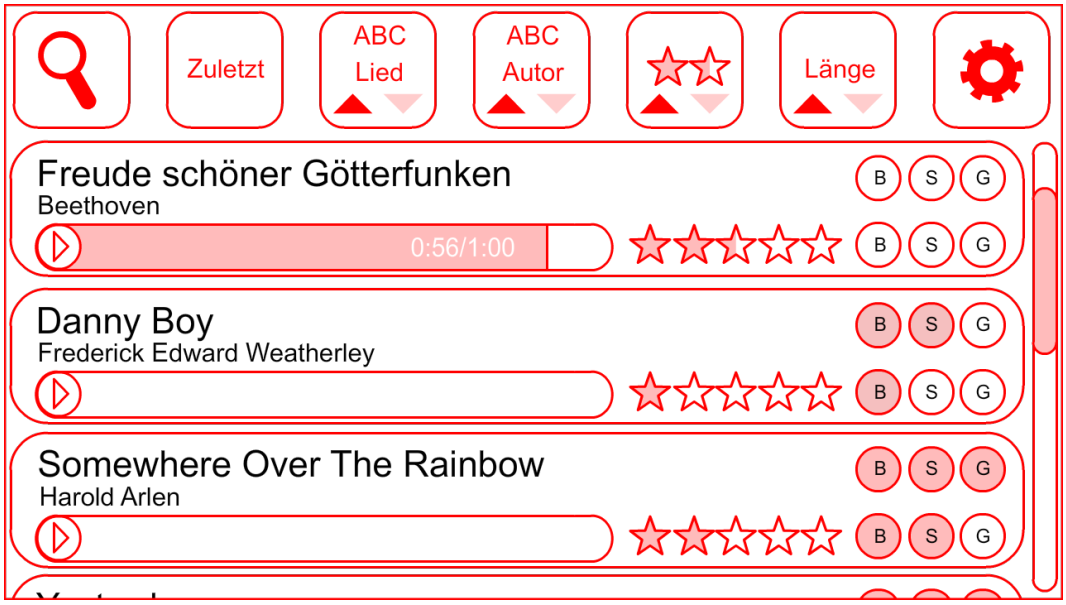

Abbildung 31: Entwurf des Titelauswahlbildschirmes

Zugegeben, in diesem Entwurf sieht alles etwas überladen aus. Um den Nutzer nicht mit zu vielen Informationen zu überhäufen, ist es möglich, nach und nach neue Informationen den einzelnen Titelfeldern hinzuzufügen. Die allgemeine Schwierigkeitsstufe eines Titels wird durch die Sterne symbolisiert. Die sich rechts befindenden Medaillen zeigen, was man bei dem Titel schon erreicht hat. Die Darstellung des Musikstückes, des Autors sowie die Möglichkeit sich das Stück vorher anhören zu können, sollten hingegen von Anbeginn zur Verfügung stehen.

Auch wurde darauf geachtet, dass dem Nutzer eine sinnvolle, aber auch Vielfältige Auswahl an Sortiermöglichkeiten direkt zur Verfügung steht, um schneller das Lied zu finden, welches er sucht.

3.3.3.3 Unterstützung beim Bending

Für fortgeschrittene Nutzer werden auch dementsprechende schwierigere Spieltechniken interessanter. So sollte es auch die Möglichkeit geben, diese Techniken zu üben. Da es bei diesen Techniken darum geht, Töne auf der Mundharmonika zu spielen, die nicht auf den einzelnen Stimmzungen eingestellt sind, ist es sinnvoll, eine Anzeige in das Programm zu implementieren, die anzeigt welcher Ton gerade vom Nutzer erzeugt wird. So kann man auch überprüfen, ob die verwendete Mundharmonika auch der Stimmung entspricht, von der der Nutzer ausgegangen ist.

Zur groben Illustration soll auch hier ein Konzept dienen:

Abbildung 32: Konzept des Bildschirms für das Üben von Zwischentönen

Auf diesem Bildschirm sind nur wenige und damit die nötigsten Informationen vorhanden. Anhand der Klaviatur kann man direkt sehen, welche Taste dem gespielten Ton entsprechen würde. Auch wird der

momentan gespielte Ton mit Namen groß in der Mitte der Bildschirmes angezeigt. Die Halbtöne (D#6 und F#6), die hier abgebildet sind, können auch mit ihren Äquivalenten Eb6 und Gb6 betitelt werden (oder beides).

Ändert der Nutzer die Form seinen Mund-Rachenraum um zum Beispiel das Bending zu üben, verändert sich leicht der Ton. Das kann dargestellt werden, in dem der untere Pfeil, welcher auf die Klaviatur zeigt, in Richtung des nächsten Tones verschoben wird. Auch die angezeigten Notennamen können größer und zentrierter dargestellt werden je eher der dementsprechende Ton getroffen ist. Man kann dies mit dem Dock in den Apple Desktop-Betriebssystemen vergleichen.

3.3.3.4 Immer zugängliche Optionen

Es kann immer sein, dass die Erkennung der Töne nicht korrekt genug stattfindet. Das kann an einer fehlerhaften Mikrophonkonfiguration liegen oder weil sich die Umgebungsbedingungen geändert haben. Daher ist es wichtig, von so vielen Stellen wie möglich im Programm schnell auf die Option der erneuten Kalibrierung zurückgreifen zu können. Weiterhin soll man auch jederzeit ins Hauptmenü zurückkommen können. Bei Androidgeräten kann auch der Back-Soft-Button genutzt werden, um schnell zu vorherigen Menüpunkten zu gelangen.

3.3.4 Wie man neue Lieder bekommt

3.3.4.1 Die rechtliche Lage

Ein Programm, mit dem man Lieder lernen kann, ist nicht viel Wert, wenn es keine Lieder gibt, die man spielen könnte. Man darf allerdings nicht einfach jedes Lied, das man findet, runterladen und nutzen. Wer den neuesten Hit in den Charts lernen will, wird zunächst nicht drum herum kommen, eine entsprechende Datei im richtigen Format aufzutreiben (welche Formate das sind, ist in Kapitel 3.2.3 Das Dateiformat der Musikstücke dargelegt). Und selbst wenn man diese Stücke findet, bleibt immer noch die Frage zu klären, ob man diese Stücke runterladen und benutzen darf.

Laut dem Urheberrechtsgesetz (UrhG)§64 gilt ein Werk 70 Jahre nach dem Tod des Urhebers als gemeinfrei und kann kostenlos und im vollen Umfang, auch kommerziell, verwendet werden. Das betrifft viele sehr alte Werke, wie zum Beispiel die Kompositionen Beethovens oder Bachs.

Dies sind Stücke, die also auch ohne weitere rechtliche Fragen mit der Software ausgeliefert werden können, da diese keinem Urheberrecht mehr unterliegen.

Anders verhält es sich bei Werken, welche noch nicht so alt sind. Auch wenn man ein Lied nach Gehör rekonstruiert, liegen die Kompositions- und Vertriebsrechte beim jeweiligen Lizenzgeber. Meist ist dies die Plattenfirma, unter der das Lied produziert wurde. Auch das starke Reduzieren eines Stückes ist nach UrhG §23 ohne die Erlaubnis des Urhebers nicht gestattet.

Möchte man Lieder mit der hier erstellten Software nutzen, fällt also im besten Falle eine Lizenzgebühr an.

Daher werden von Haus aus nur Musikstücke in der Software enthalten sein, die keinerlei Urheberanspruch mehr besitzen, um die Software so kostenlos zur Verfügung stellen zu können.

3.3.4.2 Bezugsquellen neuer Lieder

Die für den Nutzer komfortabelste Lösung ist eine in das Programm integrierte Download-Funktion für neue Musikstücke. Diese sollten, wie auch im herkömmlichen Auswahlmenü der Lieder, gesucht und sortiert bzw. auch direkt dort integriert werden.

Freie Stücke können direkt geladen werden. Stücke, für die eine Lizenz erworben werden müsste, können mittels In-App-Käufe zur Verfügung gestellt werden.

Eine andere Möglichkeit ist, eine Web-Seite zur Verfügung zu stellen, auf der der Nutzer nach neuen Liedern suchen und herunterladen kann. So muss der Nutzer allerdings die heruntergeladenen Lieder selbstständig in den richtigen Ordner des Programmes kopieren. Das könnte die Mehrzahl der Nutzer, besonders auf mobilen Geräten, überfordern, da die gegebenen Ordnerstrukturen auf mobilen Geräten oft sehr komplex sind.

Wer einen bestimmten Titel nicht findet, aber musikalisch etwas begabt ist, kann auch selbst zur Tastatur greifen und sich sein eigenes Stück schreiben. Dafür würden eigens entwickelte Editoren helfen und so dem Entstehungsprozess neuer Werke zugute kommen.

Da der Autor beim Verfassen dieser Arbeit eine kleine Fangemeinde um sich aufgebaut hat, konnten kleinere Editoren und auch eine voll funktionsfähige Web-Seite bereits von Außenstehenden zur Verfügung gestellt werden. Auf dieser lassen sich kleine Testlieder runterladen.

Die Web-Seite ist zu diesem Zeitpunkt (2014-02-24) unter folgender URL erreichbar:

http://spielplatz.site50.net/Songdatenbank/beliebtesteTitel

Sollte dies nicht der Fall sein, findet sich eine entsprechende Verlinkung sicherlich unter

http://www.10000faces.de

3.4 Der Nutzer als Testroutine

3.5.1 Maschinelles Testen

Möchte man eine Software auf ihre Funktionalitäten hin überprüfen und auf ihr richtiges Verhalten hin untersuchen, so sind Testroutinen unverzichtbar. In Codeteilen in denen Eingabewerte hinein und Ausgabewerte herauskommen, die man klar definieren kann, sind sogenannte Unit-Tests das Mittel der Wahl. Dabei werden bestimmte Software Module auf ihr korrektes Verhalten mittels vordefinierter Tests automatisch auf ihre Richtigkeit geprüft.

Für Java bietet sich das Framework JUnit an. In JUnit können Testklassen geschrieben werden, die die Funktionen einer Klasse so durchführen, dass jeder Test unabhängig von den Ergebnissen der vorherigen Tests ist.

Abseits dieser maschinellen Tests gibt es aber noch Teile einer Software, die sich in dieser Art nicht testen lassen. Und das sind Tests, die der Nutzerfreundlichkeit zu Gute kommen. Denn die perfekte Software hat nicht viel Sinn, wenn man sie nicht oder nur umständlich nutzen kann.

3.5.2 Usability Expert Evaluation

Eine Möglichkeit, sein System auf die Benutzerfreundlichkeit testen zu lassen, ist die, eine außenstehende Firma oder Abteilung damit zu beauftragen. Dabei prüfen Experten in mehreren Phasen (einzeln oder im Team) die Software auf ihre Nutzerfreundlichkeit.[64]

Das heißt, die Experten greifen dabei auf ihr eigenes Wissen und Erfahrungen im Umgang mit Software zurück. Das kann bedeuten, dass hierbei bestimmte Studien oder Abstufungssysteme genutzt werden, es

64 http://www.gamasutra.com/view/feature/2333/better_games_through_usability_.php (2014-02-24)

kann aber auch bedeuten, das hier lediglich das Bauchgefühl einiger Wenige über eine Software entscheiden, die eventuell für eine Zielgruppe geschaffen ist, die eine ganz andere Vorstellung von Nutzerfreundlichkeit hat.

Der positive Aspekt dieser Methode ist dahingegen allerdings, dass sie bereits in der konzeptionellen Phase der Softwareentwicklung eingesetzt werden kann. Ein Punkt also, an dem den Experten nur allgemeine Abläufe auf dem Papier dargelegt werden können. So können eventuelle Fehlentscheidungen bereits in der Planungsphase reduziert werden und somit Zeit und Kosten gespart werden. Ein Vorteil bildet die Erfahrung, welche die durchzuführenden Experten haben,denn sie wissen, welche Punkte bei der Benutzerfreundlichkeit beachtet werden müssen. Der normale Nutzer wird vermutlich oft zwar unzufrieden sein, aber nicht wissen warum. Eine mit der Nutzerfreundlichkeitsüberprüfung beauftragte Firma hingegen kann nicht nur fest machen, was nicht optimal ist, sondern auch gleichzeitig sinnvolle Lösungsvorschläge bieten.

3.5.3 Usability Tests mit Google Analytics

Eine kostengünstige Möglichkeit, seine Software auch nachträglich auf ihre Nutzerfreundlichkeit hin zu untersuchen, ist das Einsetzen der Analytic Tools von Google.[65]

Hierbei können vom Nutzer ausgeführte Aktionen online nachverfolgt werden. Zum Beispiel, wie oft er ein neues Lied wählt, ob er die Mikrophonkonfiguration oft neu starten muss oder auch wie lange eine Nutzung des Programmes dauert. Praktisch ist auch das Verfolgen der Menüaufrufe. Somit können Funktionen, die überdurchschnittlich oft gebraucht werden, welche aber durch zu viele Menüverschachtelungen schwer oder umständlich zu erreichen sind, beim nächsten Update der Software leichter zugänglich positioniert werden.

Voraussetzung ist dafür allerdings, dass man den Nutzer darauf

65 http://www.google.com/analytics/ (2014-02-24)

aufmerksam macht, dass man Daten von ihm sammeln möchte. Möchte er dies nicht, ist es ratsam, ihm nicht die Nutzung der Software zu verbieten, sondern lediglich das Aufzeichnen der Daten nicht durchzuführen.

Weiterhin ist es für Deutschland wichtig, die Daten anonymisiert zu aufzuzeichnen, da hier das Datenschutzgesetz greift (§ 11 BDSG). Dafür gibt es von Google eine Erweiterung mit dem Namen "anonymizeIp". Diese löscht vor der Übertragung die letzten acht Ziffern der IP-Adresse des Nutzers.[66]

3.5.4 Usability mittels Feedback von Nutzern

Vor allem in der Spieleindustrie wird die wichtigste Ressource für die Software getestet. Der Mensch. Wie gut sich ein Spiel spielen lässt, erfährt man am besten direkt vom Nutzer selbst. Dabei hängt es von der Entwicklungsphase der Software ab, wer sie testen darf. In sehr frühen Phasen (Alpha-Phase), in der noch nicht mehr als ein Grundgerüst steht und noch nicht alle Funktionen implementiert sind, sind es in der Regel firmeninterne Mitarbeiter, die erste Test an der Spielbarkeit durchführen. Sie wissen, dass es normal ist, dass noch vieles nicht so läuft wie es letztendlich sein soll und können so Kritik äußern, die für diese sehr frühe Entwicklungsphase angemessen ist.

Ist die Software soweit, dass alle Funktionen implementiert sind, spricht man von einer Beta-Phase. Das Spiel ist im Grunde fertig. Was allerdings noch fehlt, ist das Balancing. Das bedeutet, dass das Spiel eventuell noch zu leicht oder zu schwer ist. Weiterhin ist dies die Phase, in der noch Bugs (Programmierfehler und unverhofftes Verhalten) in der Software sind. Nun kann wieder firmenintern getestet werden. Einige Entwickler gehen hier aber bereits auch den Schritt in die Öffentlichkeit. In der sogenannten Closed-Beta wird eine bestimmte Anzahl an Nutzern ausgewählt (Zufällig oder nach eigenen Kriterien) und Zugang zur Software gewährleistet. Dies

66 http://kress.it/2010/07/google-analytics-anonymizeip-ip-adressen-kurzen-richtiger-code/ (2014-02-24) Anleitung zum richtigen Umgang mit anonymizeIp

passiert oft online mittels Beta-Keys, wie zum Beispiel bei dem Spiel Titan Fall[67] In einer darauf folgenden Open-Beta darf jeder die Software nutzen. Dabei werden in der Regel im Hintergrund Nutzerdaten gesammelt, um so letzte Änderungen durchführen zu können. Durch die große Anzahl an Nutzern und eines Kontaktformulars, ist es möglich, viele Bugs die bisher noch gar nicht bekannt waren, zu entdecken. Denn im Gegensatz zum Entwickler, der instinktiv um Fehler herum spielt, ist der Endnutzer unvoreingenommen und löst so Situationen und Ablaufszenarien aus, an die vorher noch keiner gedacht hat.

Dass diese Form des Softwaretests nicht nur dazu da ist, Programmierfehler aufzuspüren, zeigt das Beispiel für das Nutzerinterface des Spieles "The Sims"[68]. Dieses wurde rund 100 Spielern in 11 Iterationszyklen immer wieder so angepasst, dass der Nutzer einen möglichst einfachen Umgang mit dem Spiel hat.[69]

Der Autor dieser Arbeit nutzt ein ähnliches Verfahren. Er betreibt einen Kanal auf YouTube[70], auf dem er, schon in der Zeit bevor diese Arbeit entstanden ist, Videos über die allgemeine Spieleprogrammierung zeigt. In den Videos geht es unter anderem um in der Entwicklung befindlicher Projekte, die dem Zuschauer bereits in dieser Phase zur Verfügung gestellt werden. So bildet sich eine Gemeinschaft von freiwilligen Testern und gleichzeitig Kritikern, die der Zielgruppe entsprechen. Wie bei allen Testern muss man aber immer darauf achten, dass das, was die Nutzer meinen zu wollen, nicht unbedingt das ist, was sie wirklich benötigen. Es gilt also, nicht blind die Kritik zu übernehmen und umzusetzen, sondern auch zu hinterfragen, ob das, was in diesem Verbesserungsvorschlag verlangt wird, wirklich das ist, was im Programm geändert werden muss.

67 http://socialgames-mag.de/titanfall-beta-key-bekommen-fuer-pc-oder-xbox-one/ (2014-02-24)
68 http://www.thesims.com/de-de (2014-02-24)
69 http://ocw.metu.edu.tr/file.php/85/ceit706_2/10/p393-jorgensen.pdf (2014-02-24)
70 http://www.youtube.com/user/Tomzalat (2014-02-24)

4. Spielen nicht lesen!

4.1 Mehr als die Summe ihrer Teile

Diese Arbeit zeigt eine Eigenschaft der Anwendungsentwickling sehr schön. Es reicht nicht aus, programmieren zu können, um etwas zu erschaffen, was den Menschen und damit den Nutzern der Software auch nützlich ist. Die Informatik ist nicht mehr als ein großer und vielseitiger Werkzeugkasten, der es Entwicklern erlaubt, anderen Menschen auf ihren Wegen zu unterstützen. So musste man für dieses kleine Projekt, in dem es darum ging einfache Lieder auf der Mundharmonika zu erlernen auch in viele andere Wissensbereiche vordringen. Es stellte sich heraus, dass es eine große Vielzahl, verschiedener Modelle und eine ebenso große Anzahl an Spieltechniken für die Mundharmonika gibt. Weiterhin stellte sich heraus, dass man sich dafür eben nicht nur mit Musik und deren Notation auseinandersetzen muss, sondern auch mit den physikalischen Gesetzen von Tönen und wie man diese mittels mathematisch komplexen Verfahren filtern und programmatisch analysieren kann. Weiterhin hat man gesehen, dass es nicht ausreicht, dass eine Software funktioniert. Sie muss auch einfach zu bedienen sein und Spaß machen. Dass Spaß ein wichtiger Bestandteil des Lernens ist und das es nicht ausreicht ein mal in der Woche zehn Minuten zu üben, um einen guten Lernerfolg verzeichnen zu können, wurde auch dargelegt. Wer programmiert, weil er programmieren möchte, darf dies natürlich tun. Er soll sich aber nicht wundern, wenn er einem Menschen begegnet, der mit einem Hammer auf einen Amboss schlägt, ohne dabei etwas Konkretes zu formen. Eine Einzelwissenschaft ergibt immer nur dann Sinn, wenn man sie in der Verzahnung mit anderen Wissenschaften nutzen kann um somit etwas Neues zu schaffen.

4.2 Die Früchte dieser Arbeit

Im Rahmen dieser Arbeit ist kein fertiges und komplexes Lernprogramm entstanden. Es war aber auch nicht Schwerpunkt dieser Arbeit, ein komplettes Programm zu entwickeln. Es sollte gezeigt werden, dass es möglich ist, mit genügend Zeit und Wissen eine solche Software zu ermöglichen.

Es gibt allerdings einen kleinen Prototyp, der genau dies aufzeigt. Unter dem Titel "Harpo" lässt sich das Programm im Play Store von Google für Androidgeräte finden.

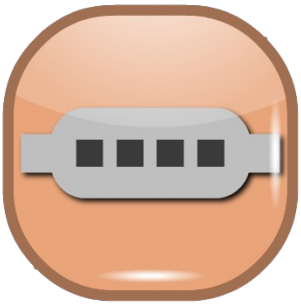

Abbildung 33: Icon der App Harpo

Link zum Play Store:

https://play.google.com/store/apps/details?id=lgd.bleek.tom&hl=de

In absehbarer Zeit soll dieser einfache Prototyp weiterentwickelt werden, da es schade ist, diesen so unfertig liegen zulassen. Denn die Auswahl an ähnlicher Software ist im Grunde nicht vorhanden.

Eine andere Sache, die der Autor mit dieser Arbeit herausgefunden hat, ist, dass Einem fehlendes Wissen gar nicht auffällt und am Anfang alles leichter erscheint als es am Ende dann tatsächlich ist. Und dass fehlendes Wissen aus einer ungenügenden Zeitplanung resultieren kann. Von daher sollte man, bevor man ein Projekt plant, sich ein ausreichend fundiertes Wissen zum dem entsprechenden Themengebiet angeignen.

4.3 Danksagungen

Lassen Sie mich für diesen Teil meiner Arbeit etwas persönlicher sein.

Viele Menschen haben mich bei dieser Arbeit unterstützt. Die einen wissentlich die anderen unwissentlich. Besonderen Dank gebührt xxxxxxxxx xxxxxxx, die diese Arbeit immer wieder auf Grammatik- und Rechtschreibfehler hin überprüft hat. Ohne sie würde dieser Text aussehen wie von einem Sechsjährigen geschrieben. Einen weiteren großen Dank gilt xxxxxx xxxxxxxxxx, die es ausgehalten hat, viele Wochenenden ohne mich verbringen zu müssen, da ich es ihr vorzug diese Arbeit zu verfassen. xxxxxxxx xxxxxxxxxxx danke ich dafür, dass er es geschafft hat, um mich herum für Sauberkeit und humoristischen Abwechslungen zu sorgen.

Prof. Dr. Xxxxxxx xxxxxx und Prof. Dr. Xxxxxxx xxxxx danke ich für ihre Zeit, die sie investiert haben, diese Arbeit zu lesen und sie nach bestem Ermessen zu bewerten.

Zum Schluss möchte ich meinen vielen freiwilligen Zuschauern und Testern danken, die mich seit nun über einem Jahr hinweg online verfolgen und vielleicht sogar diese Arbeit gelesen haben.

Ach ja, ich danke Ihnen für das Lesen dieser Arbeit.

Tom Bleek

5. Anhang

5.1 UML Diagramme

Um auch die Implementierung des Prototypen besser nachvollziehen zu können, sind folgend zwei UML-Klassendiagramme aufgelistet.

Auf dem ersten Klassendiagramm ist der Aufbau der drei Nutzeransichten dargestellt:

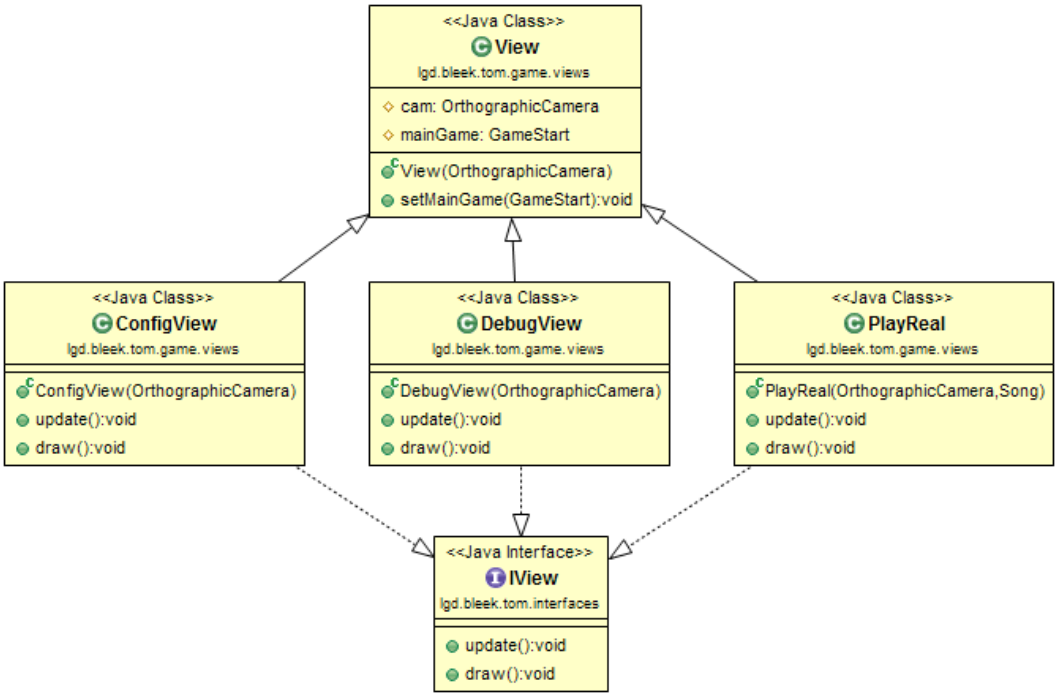

Abbildung 34: Klassendiagramm View-Hierarchie

Aus Gründen der Übersichtlichkeit wurden alle privaten Member und Methoden in der Abbildung außen vor gelassen. Man Sieht hier, dass die drei Klassen, ConfigView, DebugView und PlayReal von der Klasse View erben. Die View Klasse ist dazu zuständig, allgemeine Einstellungen durchzuführen, die den drei Unterklassen gemein sind. Wie das Einstellen der Kamera und das setzen der Haupt-Scenen-Manager-Klasse (GameStart).

Damit die GameStart-Klasse alle View gleich behandeln kann wurden sie

durch das IView Interface ergänzt.

Auf dem nächsten Klassendiagramm wird die Struktur der Audioklassen aufgezeigt. Diese sind dafür zuständig, die auditiven Eingaben des Nutzers auszuwerten und die Ergebnisse zur Verfügung zu stellen.

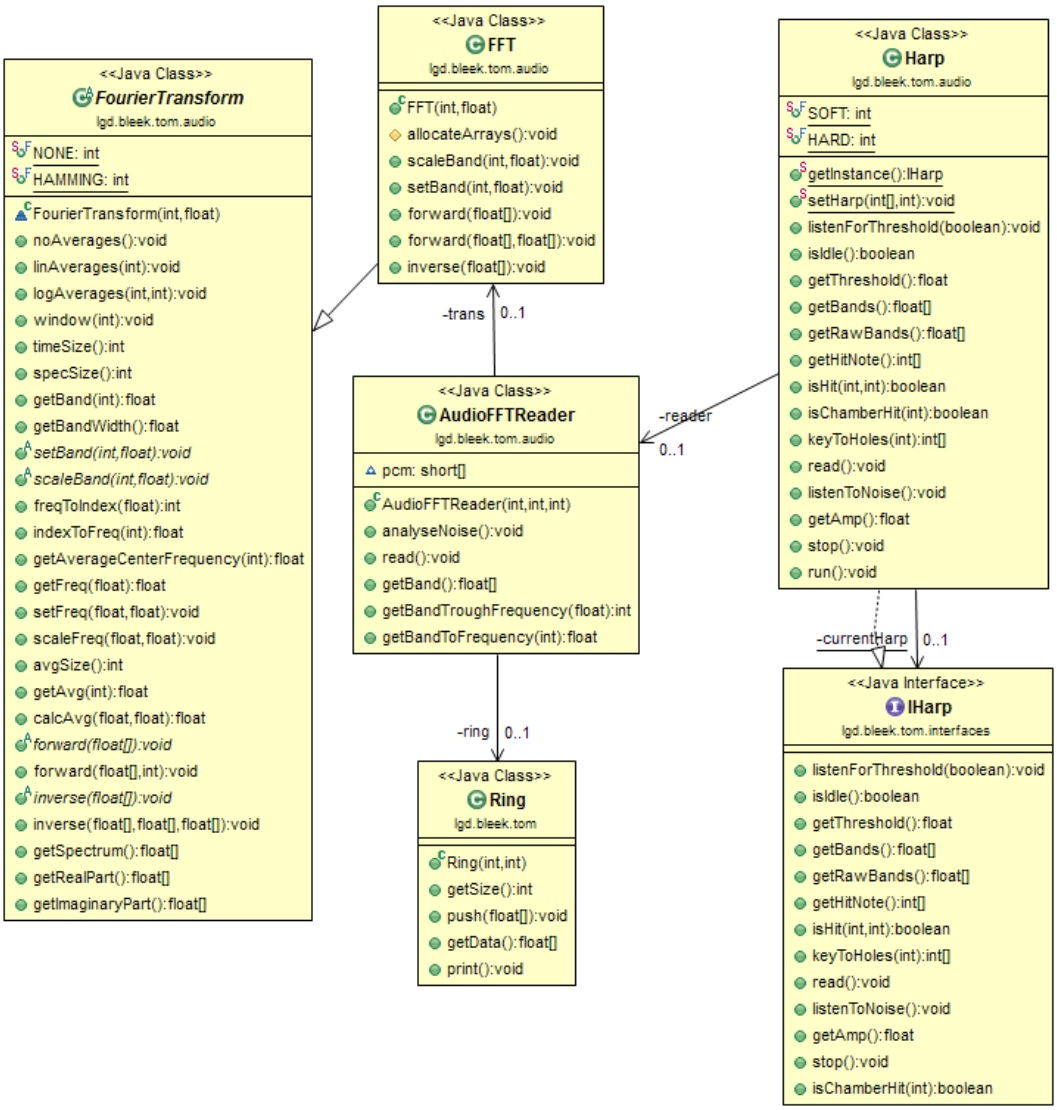

Abbildung 35: Klassendiagramm - Audioanalyse

Die Klassen FourierTransform und FFT sind früherer Bestandteil der libGDX Bibliothek und mussten von Hand aus alten Versionen importiert werden. Diese wurden unverändert genutzt. Die AudioFFTReader-Klasse ist dafür zuständig, den Minkrophoninput aufzunehmen, von Störungen zu beseitigen und mittels der FFT Klasse in das Frequenzspektrum

umzuwandeln, um es so der Harp-Klasse zur weiteren Verarbeitung zur Verfügung stellen zu können. Die Harp-Klasse ist eine Implementierung des IHarp Interfaces welches die wichtigsten Funktionalitäten für die drei Iview-Implementierungen bereitstellt um dem Nutzer ein nutzbares visuelles Feedback gewährleisten zu können.

Im Folgenden ist noch der grundlegende Aufbau des Softwareprototypen beschrieben.

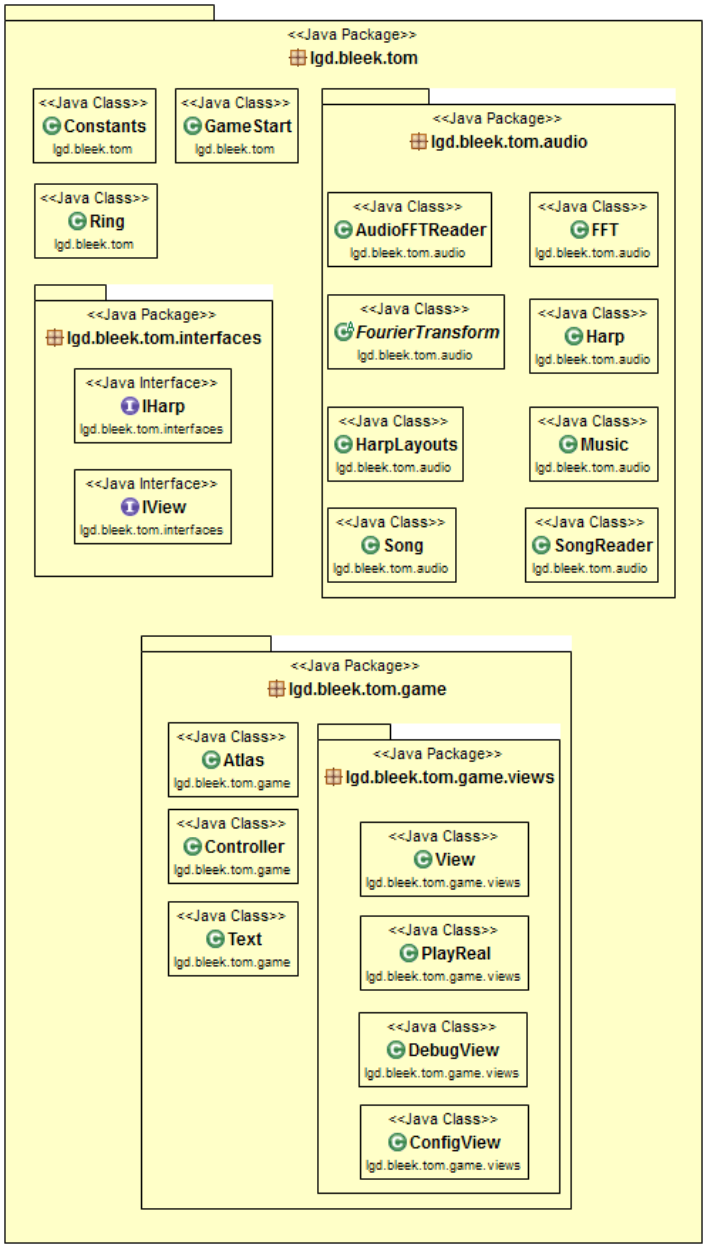

Abbildung 36: Paketdiagramm des Prototypen

5.2 Code Dokumentation (CD-ROM)

An dieser Stelle den gesamten Programmcode darzulegen, ist nicht gerade sehr ökologisch und würde die Arbeit unnötig aufblähen. Daher ist auf der beigelegten CD eine komplette Javadokumentation sowie der Source Code des Prototypen vorhanden.

Weiterhin befindet sich auf dem beiliegendem Datenträger das kompilierte Programm als .jar-Datei in dem Ordner „Spiel“.

Sie als Leser der Onlineversion haben an dieser Stelle leider pech gehabt und gehen leer aus.

6. Quellenverzeichnis

6.1 Offlinequellen

Holdermann, C. / Müller-Beck, H. / Simon U. (2001)
Eiszeitkunst im suddeutsch-schweizerischen Jura. Anfänge der Kunst

Gasser, P. (2010)
Gehirngerecht lernen – Eine Lehranleitung auf neuropsychologischer Grundlage

Spitzer, M. (2006)
Lernen – Gehirnforschung und die Schule des Lebens

Lander, J. / Gregory, J. / Taylor & Francis Ltd (2009)
Game Engine Architecture

6.2 Onlinequellen

6.2.1 Erläuterungen

Folgend werden Quellen aufgelistet welche Online verfügbar sind. Die in "[ ]" eingeschlossene Zahl gibt an in welchem Unterordner eine Kopie der dementsprechenden Web-Seite zu dem Zeitpunkt der Recherche liegt.

6.2.2 Informatik Allgemein

[44] Erläuterung deines Ringspeichers

http://www.embedded.com/electronics-blogs/embedded-round-table/4419407/The-ring-buffer (2014-02-19)

[49] Übersicht der ASCII Zeichen

http://www.ascii-code.com/ (2014-02-23)

6.2.3 Musiklehre und Akustik

[01] Hörschwellengrafik modifiziert verwendet (2014-01-10)

http://www.dasp.uni-wuppertal.de/ars_auditus/psychoak/psychoak4.htm

[02] Duffin, D.W.(2007)
Kleine Zusammenfassung der Geschichte der Gleichstimmung
http://kilchb.de/rein_duffin.html (2014-01-15)

[03] Liste an verschiedenen Stimmarten
http://www.groenewald-berlin.de/Inhaltsverzeichnis.html (2014-01-15)

[04] Rounds, T
Hörbeispiele für unterschiedliche Kammertöne
http://www.tedrounds.com/Can_you_hear_the_difference.htm
(2014-01-16)

[05] Übersicht über Frequenzen der Klaviatur in Gleichstimmung
http://www.sengpielaudio.com/Rechner-notennamen.htm (2014-02-06)

[06] Diverse Umrechner für musikalische Einheiten
http://www.sengpielaudio.com/Rechner-centfrequenz.htm (2014-01-22)

[07] Erklärung von Dur- und Moll-Tonarten
http://www.harmonielehre.info/tonarten-und-tonleiter.html (2014-01-22)

[08] Videobeispiels eines Terpodions
http://commons.wikimedia.org/wiki/File%3ATerpodion2.ogv (2014-01-25)

[12] Tonsysteme
http://kilchb.de/axiomensystem.html (2014-01-31)

[30] Artikel über das menschliche Höhrempfinden
http://dasgehirn.info/wahrnehmen/hoeren/was-hoert-man-eigentlich-128/
(2014-02-07)

[50] Offizielle MusicXML Webpräsenz
http://www.musicxml.com (2014-02-23)

[51] Offizielle ABC Webpräsenz
http://abc.sourceforge.net/ (2014-02-23)

[52] Erklärung der ABC-Notation
http://penzeng.de/Geige/Abc.htm (2014-02-23)

[53] Marx, B. (2003)
Der Online MIDI Guide
http://unseretollepage.de/hosted/midiguide/vorwort.html (2014-02-23)

[54] Video über ein Piano welches die menschliche Stimme mittels MIDI nachahmt
http://www.youtube.com/watch?v=muCPjK4nGY4 (2014-02-23)

6.2.4 Mundharmonika

[09] Eine tabelarische Übersicht über die Geschichte der Mundharmonika
http://www.klausrohwer.de/privat/hobbies/muha/geschi.htm (2014-01-25)

[10] Kurze Geschichte der Mundhamronika in Amerika
http://www.bluessource.de/blues-harp-die-geschichte-der-mundharmonika/ (2014-01-25)

[11] Rohwer, K.(2003)
Aufbau der Mundharmonika
http://www.klausrohwer.de/privat/hobbies/muha/muhapubl/muhakunde.htm (2013-01-22)

[13] Sonderstimmungen der Mundharmonika
http://www.angelfire.com/tx/myquill/SpecialTunings.html (2014-01-31)

[14] Erläuterung der Richterstimmung

http://www.harmonikaverband.at/muha/mundharmonika_richter.htm (2014-02-06)

[15] Rohwer, K.

Schematische Darstellung verschiedener Mundharmonikabauarten

http://www.klausrohwer.de/privat/hobbies/muha/muhapubl/muhakundeabb/abb02_gesamt.png (2014-01-31)

6.2.5 Frequenzanalyse

[31] Überblick über Sinus und Cosinus

http://www.visionlearning.com/en/library/Math-in-Science/62/Wave-Mathematics/131 (2014-02-06)

[32] Kurze erläuterung zu Joseph Fourier

http://www.britannica.com/EBchecked/topic/215097/Joseph-Baron-Fourier (2014-02-06)

[33] Prof. Dr. Speanhauer, A. / Prof. Dr. Steiner-Curtis M.

Einführung in die Theorie der Fourierreihen und Fouriertransormation

http://web.fhnw.ch/personenseiten/marcel.steiner/Unterricht/Datenanalyse/Fourier-Skript-Marcel-Steiner.pdf (2014-02-06)

[34] Erläuterung des Nyquist-Shannon Abtasttheorems

http://www.uni-protokolle.de/Lexikon/Nyquist-Shannon_Abtasttheorem.html (2014-02-08)

[35] Erklärung verschiedner Fehlerquellen der Fourier-Transformation

http://www.uni-giessen.de/~gd1186/F-Prak/node9.html (2014-02-11)

[36] Lülff, J. (2009)

(Fast) Fourier Transformation und ihre Anwendungen

http://pauli.uni-muenster.de/tp/

fileadmin/lehre/NumMethoden/SoSe2009/Vortrag_FFT_220409.pdf (2014-02-11)

[37] Videoerklärung zur FFT

http://youtu.be/EsJGuI7e_ZQ (2014-02-11)

[38] Offizielle FFTW Webpräsenz

http://www.fftw.org/ (2014-02-11)

[45] Im Programm verwendete Klasse zur FFT

https://github.com/libgdx/libgdx/blob/master/extensions/gdx-audio/src/com/badlogic/gdx/audio/analysis/FFT.java (2014-02-22)

6.2.6 Benutzerfreundlichkeit

[55] ISO 9241 Ergonomie der Mensch-System-Interaktion

http://www.handbuch-usability.de/iso-9241.html (2014-02-22)

[57] Artikel über Usability Tests in Videospielen

http://www.gamasutra.com/view/feature/2333/better_games_through_usability_.php (2014-02-24)

[58] Offizielle Google Analytics Webpräsenz

http://www.google.com/analytics/ (2014-02-24)

[59] Anleitung zum richtigen Umgang mit anonymizelp

http://kress.it/2010/07/google-analytics-anonymizeip-ip-adressen-kurzen-richtiger-code/ (2014-02-24)

[62] Joegensen, A. H. (2004)

Marrying HCI/Usability and Computer Games: A Preliminary Look

http://ocw.metu.edu.tr/file.php/85/ceit706_2/10/p393-jorgensen.pdf (2014-02-24)

6.2.7 Lernen

[39] Breuer, J. (2010)

Spielend lernen? Eine Bestandaufnahme zum (Digital) GameBased Learning

http://www.lfm-nrw.de/fileadmin/lfm-nrw/Publikationen-Download/Doku41-Spielend-Lernen.pdf (2014-02-11)

[40] Interview über das Üben mit Sloboda J. (Psychologe)

http://www.zeit.de/2008/02/P-John-Sloboda (2014-02-17)

6.2.8 Videospiele

[16] F.A.Q Seite von Ultrastar

http://dhost.info/ultrastar/faq_allgemein.html (2014-01-22)

[17] Offizielle Singstar Webpräsenz

http://www.singstar.com/de_LU/ (2014-01-22)

[18] Downloadseite für Ultrastar

http://sourceforge.net/projects/ultrastardx/files/ (2014-01-22)

[19] Datenblatt der ersten Singstarversion

http://www.mobygames.com/game/singstar (2014-01-22)

[20] Test des Spieles Singstar (PS3) (2008)

http://www.computerbild.de/artikel/cbs-Tests-PS3-Singstar-2153339.html (2014-01-22)

[21] Guitar Hero Screenshot

http://theherofeed.com/4403/july-guitar-hero-dlc-sale-announced/ (2014-01-22)

[22] Guitar Hero Kontroller

http://upload.wikimedia.org/wikipedia/commons/thumb/8/80/

Guitar_Hero_series_controllers.jpg/
800px-Guitar_Hero_series_controllers.jpg (2014-01-22)

[23] Offizielle Rock Band Webpräsenz
http://www.rockband.com (2014-01-22)

[24] Offizielle Rocksmith Webpräsenz
http://rocksmith.ubi.com (2014-01-22)

[25] Deifinition des Wortes Edutainment
http://wirtschaftslexikon.gabler.de/Definition/edutainment.html (2014-01-22)

[26] Videoreview des Spieles Rocksmith
http://www.youtube.com/watch?v=eGp2p_mAXjo (2014-01-22)

[27] Review des Spieles Rocksmith
http://www.joystiq.com/2011/10/18/rocksmith-review/ (2014-01-22)

[28] Videoreview des Spieles Rocksmith
http://www.youtube.com/watch?v=-aT0C-Etf04 (2014-01-22)

[29] Offizielle Synthesia Webpräsenz
http://www.synthesiagame.com/ (2014-01-22)

[41] Offizielle pixi.jsWebpräsenz
http://www.pixijs.com/ (2014-02-17)

[42] Offizielle RPG Maker Webpräsenz
http://www.rpgmakerweb.com/ (2014-02-17)

[43] Nystrom B. (2014)
Double Buffer
http://gameprogrammingpatterns.com/double-buffer.html (2014-02-22)

[46] Offizielle Starling Webpräsenz
http://gamua.com/starling/ (2014-02-24)

[47] Offizielle libGDX Webpräsenz
http://libgdx.badlogicgames.com/ (2014-02-24)

[48] Offizielle Cocos2D-x Webpräsenz
http://www.cocos2d-x.org/ (2014-02-24)

[56] Halo: Reach Achievements Guid
http://www.gamesradar.com/halo-reach-achievement-guide/ (2014-02-23)

[60] Artikel über Beta-Keys für das Videospiel Titan Fall
http://socialgames-mag.de/titanfall-beta-key-bekommen-fuer-pc-oder-xbox-one/ (2014-02-24)

61
Offizielle The Sims Webpräsenz
http://www.thesims.com/de-de (2014-02-24)

[63] YouTube-Kanal des Autors
http://www.youtube.com/user/Tomzalat (2014-02-24)

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

_____

Tom Bleek

Ort, Datum, Unterschrift