Entwicklung von Methoden und Konzepten für ein spielerisches Trainingsprogramm zur Gehörbildung

Bachelor-Thesis

Lukas Rohde
KOM-type-number

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Entwicklung von Methoden und Konzepten für ein spielerisches Trainingsprogramm zur Gehörbildung
Development of methods and concepts for a game-based ear training program

Bachelor-Thesis
Studiengang: Informatik
KOM-type-number

Eingereicht von Lukas Rohde
Tag der Einreichung: 15. Juli 2021

Gutachter: Prof. Dr.-Ing. Ralf Steinmetz
Betreuer: Dr.-Ing. Stefan Göbel

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Lukas Rohde, die vorliegende Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Darmstadt, den 15. Juli 2021

_____

Lukas Rohde

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Test

1 Einleitung

2 Grundlagen

2.1 Signalverarbeitung

2.1.1 Nyquist-Shannon-Abtasttheorem

Das Nyquist-Shannon-Abtasttheorem besagt, dass eine Frequenz f_{max} nur dann exakt rekonstruiert werden kann, wenn es mit einer mindestens doppelt so hohen Frequenz $2f_{max}$ abgetastet wird. Auf die Akustik bezogen heißt das also, dass man um eine Frequenz von 20kHz ohne Verlust aufzunehmen, man den Ton mit mindestens 40kHz „abtasten“muss. Abtasten heißt hier, den aktuellen Wert bzw. Frequenz zu speichern.

2.1.2 Fouriertransformation

Eine Fouriertransformation ist eine Methode, mit der man ein Signal in ein Spektrum zerlegen kann. Diese Definition ist bewusst weitläufig gewählt, da ich nun nur auf die relevante spezifischere Bedeutung, im Zusammenhang mit Akustik, eingehen werde. Da sich bei einem Ton ein bestimmtes Muster über einen endlichen Zeitraum wiederholt, handelt es sich um ein periodisches Signal. Es wird also eine Fouriertransformation angewandt, welche ein periodisches Signal in ein Linearspektrum zerlegt. Ein Signal zu zerlegen heißt dabei nichts anderes, als mit einer endlichen Summe an Sinusschwingungen zu modellieren. Untersucht man die entstandene Funktion, so kann man die im Signal vorhandenen Frequenzen herausfiltern bzw. erkennen. Diese Eigenschaft ist offensichtlich essenziell bei der Tonerkennung. Es treten Probleme auf, falls eine nicht ganzzahlige Anzahl an Perioden untersucht wird, was so viel heißt, dass eine Periode abgeschnitten wurde. Es kann dann zu “schwammigen” Spektren kommen. Dieses Problem kann durch Fensterfunktionen beseitigt werden. Fensterfunktionen sorgen dafür, dass die Amplituden an den Flanken weich gezeichnet werden, sodass die Endpunkte der Perioden aneinander angeglichen werden und es zu keinen unstetigen Übergängen kommt [But06].

2.2 Musikalische Grundlagen

2.2.1 Intervalle

Ein Intervall im musikalischen Sinn bezeichnet den Tonabstand zwischen zwei Tönen. Man kann dabei zwischen den sogenannten harmonischen Intervallen und den melodischen Intervallen unterscheiden. Bei den harmonischen Intervallen erklingen die Töne gleichzeitig, wohingegen bei melodischen Intervallen die Töne hintereinander erklingen. Intervalle lassen sich mathematisch als Proportionen der erklingenden Töne zueinander beschreiben, das wichtigste Intervall ist hierbei die Oktave. Die Oktave hat ein Frequenzverhältnis von 2:1, sie entspricht also immer der doppelten Frequenz von einem Ton zu dem anderen und kann von mehreren Tonsystemen unterteilt werden. Alle dieser Unterteilungen benennen die Schritte innerhalb einer Oktave nach den lateinischen Ordinalzahlen, wobei Oktave Acht bedeutet. Wie der Name Oktave also schon vermuten lässt, liegen innerhalb einer Oktave sieben andere Intervalle. Diese Annahme ist allerdings nur bedingt richtig, da neben den sieben nach den Ordinalzahlen benannten Intervallen, auch große und kleine, sowie verminderte und übermäßige Intervalle existieren. Diese weitere Einstufung der Intervalle ermöglicht es Intervalle mit einem halben Ton abstand zu notieren, neben den Ganztönen. Berücksichtigt man all das, so lässt sich eine Oktave in 12 Halbtöne aufteilen,

wobei jeder mögliche Abstand zwischen diesen Halbtönen benannt werden kann.

Es gibt neben den Einteilungen von Intervallen innerhalb einer Oktave auch bezeichnungen für Intervalle, welche einen Abstand größer als eine Oktave umspannen. Ich möchte diese aufgrund der Vollständigkeit auch erwähnen, jedoch werden diese nicht weiter in der folgenden Arbeit berücksichtigt. Diese Intervalle folgen weiterhin der lateinischen Ordinalzahlreihe, sodass ein Intervall, welches einen Ganzton weiter umspannt als die Oktave, dementsprechend None genannt wird. [Zie09]

2.2.2 Die Grundfrequenz und Obertöne

Bei allen musikalisch erzeugten Tönen treten neben dem sogenannten Grundton bzw. der Grundfrequenz auch Nebenschwingungen oder Obertöne mit auf. Bei der Grundfrequenz handelt es sich um die tiefste auftretende Schwingung in diesem Spektrum, die restlichen messbaren Schwingungen sind die sogenannten Obertöne, diese bestimmen vorallem die Klangfarbe, verändern aber nicht den erklingenden Ton in seiner Höhe. In der Musik bestimmt also die Grundfrequenz den wahrgenommenen erklingenden Ton und die Obertöne nur die Wahrnehmung des Tons. Die Obertöne können dabei aber häufig einen stärkeren Ausschlag verursachen, untersucht man einen Ton mithilfe einer Fouriertransformation. Ein Ton im musikalischen Sinn beschreibt also nicht eine reine Sinuswelle, sondern vielmehr eine überlagerung mehrerer Schwingungen, welche zusammen den Ton mit Klangfarbe ausmachen. Obertöne sind in der Regel ganzzahlige vielfache des Grundtons, Töne mit dieser Eigenschaft nennt man Harmonisch. Ausnahmen zu dieser Regel sind vorallem Klänge welche für das menschliche Ohr als unschön oder störend empfunden werden, wie beispielsweise Glocken oder fallende Rohre und Stangen, sie werden auch unharmonisch genannt. [Zie09]

2.2.3 Tonsysteme und Stimmung

Wie bereits in 2.2.1 angesprochen, gibt es verschiedene Systeme um die Tonabstände zu bestimmen, ich möchte hier nun die wichtigsten dieser Arten beleuchten und auf ihre Anwendungen und Herkünfte eingehen. Verschiedene Arten von Unterteilungen werden auch Stimmungen genannt. Betrachten wir zunächst den Tonraum, in welchem wir un befinden.
In dem geordneten Tonraum lässt sich jedem Ton genau eine Frequenz zuordnen, wobei gilt, umso höher die Frequenz, desto höher ist auch der erklingende Ton. Außerdem ist jedem Paar von Tönen t_1, t_2 bzw. Frequenzen f_1, f_2 genau ein Intervall i_{12} zugeordnet, wir können daraus folgern, dass das Intervall der Töne, ein Frequenzverhältnis von $f_2 : f_1$ hat. Es wird außerdem in der Musik die Einheit Cent herangezogen um genauere Angaben zu verschieden hohen Tönen zu machen. Ein Cent ist definiert als $1/1200$ Oktave. Betrachtet man den Raum der Intervalle und der Frequenzverhältnisse, so fällt auf, dass ein Homomorphismus zwischen diesen beiden Räumen gegeben ist. Addiert man zwei Intervalle im Intervallraum, so multipliziert man deren Frequenzverhältnisse bzw. Proportionen.
Ein kurzes Beispiel soll diesen Zusammenhang verdeutlichen. Nehmen wir die kleine Terz $i_1 = 316$ Cent und die Quinte $i_2 = 701$ Cent und ihre dazugehörigen Frequenzverhältnisse $p_1 = 6/5$ und $p_2 = 3/2$ als gegeben an.

$$i = i_1 + i_2 = 1017$$

(kleine Septime)

$$p = p_1 * p_2 = 9/5$$

(kleine Septime)

$$=> 1200 * \log_2 9/5 \approx 1017$$

Es gibt nun verschiedene Verhältnisse welche verschiedene sogenannte Stimmungen bilden. Zu den bekanntesten Stimmungen zählen die reine Stimmung, die pythagoreische Stimmung und die gleichstufige Stimmung. All diese Stimmungen haben die Gemeinsamkeit, dass sie durch Cent beschrieben werden und somit alle eine Oktave als 1200 Cent bzw. als ein Stimmungsverhältnis von 1:2 definieren. Die Unterschiede der Stimmungen ergeben sich dann, wie die einzelnen Töne gestuft sind. Die älteste Stimmung aus dem europäischen Raum ist die pythagoreische Stimmung. Pythagoras reihte für die Berechnung zwölf Quinten aneinander und erhielt so sieben Oktaven höher wieder den Ausgangston. Bei dieser Stimmung traten einige ungereimtheiten auf, sodass bspw. das fis einige Cent höher lag als das ges. Dieser Unterschied wird das pythagoreische Komma genannt. Die Töne fis und ges wurden später von der gleichstufigen Stimmung gleichgesetzt. Die gleichstufige Stimmung definiert einen Halbtonschritt als 100 Cent, sodass die ganze Oktave gleichstufig verteilt ist und das pythagoreische Komma aufgelöst wird. Die reine Stimmung basiert, entgegen der anderen beiden Stimmungen, auf der natürlich auftretenden harmonischen Obertöne (2.2.2) eines Tons. Bei der reinen Stimmunge treten noch mehr Probleme mit der Bestimmung der Ganz und Halbtonschritte auf. Es ergeben sich aus der Definition der reinen Stimmung zwei verschiedene Frequenzverhältnisse für einen Ganzton. [Zie09]

3 Analyse

3.1 Analyse von Trainingsprogrammen zur Gehörbildung

3.1.1 State of the Art

Zu den State of the Art Programmen gehören vorallem online Tools zur Unterstützung bei der Gehörbildung. Die folgenden Tools habe ich vorallem durch Befragung von Musikstudenten, sowie durch einfach Suchanfragen gefunden.

Gehörbildungswebsite der staatlichen Hochschule für Musik und darstellende Kunst Mannheim

Hierbei handelt es sich um ein online Tool, bei welchem aus einer festen Anzahl an aufgenommen Hörbeispielen Übungen generiert werden. Dabei werden die Themen Intervall- und Akkord-Intonation, sowie eine Möglichkeit zur Überprüfung der eigenen Fähigkeiten angeboten. Das Intervalllernen ist so strukturiert, dass der Nutzer Töne vorgegeben bekommt und die Intervalle zwischen den Tönen bzw. zu einem Grundton bestimmen soll. Die Website überprüft dabei nicht selbst, ob der Nutzer die Aufgabe richtig absolviert hat, der Nutzer muss sich selbst überprüfen. Es gibt mehrere Arten von Intervalltraining, so kann man seine Fähigkeiten in diatonischen Stufen, also innerhalb einer Tonleiter ohne Halbtöne, oder reines Intervalle erkennen zwischen einzelnen Tönen, verbessern. Es gibt weiterhin die Möglichkeit in ganzen Tonreihen direkte und indirekt erklingende Intervalle zu erhören bzw. Wahrzunehmen. Außerdem bietet die Website vorgefertigte Übungen zu Rhythmus, Melodie, Akkorden, Harmonik und Intonation an. Man erhält zugriff auf die Übungen, nachdem man einen Account erstellt hat. Die Übungen sind dann unterteilt in einzelne Reiter in die einzelnen Lektionen. Es existiert ein kurzer einleitender Text, welcher den Nutzen der Website kurz erläutert und eine schnelle ERklärung, wie vorzugehen ist beim Lernen. Es werden keine Module oder Lektionen angeboten, welche die Grundlagen der Aufgaben näher erläutern. Die Website bietet allerdings nur die Aufgaben an, das heißt es gibt keine eingebaute Überprüfung, weder als Texteingabe noch als Audioinput vom Nutzer. [Man09]

Musictheory.net und Tenuto

Die Website Musictheory bietet eine vielzahl von Übungen in verschiedenen Disziplinen an. Es existieren theoretische Lektionen, welche das fundamentale Wissen herstellen sollen, sowie Übungen, um dieses abzufragen und zu testen. Die Lektionen umfassen die Grundlagen der Musik, wie etwa die verschiedenen Notenschlüssel oder das Notenlesen, bis hin zu verschiedenen Arten von Akkorden und wie diese aufgebaut werden. Die Übungen umfassen dementsprechend ein ähnlichen Wissensumfang, es wird beispielsweise das Notenlesen abgefragt oder die Tonart, aber man kann auch Gehörbildung üben, wo man das Erkennen von Tönen bis hin zu Akkorden üben kann. Bei diesen Übungen kann man allerdings nur das gehörte aus einer List auswählen. Ein Unterschied zu der Website der HfMdK Mannheim besteht darin, dass Musictheory.net Antwortmöglichkeiten gibt und bei diesen ein Feedback gibt, ob die Antwort richtig war. Auch hier gibt es kein Features, welches gesungene Intervalle überprüft. Musictheory.net hat eine App für iOS entwickelt, welche die Aufgaben, welche auch auf der Website zur Verfügung stehen, mit erweiterten Funktionen bereitstellt. Die Funktionen der App umfassen zusätzlich zu den Aufgaben der Website eine Möglichkeit Aufgabetypen anzupassen umso Aufgabentypen spezifisch zu lernen. Desweiteren wurde ein Challenge Modus in der App hinzugefügt, welcher den Nutzer herausfordert unter

Zeitdruck Aufgaben richtig zu beantwortern und den Highscore zu verbessern. Das User Interface der App wird extra beworben auf der Website als einfach zu bedienen und mit einer klaren Benutzer Erfahrung. Die App kostet 3.99$, wohingegen die Website kostenfrei ist und ohne Benutzeraccount verwendet werden kann. Es ist noch anzumerken, dass sowohl die Website, als auch die App nur auf Englisch verfügbar sind, was eine Behinderung für den Nutzer darstellen kann. [mus09]

Teoria

Teoria bietet ähnlich wie Musictheory, einen Theorieteil und einen Praxisteil an. Im Theorieteil werden auch hier die Grundlagen der Musik vermittelt, wie beispielsweise das Notenlesen oder was ein Intervall ist und wie diese aufgebaut werden. Im praktischen Teil ermöglicht Teoria es dem Nutzer seine eigenen Aufgaben selbst zu gestalten, indem man die abzufragenden Intervalle einstellen kann oder auch den Grundton. Es ist außerdem möglich das sogenannte vom Blatt Singen zu trainieren, was nichts anderes heißt als direkt die Noten und den Rhythmus richtig zu singen, ohne es vorher gehört oder geübt zu haben. Das Singen wird allerdings auch hier nicht überprüft, es ist nur ein Angebot gegeben sich selbst überprüfen zu können. Es werden weiterhin Übungsangebote in den Aufgabenbereichen der Intervalle, Akkorde, des Rhythmus und der Tonarterkennung. Die Aufgaben werden durch den Nutzer voreingestellte Parameter zufällig generiert und können somit auch an die Bedürfnisse des Nutzer angepasst werden. Die Website hat ein relativ einfaches und intuitives Design, ist allerdings nur auf Englisch verfügbar, was beispielsweise die Namen der Akkorde betrifft und durchaus eine Umstellung für den Nutzer darstellen kann. [Alv09]

JKG Neigungskurs Musik

Die Website des Justus-Knecht-Gymnasium Bruchsaal bietet ein kleines vordefiniertes Set an Aufgaben, für die Vorbereitung auf die Musik Abiturprüfung, an. Zu den Aufgabentypen zählen Rhythmusdiktate, Melodiediktate, sowie Intervalle und Akkorde erkennen. Wie bereits erwähnt sind die Aufgaben vordefiniert und bieten keinerlei Möglichkeit sie anzupassen nach den Bedürfnissen des Nutzers. Die Aufgaben bilden eine grobe Übersicht über die möglichen Aufgabentypen und dienen nur als Lernhilfe oder zur Überprüfung, sind jedoch nicht geeignet als alleiniges Lernmittel oder zur täglichen Festigung der Fähigkeiten. Es existiert dementsprechend auch keine Schrittweise Einführung in die Grundlagen der Musik oder der Gehörbildung. Desweiteren ist die Bedienbarkeit und Nutzerfreundlichkeit auf der mangelhaft, da die Soundbeispiele über Soundcloud zwar in die Website eingebunden sind, jedoch die Lösungen nur in PDFs zu finden sind, welche zunächst heruntergeladen werden müssen. [Bru21]

Musikgrad

Musikgrad bietet einige simple Aufgabentypen und Trainingsmethoden zu Intervallbestimmung, aber auch zu der Bestimmung von Akkorden und den Grundlagen der Musik, wie etwa das Notenlesen. Die Theorieaufgaben bzw. Lektionen sind dabei strukturiert in Lektionen und Module innerhalb einer Lektion, welche das Thema so Schritt für Schritt näher erläutern. Da einige Aufgaben allerdings leider Flash Player benötigen, sind diese nicht mehr zugänglich. Ich beschränke mich im Folgenden nur auf die verfügbaren Aufgaben, welche allerdings immernoch die wichtigsten Funktionen abdecken. Es wird die Möglichkeit geboten Intervalle zu erhören, dabei werden zwei Töne gleichzeitig abgespielt und der Nutzer muss aus allen Intervallen entscheiden, welches erklungen ist. Es wird dabei die Möglichkeit gegeben, die abzufragenden Intervalle zu konfigurieren, umso auf eigene Schwächen genauer einzugehen. Es gab offensichtlich mal die Möglichkeit auch zu einem gegebenem Grundton ein Intervall zu vervollständigen in einem Notensystem, diese Funktion ist allerdings mittlerweile veraltet und nicht mehr Verfügbar. Auch bei Musikgrad gibt es keine Möglichkeit, dass der Nutzer singen kann und dieser Gesang automatisch überprüft und angezeigt wird. Die Website an sich ist simpel aufgebaut, sodass man durch eine

einfache Auswahl zu den verschiedenen Übungen kommt. Die Übungen sind verfügbar innerhalb einer eingebundenen Anwendung im Browser, sodass kein Vollbildmodus verfügbar ist. [Rie]

Earbeater

Earbeater ist ebenfalls ein Onlinetool, welches mehrere Übungsangebote spannend von Intervallbezeichnung bis hin zu der Identifizierung von Tonleitern. Die Aufgabentypen sind hierbei nochmals in Unterkategorien eingeteilt, welche verschiedene Bereiche der Oberkategorie abdecken. Bei Intervallen handelt es sich hier beispielsweise um verschiedene Kombinationen an Intervallarten, welche aufsteigend oder absteigend abgefragt werden können. Es werden in jeder Aufgabe kurz die relevanten Schlüsselwörter vorher erklärt, sodass der Nutzer das nötige Wissen hat um die Aufgabe zu bearbeiten. Es ist dem Nutzer möglich die Intervalle unter mehreren Auswahlmöglichkeiten zu wählen, jedoch eine andere Eingabe der Lösung ist nicht möglich. Eine Tonerkennung ist also auch hier nicht vorhanden. Die Website bietet die Möglichkeit an ein Benutzerprofil zu erstellen, um die Fortschritte und Highscores der Aufgaben zu speichern. Es ist außerdem möglich eigene Aufgaben zu erstellen und diese auch mit anderen Nutzern zu teilen. Earbeater ist ebenfalls wie Teoria nur auf Englisch verfügbar. Die Website besticht mit einem modernen und übersichtlichem Design, welches sowohl für eine einfache Handhabung sorgt, als auch für eine übersichtliche Darstellung der Ergebnisse der Übungen. Earbeater bietet außerdem eine iOS Anwendung an, sodass die Nutzer auch von Unterwegs üben können. [Vesa]

Tonedear

Tonedear bietet ebenfalls wie die meisten anderen Tool mehrere Aufgabentypen, sowie die Möglichkeit diese zu konfigurieren. Es werden allerdings keine vorgefertigten Kurse angeboten, es gibt also nur zufallsgenerierte Aufgaben, welche keinen aufbauenden Lehrinhalt vermitteln. Dieses Defizit kann Tonedear dafür mit der Möglichkeit der Erstellung von Lehreraccounts ausgleichen. Ein Lehrer kann somit für seine Klasse eigenen Aufgaben erstellen und somit auch einen Lehrplan verfolgen. Zu den Funktionen des Lehreraccounts zählen außerdem die Möglichkeit die Abgaben der Schüler einzusehen und die Klassenliste zu verwalten. Es ist dem Nutzer auch hier die Möglichkeit gegeben die eigenen Aufgaben zu einem gewissen Maß zu konfigurieren, wie etwa die Auswahl aus drei verschiedenen Aufgabensets bei den Intervallaufgaben. Eine Mikrofonunterstützung wird auch von Tonedear nicht angeboten. Die Website ist einfach aufgebaut und bietet eine simple Navigation zu den einzelnen Aufgaben. Das Konfigurieren der Aufgaben ist ebenfalls intuitiv. [Vesb]

Earmaster

Earmaster ist eine professionell Entwickelte Gehörbildungssoftware, welche von dem gleichnamigen Unternehmen entwickelt wird. Earmaster existiert seit den 1990er Jahren und wird seit dem immer weiter entwickelt und deckt somit einen Großteil der Bedürfnisse der Nutzer ab. Die Software kann online für 4$ im Monat gekauft werden und bietet neben Kapitelweise aufgebauten Modulen auch die Möglichkeit benutzerdefinierte Übungen zu generieren. Die Funktionen umfassen eine modulweise Einführung in die Gehörbildung, wobei Themenbereiche von der Tonhöhe bis hin zu Akkord Fortschreitungen abgedeckt werden. Weiterhin bietet Earmaster neben diesem sogenannten Einsteigerkurs auch Workshops, welche das erlangte Wissen aus dem Einsteigerkurs weiter festigen sollen. Diese Workshops gehen dabei auch Modulweise vor und fragen dabei die gewünschten Inhalte gezielter ab. Betrachtet man hierbei beispielsweise den Intervall singen Workshop, so werden dort einzelne Herangehensweisen des Intervallsingens behandelt und abgefragt. Die Software bietet für ihre Zahlreichen verschiedenen Aufgabentypen eine Vielfalt an Arten der Eingabe der Lösungen. Es ist möglich Intervalle selbst zu singen, Töne einzeln auf das Notensystem zu platzieren, sowie aus einer Auswahl an Antworten zu wählen. Die Tonerkennung

ist konfigurierbar, es werden dem Nutzer mehrere Tonlagen zur Auswahl gestellt, sowie die Möglichkeit gegeben seine eigene Tonlage zu definieren. Weiterhin werden nicht nur die tatsächlichen Tonhöhen als richtige Antwort gewertet, sondern auch die jeweiligen Oktaven des gesungenen Tons, sodass der Nutzer immer in der angenehmsten Tonlage singen kann. Außerdem gibt es einige extra Jazz Workshops, welche die besonderen Eigenschaften der Akkorde und Akkordfolgen der Jazzmusik behandeln. Der Nutzer kann außerdem seine bereits vollendeten Lektionen in einer Statistik einsehen, wobei dort auch der Gesamtfortschritt der Übungen eingesehen werden kann. Bei dem Absolvieren der Übungen selbst bekommt der Nutzer direktes Feedback in Form von 5 möglichen zu erreichenden Sternen, welche je nach Genauigkeit und Intonation des Tons vergeben werden. Weitere Gamification Elemente, wie etwa eine das Aufzeichnen einer Übungsstreak, welche angibt wie viele Tage man täglich geübt hat, gibt es nicht. [Jak]

3.1.2 Analyse der relevantesten Anwendungen

Im Folgenden möchte ich die oben genannten Anwendungen zu Gehörbildung unter bestimmten Gesichtspunkten vergleichen und analysieren, um so auf die Stärken und Schwächen dieser schließen zu können. Die wichtigsten Gesichtspunkte sind dabei die Übungsvielfalt sowohl innerhalb eines Aufgabenstyps, sowie das generell Angebot der Aufgabentypen, ob die Übungen erläutert werden und wie gut diese Erläuterungen sind und außerdem die Nutzbarkeit der Anwendung. Unter die Nutzbarkeit fallen außerdem Punkte wie etwa die Übersichtlichkeit der Anwendung, die Vielfalt der Eingabemethoden und inwiefern dem Nutzer Feedback zu der Aufgabe gegeben wird. Weitere Aspekte sind Anreize kontinuierlich zu Üben, sowie, insofern denn vorhanden, die Genauigkeit der Tonerkennung. Ich erhoffe mir mit den Ergebnissen dieser Analyse auf die wichtigsten Merkmale einer Gehörbildungsanwendung schließen zu können und wie man eine solche optimieren kann.

Alle Anwendungen haben gemeinsam, dass sie die mehrere Aufgabentypen zu Intervallen, Akkorden und Tonhöhe abfragen können. Diese Aufgabenbereiche sind zentrales Thema der Gehörbildung und dementsprechend wichtig und vielfältig umgesetzt. Die Umsetzung der Aufgaben jedoch weicht stark unter den bekanntesten Programmen ab. So ermöglichen die Webseiten der HfMdK Mannheim und des JKG Neigungskurses nur das beantworten von vordefinierten Aufgaben, wohingegen der Rest entweder zufällig generierte Aufgaben abfragt oder anpassbare Aufgaben generiert. Einer der Unterschiede in diesem Zusammenhang ist allerdings auch, dass vordefinierte Aufgaben aufeinander aufbauen können oder eine gewisse interne Lernstruktur verfolgen können, welche beispielsweise stetig den Schwierigkeitsgrad erhöht. Weiterhin können vordefinierte Aufgaben den Bezug zur echten Musik besser herstellen, in fällen wie etwa der Häufigkeit von verschiedenen Akkorden in der Musik. Allerdings lässt sich durch zufällig generierte Aufgaben eine größere Vielfalt an Aufgaben gewährleisten, was den Nutzer mehr dazu anregt täglich zu üben. Ein idealer Ansatz wäre hier also einige vordefinierte, bestenfalls von professionellen Musiklehrern entwickelte, Aufgaben zur Verfügung zu stellen, aber auch die Möglichkeit zu bieten, durch konfigurierbare Endlosübungen, dieses Wissen zu festigen, wie es Earmaster oder Earbeater anbieten. Im Zusammenhang mit der Übungsvielfalt steht natürlich auch die Abdeckung der Teilbereiche der Gehörbildung. Bestenfalls bietet eine Andwendung einen möglichst kompletten Überblick über alle relevanten Themen und bietet zu diesen auch Übungen an. In der Realität setzen das auch die meisten Übungsangebote durch. So bietet die Website des Musik Neigungskurses der JKG Bruchsaal zwar nur vordefinierte Aufgaben an, dafür aber in dem Großteil der relevanten Themengebiete. Tatsächlich bietet jedes der oben genannten Trainingsprogramme die Möglichkeit eine Vielzahl an verschiedenen Übungsarten zu absolvieren.
Soll das Trainingsprogramm allerdings nicht nur für bereits ausgebildete Musik zugänglich sein, so benötigt dieses auch theoretische Teile, welche mit den praktischen Übungen Hand in Hand gehen, um neuen Musikern diese Konzepte möglichst verständlich zu vermitteln. Diese Einführung in Themen setzen alle der oben genannten Programme um, indem sie Zugriff auf theoretische Sachtexte bieten, welche

die Musiktheorie hinter den Konzepten erklären oder die theoretischen Teile zusammen mit praktischen Übungen erklären, wie etwa Earmaster. Jedoch hat der Großteil der Trainingsprogramme nur ein rudimentäres Angebot der theoretischen Weiterbildung, was meiner Meinung nach eins der größten Probleme mit diesen Trainingsprogrammen ist. Die theoretischen Übungen bzw. Texte können nicht nur einem Anfänger hilfreich sein, sich neues Wissen anzueignen, sondern auch erfahrenen Musikern, welche vielleicht nur eine Auffrischung der theoretischen Inhalte benötigen. Ein weiterer großer Kritikpunkt an den oben genannten Programmen ist, dass nur minimal Elemente der Gamification implementiert wurden. So implementiert Earmaster ein Feedback System, welches dem Nutzer, nach einer abgeschlossenen Aufgabe, zwischen einem und fünf Sternen verteilt. Die verteilten Sterne werden allerdings nicht in einer Statistik festgehalten, oder gar in einem online Leaderboard mit Freunden oder anderen Nutzern verglichen. Die restlichen Anwendungen setzen in dieser Hinsicht gar kein Gamification Element um. Ich sehe hier bei allen Anwendungen großes verschenktes Potential und denke ein solches System, würde dem Nutzer einen viel größeren Reiz schenken sich weiter bilden zu wollen. Tatsächlich wird bei allen Anwendungen die intrinsische Motivation des Nutzer vorrausgesetzt, dass dieser sich fortbilden möchte. Wichtig ist außerdem die Bedienbarkeit der Anwendungen. Die meisten Anwendungen ermöglichen es dem Nutzer mithilfe von mehreren Button unter einem Notensystem, welches die Aufgabe anzeigt. Dieses Design ist einfach und übersichtlich, jedoch reizt es meiner Meinung nach nicht sämtliche Möglichkeiten der Wissensvermittlung aus. Dem Nutzer kann zusätzlich zu dem Ablesen aus dem Notensystem und dem anschließenden Beantworten der Frage durch einen Knopfdruck, das Verwenden und Lesen eines Notensystem weiterhin beigebracht werden, indem man beispielsweise eine Eingabe durch das klicken in ein Notensystem ermöglicht. Diese Art der Eingabe wird von manchen der Anwendungen umgesetzt, wie etwa Musictheory oder Earmaster.
Eine weitere Eingabemethode, im Zusammenhang mit Noten, ist das erkennen von gesungenen oder anders erzeugten Tönen vom Nutzer. Eine solche echtzeit Erkennung von erklingenden Noten setzt keins der oben genannten Programme um, bis auf Earmaster. Earmaster ermöglicht es 'blind' zu singen, das heißt, ohne dass der Nutzer seinen aktuell gesungenen Ton überprüfen kann. Hat der Nutzer die Note lang genug gehalten, so wird Feedback gegeben mithilfe einer Linie durch das Notensystem, welche Töne der Nutzer alle gesungen hat. Ich halte dieses Feature als eines der wichtigsten in allen oben genannten Anwendungen, da es bei dem Nutzer nicht nur Wissen abfragt, sondern auch aktiv die musikalischen Fähigkeiten trainiert. Eine weitere gute Eingabemethode, welche von einigen der genannten Anwendungen umgesetzt wird, ist die Eingabe der Töne mit einer virtuellen Klaviatur. Diese Art der Eingabe macht vorallem für solche Nutzer Sinn, welche bereits vertraut mit dem Klavier sind und so die Verbindung von Gelerntem direkt auf ihr Instrument übertragen können. Für Anfänger ist diese Eingabemethode offensichtlich nicht geeignet, da diese häufig nicht über das Wissen verfügen, wo welche Töne auf einer Klaviatur sind.

Es lässt sich zusammenfassend sagen, dass viele der Anwendungen offenbar als simple Unterstützungen beim Lernen gedacht sind. Sie sind dementsprechend nicht für die breite Öffentlichkeit entwickelt worden, sondern zum Großteil für Musikstudenten oder Schüler mit einem Leistungsfach Musik. Die einzige Ausnahme bildet augenscheinlich Earmaster, wo man argumentieren kann, dass die Anwendung so umfangreich ist mit so vielen verschiedenen Modulen zum Lernen und einer sehr großen Anpassungsmöglichkeit an die einzelnen Bedürfnisse der Nutzer. Sei es die Möglichkeit an den Computer angeschlossene MIDI Instrumente zu erkennen und zur Eingabe zur verwenden oder den Tonraum an die eigenen Bedürfnisse anzupassen.

3.2 Analyse von Methoden u. Konzepten zur echtzeit Erkennung und Verarbeitung von Audio Input vom Anwender mit Unity

3.2.1 Recherche

Ich bin bei der Recherche schrittweise vorgegangen und habe zunächst generell nach der Tonverarbeitung in Unity recherchiert. Die erste Quelle für diese Informationen ist in der Regel die Dokumentation, dementsprechend habe ich mir die Unity Dokumentation angesehen und nach allem gesucht was Audio, Sound oder Frequency erwähnt. Dabei bin ich zunächst auf die allgemeine Informationen Seite von Unity zu Audio gestoßen [Tec09a], diese leitete mich weiter zu detaillierteren Artikeln zu Audio Verarbeitung in Unity. Zu diesen Artikeln gehörten Audio Source, Audio Listener, Audio Mixer, Audio Effects und Reverb Zones, wobei sich die Reverb Zone nach kürzerer Recherche als irrelevant herausstellte. Die restlichen Artikel waren hingegen sehr von Nutzen für mein weiteres Vorgehen. Die wichtigste Information habe ich aus der Skripting Dokumentation der Audio Source erhalten. Die Audio Source besitzt in der Skripting API eine statische Methode, welche es mir ermöglicht auf die Spektrum Daten des dazugehörigen AudioClips zuzugreifen. Bevor ich mich allerdings mit Spektrum Daten befassen konnte, musste ich erst herausfinden, wie ich Audio Daten vom Nutzer einlese, mein nächster Schritt war also nach einer Mikrofon Klasse in der Unity Dokumentation zu suchen. In der Unity Dokumentation habe ich die Microphone Klasse gefunden, über diese fand ich heraus, dass es möglich ist die Position der aktuell ausgelesenen Daten zu erhalten. Mithilfe der Position der Daten liese sich also eine Art Ringbuffer implementieren, indem man die aktuelle Position des Mikrofons als Pointer auf einen Ringbuffer zeigen liese. Mein nächster Gedanke war daraufhin, einen eigenen Ringbuffer zu implementieren. Ich habe mir also Gedanken gemacht, welche weiteren Funktionen ich benötigen würde, wenn ich diesen Ansatz weiter verfolgen würde. Da ich über diesen Ansatz nur die Audiorohdaten speichern könnte, müsste ich außerdem eine geeignete Schnelle Fourier Transformation implementieren bzw. finden. Ich stellte diesen Ansatz erstmal hinten an, um zunächst Ansätze zu erforschen, welche näher an der Unity Engine liegen und Diese mehr ausnutzen. Ich recherchierte also weiter in der Unity Dokumentation und fand nach kurzer Zeit über das Schlagwort Audio Source die Funktion GetSpectrumData. Die Funktion ermöglicht es direkt die Audiodaten einer AudioSource mithilfe einer schnellen Fouriertransformation zu erhalten. Ich habe mich nun also mehr mit den mir zur Verfügung stehenden schnellen Fouriertransformationen von Unity befasst. Zu den von Unity bereitgestellten Fensterfunktionen zählen Rectangular, Triangle, Hamming, Hanning, Blackman und BlackmanHarris. Ich gehe genauer auf die Fouriertransformation und die Fensterfunktionen in einer späteren Sektion ein.
Ich testete nun mit der Fouriertransformation einige Ansätze aus, ob und wie gut sich aus einer einfachen Fouriertransformation der erklingende Ton herausfiltern lässt. Ein naiver Ansatz war zunächst das absolute Maximum der Fouriertransformation zu wählen und diesen als erkannten Ton weiter zu verarbeiten. Dieser Ansatz funktionierte zunächst sehr gut mit Sinuswellen förmigen Tönen. Als ich anfing die Erkennung mit meiner eigenen Stimme zu testen, wurde mir jedoch schnell klar, dass die Obertöne (2.2.2) der menschlichen Stimme einen komplexeren Ansatz der Tonerkennung benötigen würde. Ich setzte mich als nächstes also mit einigen Pitch Detection Algorithmen (PDA) bzw. Ansätzen auseinander. Bei der Analyse der PDAs musste ich vorallem zwei Punkte berücksichtigen. Einerseits war es wichtig, dass der PDA eine gute Erkennungsrate haben würde, sodass ich auf Basis dessen ein Trainingsprogramm aufbauen kann. In dem Zusammenhang des Trainingsprogramms war es daher offensichtlich auch wichtig, dass der PDA nicht zu viel Rechenleistung verbrauchen würde. Weitere Aspekte bei der Auswahl eines Algorithmus waren, wie umsetzbar dieser mit den von Unity gegebenen Ressourcen ist. Ich notierte mir als besonders Interessante Ansätze die drei Pitch Detection Algorithmen Power Cepstrum [NK03], einem auf Fast Direct Transform aufbauendem Algorithmus (FDT Algorithmus) [YMFA05] und Zero-Crossing [AVF08].

Ein größeres Problem, welches aufkam während den ersten Tests mit einem simplen PDA war, dass selbst ein sehr rechenarmer Algorithmus nicht in echtzeit lief. Mein erster Gedanke hierzu war, dass das Aufnehmen und Bereitstellen der Daten zu Problemen führt. Ich wusste glücklicherweise dass in den Projekteinstellungen von Unity die Audioqualität angepasst werden kann unter Verlust der Qualität [**?**].

Ich änderte die Einstellung also auf beste Performance und nachdem die Qualität sich kaum veränderte, jedoch sich die Performance dramatisch verbesserte behielt ich diese Einstellung bei.

3.2.2 Analyse der Ansätze zur echtzeit Audio Verarbeitung und Entwurf eines echtzeit Algorithmus

Wie bereits in der Recherche erwähnt gab es mehrere Ansätze, um echtzeit Audio Daten vom Nutzer zu lesen. Ich möchte in dem folgenden Abschnitt meine Handlungsweise und Entscheidung erläutern. Die Analyse der Ansätze zur echtzeit Audio Verarbeitung beinhaltet nicht nur die auftretenden Probleme des Einlesens der Daten des Mikrofons, sondern auch die Verarbeitung dieser um die musikalische Note bestimmen zu können. Da das Einlesen und Speichern der Daten einen direkten Einfluss auf die möglichen weiteren Rechenschritte nimmt, lassen sich diese Punkte nur schwer getrennt von einander analysieren. Ich werde daher auf die Aspekte der Architektur und der Erkennung zusammen eingehen umso meinen Denkprozess nachvollziehbar zu erläutern.
Doch zunächst muss ich auf den Begriff “Echtzeit” eingehen und was Audio Verarbeitung in Echtzeit bedeutet. Echtzeit beschreibt einen Zeitabschnitt, in welchem ein Mensch keine störende Verzögerung wahrnimmt. Diese Verzögerung liegt bei der Audiowahrnehmung bei maximal 20ms bis 30ms, aber bestenfalls natürlich so niedrig wie möglich. [LK04]
Die interessantesten Ansätze der Datenarchitektur waren die Implementierung eines Ringbuffers um die eingelesenen Frequenzdaten zu speichern und das einmalige allozieren einer festen Speichergröße in der Art eines Arrays, welches mit jedem neuen Frame überschrieben wird. Ich möchte kurz auf die Entscheidungsfindung bei dieser Auswahl eingehen. Ein großer Vorteil eines Ringbuffers ist es, auf alte Daten und neue Daten gleichzeitig zugreifen zu können, da die neuen Daten nur schrittweise die Alten überschreiben. Die Idee hinter diesem Ansatz war, dass man nicht auf einen gesamten Datenblock von Unity warten muss, sondern über die Funktionen GetData und GetPosition [Tec09b] auch nur Teilstücke des Audioinputs von Unity einlesen kann und somit eine bessere echtzeit Verarbeitung erreichen kann. Da der Ansatz jedoch auch Funktion GetData benötigte, testete ich zunächst einen einfacheren Ansatz der echtzeit Toneingabe über das Mikrofon, indem ich direkt auf die Spektrum Daten des von Mikrofon erzeugten Audioclips zugriff. Nach einer Performance Analyse mithilfe des von Unity zur verfügung gestellten Profilers ergab sich ein sehr positives Bild der Funktion GetSpectrumData. Das Berechnen der FFT und anschließende Schreiben in ein Array verbrauchte auf einem MacBook Pro von 2011 gerade mal 11% des gesamten CPU Verbrauchs der Anwendung innerhalb eines Frames und verursachte eine durchschnittliche Rechenzeiterhöhung von 1.5ms. Eine Verzögerung von 1.5ms durch die eigentliche Be-

Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
EditorLoop	67.1%	67.1%	1	0 B	12.20	12.20
▼ PlayerLoop	31.2%	0.3%	1	8.8 KB	5.68	0.06
▼ FixedUpdate.ScriptRunBehaviourFixedUpdate	12.2%	0.0%	1	3.6 KB	2.22	0.00
▼ FixedBehaviourUpdate	12.2%	0.0%	1	3.6 KB	2.22	0.01
▼ Assembly-CSharp.dll!::FrequencyHandler.FixedUpdate()	11.8%	0.0%	1	3.6 KB	2.14	0.00
▼ FrequencyHandler.ComputeFrequency()	11.8%	0.2%	1	3.6 KB	2.14	0.04
▼ AudioSource.GetSpectrumData()	8.5%	0.0%	1	0 B	1.54	0.00
AudioSource.GetSpectrumDataHelper()	8.4%	8.4%	1	0 B	1.54	1.54

Abbildung 3.1: Profileranalyse für GetSpectrumData

rechnung erzeugt keine Verzögerung, welche nichtmehr als echtzeit definiert werden kann, weshalb ich mich von hier an auf diesen Ansatz konzentrierte und weiter an der Tonerkennung arbeitete. Das Ziel eines Pitch Detection Algorithmus ist es, die Grundfrequenz (2.2.2) eines Tons zu bestimmen. Da die Grundfrequenz nicht immer die stärkste Frequenz im Spektrum des Tons ist, stellt diese Aufgabe eine besondere Herausforderung dar. Es ist Ziel die Grundfrequenz möglichst genau zu bestimmen, ohne eine große Menge an Rechenleistung zu verbrauchen.
Da das Ziel war, den Algorithmus im Rahmen der Möglichkeiten von Unity zu entwickeln, entschied ich mich dazu einen eigenen Algorithmus zu konzipieren, welcher vorallem die Funktion GetSpectrumData von Unity so gut wie möglich einbindet. Der erste Schritt ist immer, zunächst die Daten vom Mikrofon

mithilfe von GetSpectrumData als Spektrum zu erhalten. Die Funktion GetSpectrumData benötigt drei Parameter, wovon zwei direkten Einfluss auf den Algorithmus nehmen. Der erste wichtige Parameter ist ein Array, welches mit den Daten der FFT gefüllt werden soll. Die Länge des Array muss hierbei eine Potenz von 2 sein und darf die Länge von 8192 Feldern nicht überschreiten. Diese Limitierung führt auf die FFT zurück, welche eine Fensterfunktion anwendet um mögliche Informationsverluste zu verhinder, welche durch eine falsch "abgeschnittene" Periode zustande kommen. Eine Datensatzlänge in der Größe einer Potenz von 2 soll dieses Problem verhindern (2.1.2). So kommen wir auch direkt zu dem zweiten Parameter von GetSpectrumData nämlich der anzuwendenden Fensterfunktion. Es werden als Fensterfunktionen bereitgestellt: Rectangular, Triangle, Hamming, Hanning, Blackman und BlackmanHarris. Diese unterscheiden sich in der Komplexität und dadurch auch in der Genauigkeit. In dem obigen Ausschnitt des Profilers sieht man GetSpectrumData mit der maximalen Arraygröße und der komplexesten Fensterfunktion. Die Entscheidung liegt hier also auf der besten Qualität der Berechnung, da selbst diese Rechenzeit noch weit im Rahmen von <= 30ms (3.1) liegt [LK04]. Jetzt wo sich also das Spektrum relativ genau und schnell berechnen lässt gilt es noch den Rest der Tonerkennung darauf aufzubauen. Für die Konzeption des Algorithmus habe ich einige Ansätze aus den oben genannten Algorithmen analysiert und auf meine Verwendung angepasst. Ich habe zunächst das Spektrum in Obertöne bzw. den Grundton und Rauschen aufgeteilt. Die Obertöne bleiben dabei unverändert und das Rauschen setze ich auf null mithilfe eines einfachen Filters. Der Filter berechnet den Durchschnitt über das gesamte Spektrum und setzt alle Werte, die niedriger sind als dieser auf null. Da der Grundton ein relatives Maximum innerhalb des Spektrums ist, sollte dieser in den meisten Fällen nicht unter den Durchschnitt fallen. Der Grundton einer menschlichen Stimme schwingt bei jedem Menschen verschieden stark. Das ist vorallem der Fall bei tiefen, rauen Stimmen, weshalb ein Parameter eingeführt werden muss, welcher die Schwelle des Tonfilters regulieren kann. Einen ähnlichen Ansatz verfolgt der FDT Algorithmus, welcher ein sogenanntes "Ruggedness spectrum" aus Tälern und Bergen erzeugt, indem für jeden Datenpunkt eine Schwelle berechnet wird und daran entschieden wird, ob dieser in einem Tal oder auf einem Berg liegt. Die Schwelle wird nicht als Durchschnitt über den gesamten Datensatz, sondern nur über einen Ausschnitt rund um den zu untersuchenden Datenpunkt berechnet. Der Ansatz des FDT Algorithmus verbraucht dafür mehr Rechenzeit, denn es muss über jeden Datenpunkt iteriert werden und dann jeweils nocheinmal über die umliegenden Datensätze. Der hier vorgestelle Ansatz hingegen berechnet eine Konstante für alle Datenpunkte, indem einmal über alle Punkte iteriert wird und anschließend jeder Punkt entsprechen angepasst wird.

Die Datenpunkte sind jetzt gefiltert und müssen nur noch abgetastet werden um das letzte relative Maximum zu finden. Der Algorithmus geht zunächst davon aus, dass das absolute Maximum auch die Grundfrequenz ist, es werden zur Überprüfung alle früheren Datenpunkte untersucht, ob noch ein relatives Maximum auftritt. Existiert ein Maximum mit einem niedrigeren Index, so wird dieser Punkt als neue Grundfrequenz festgehalten. Um falsche Ausschläge trotz des Rauschfilters zu verhindern, darf die mögliche Grundfrequenz nur um einen bestimmten Prozentsatz vom absoluten Maximum abweichen. Dieses Tool wird dem Nutzer zur Anpassung an die Hand gelegt, falls der Grundton kaum von dem Rauschen abweicht. In diesem Fall ist der Rauschfilter zu extrem, da dieser den Grundton auch filtern würde. In diesem Fall bietet sich eine nicht so radikale Lösung an, in Form des prozentualen Abstands zum absoluten Maximum. Man könnte nun argumentieren, dass der prozentuale Filter für jeden Ton anders angepasst werden müsse, da sich das Spektrum jedes mal verändert. Die menschliche Stimme erzeugt allerdings auf verschiedenen Tonhöhen größtenteils die gleichen Anteile an Obertönen, sodass sich dieser prozentuale Unterschied kaum verändert. Über diese Eigenschaft lässt sich auch die zu einem großen Teil immer gleich klingende Stimmfarbe eines Menschen erklären, da die Obertöne diese ausmachen. Die einzige Ausnahme hierbei ist, wenn der Mensch aktiv versucht die Stimmfarbe zu verändern.

Aktuell kann der Algorithmus nur den Index des gefundenen Grundtons zurückgeben, ich brauche allerdings die Frequenz. Um die Frequenz aus dem Index zu berechnen, benötige ich einen Wert, welcher angibt, wie viel Hz pro Schritt im Array gegangen werden. Dieser Koeffizient lässt sich berechnen aus der Samplerate des Mikrofons geteilt durch zwei, umso die maximale abtastbare Frequenz zu erhalten

(2.1.1). Dieser Wert muss noch verteilt auf alle Werte des Spektrums betrachtet werden. Es sei s_{mic} die Samplerate des Mikrofons und $\bar{x}$ die Anzahl der Werte des Spektrums, so können wir den Koeffizienten F_c wie folgt berechnen.

$$F_c = \frac{s_{mic}}{2\bar{x}}$$

Es ist einfach von dem Index auf die Frequenz zu schließen, indem der Index mit dem Koeffizienten multipliziert wird. Die entstandene Abstufung der Frequenzen pro Index ist sehr ungenau, weshalb zuletzt noch zwischen den höchsten Werten interpoliert werden muss, um das tatsächliche Maximum annähernd bestimmen zu können und die Frequenz so genau wie möglich angegeben werden kann. Hierfür wende ich eine einfache Newton Interpolation auf die drei Punkte um den maximalen Wert des relativen Maximums. Die Rechenleistung wird so reduziert, da das Maximum innerhalb dieser drei Werte liegen muss.

Algorithm 1 Pitch Detection Algorithm

Require: filter, threshold, cutoff

```
1: function ComputeFrequency
2:     spectrum=GetSpectrumData()
3:     spectrumCut = first values of spectrum till cutoff Value
4:     absMaxIndex = 0
5:     max = maximum of spectrumCut
6:     avg = average of spectrumCut
7:     for i = 1 to length of spectrumCut −1 do
8:         if spectrumCut[i] < avg ∗ filter then
9:             spectrum[i] = 0
10:            continue to next iteration
11:        end if
12:        if spectrumCut = max then
13:            absMaxIndex = i
14:        end if
15:    end for
16:    bestIndex = HandleOvertones()
17:    interpolatedIndex = Interpolate(bestIndex)
18:    if interpolatedIndex != 0 then
19:        return interpolatedIndex ∗ F_c
20:    end if
21:    return bestIndex ∗ F_c
22: end function
23:
24: function HandleOvertones(absMaxIndex)
25:    overtoneMax = 0
26:    overtoneMaxIndex = absMaxIndex
27:    for i = absMaxIndex to 0 do
28:        currentVal = spectrumCut
29:        if currentVal = 0 then
30:            overtoneMax = 0
31:        end if
32:        if currentVal > spectrumCut ∗ threshold then
33:            if overtoneMax < currentVal then
34:                overtoneMax = currentVal
35:                overtoneMaxIndex = i
36:            end if
37:        end if
38:    end for
39:    return overtoneMaxIndex
40: end function
```

4 Konzeption eines spielerischen Trainingsprogramms zur Gehörbildung

Im folgenden Kapitel möchte ich auf die Konzeption einer spielerischen Trainingssoftware zur Gehörbildung eingehen. Unter die Konzeption fällt dabei, wie die Anwendung aufgebaut sein soll, wie die Unity Engine auf Systemsicht mit den Eingaben umgehen soll und schließlich das Erarbeiten eines Evaluationskonzept.

4.1 Aufbau für so ein Programm

Ich werde nun auf die grobe Struktur der Anwendung eingehen aus Sicht des Nutzers beschrieben. Es werden nun also nur von dem Nutzer sichtbare Designentscheidungen diskutiert. Auf die genaueren Inhalte der einzelnen Szenen innerhalb des Spiels, werde ich ebenfalls eingehen.

Ergebnisse der Befragung von Studierenden

4.2 Systemsicht: I/O Verarbeitung, Gui wie wird interagiert

4.3 Evaluationskonzept

Bei der Evaluation werde ich vorallem drei Gruppen meine Anwendung testen lassen und Sie dazu befragen. Diese drei Gruppen sind Hobbymusiker, professionelle Musiker und Musikstudenten. Dabei habe ich an jede Gruppe andere Ansprüche und möchte auf die individuellen Bedürfnisse und Umstände dieser eingehen in der Evaluation. Daraus erhoffe ich mir ein möglichst komplettes Bild zu der Umsetzung der Trainingssoftware machen zu können und somit auch ein akkurateres Fazit ziehen zu können. Ich habe die genannten Gruppen ausgewählt, da diese einen großen Teil der Musiker mit dem Blick auf das Wissen abdecken, aber auch unterschiedliche Instrumente und Musikarten abdecken können. Gerade die Studierenden sind interessant in Bezug auf die Lernwirkung, da diese im Studium das Modul Gehörbildung absolvieren müssen und sie daher einen besseren Bezug zu anderen Lernmethoden haben.
Ich möchte bei Hobbymusikern vorallem auf ihre vorhandene Motivation und neu erlangte Motivation durch das Trainingsprogramm eingehen. Außerdem möchte ich herausfinden, ob die vorrausgesetzten Kompetenzen zu viele waren und ob diese durch das Programm dennoch neu dazuerlangt werden konnten. Ich gehe bei Hobbymusikern dabei am ehesten davon aus, dass diese noch nie oder nur im Instrumentalunterricht, sich mit Gehörbildung beschäftigt haben. Auf dieser Grundlage ist es für mich interessant ob das entwickelte Trainingsprogramm sie dazu anregen konnte sich etwas näher mit Gehörbildung auseinanderzusetzen und ob sie denken, dass sie in der Ausübung ihres Hobbys einen Mehrwerte daraus ziehen können.
Die Meinung der Studierenden ist vorallem wichtig in Bezug auf das Musikstudium und spezifisch auf Gehörbildung. Ich erhoffe mir von Studierenden den Nutzen für das Studium zu erfahren und was sie sich an einer solchen Anwendung noch zusätzlich wünschen würden. Außerdem möchte ich von Studierenden erfahren, ob die Anwendung das Wissen abfragt, welches von ihnen verlangt wird und sie dementsprechend lernen müssen.
Als letzte Gruppe möchte ich versuchen einige professionelle Musiker zu befragen. Deren Meinung ist besonders interessant, in Rückblick auf ihre Ausbildung, ob sie sich vorstellen könnten, dass ihnen eine solche Software in dieser Zeit hätte geholfen. Außerdem wäre es interessant, ob sie sich vorstellen könnten so Gehörbildung weiterhin zu trainieren und ob sie diese Trainingssoftware ihren Kollegen weiter

empfehlen würden.
Abschließend gibt es einige Aspekte zu erwähnen, welche alle der oben genannten Gruppen betreffen und sie dementsprechend alle beantworten können sollten. Unter diese Aspekte fallen die allgemeinen Punkte der Software, wie etwa die Performance oder die Zufriedenheit mit der Tonerkennung. Weiterhin fallen unter die allgemeinen Punkte, wie gut die Nutzer mit der generellen Handhabung der Software zurecht kamen und ob sie die Software planen weiter zu verwenden. Weiterhin ist es wichtig grobe Angaben zu der verwendeten Hardware zu erhalten, falls die Tonerkennung nicht gut funktionierte lassen sich so hoffentlich Zusammenhänge zu unvorteilhaften Mikrofonen herstellen.

5 Prototypische Realisierung in Unity3D (nicht zu ausführlich schreiben)

6 Evaluation / Validierung der erarbeiteten Methoden und Konzepte

7 Zusammenfassung

Literaturverzeichnis (APA)

Literaturverzeichnis

[Alv09] José Rodríguez Alvira. Teoria. Online, 2021-02-09.

[AVF08] Rafael George Amado and Jozue Vieira Filho. Pitch detection algorithms based on zero-cross rate and autocorrelation function for musical notes. In *2008 International Conference on Audio, Language and Image Processing*, pages 449–454. IEEE, 2008.

[Bru21] Justus-Knecht-Gymnasium Bruchsal. Jkg bruchsaal musik. Online, 2016-12-21.

[But06] *Window Functions*, pages 69–88, 118. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[Jak] Hans Lavdal Jakobsen. Earmaster. Online.

[LK04] Nelson Posse Lago and Fabio Kon. The quest for low latency. In *ICMC*, 2004.

[Man09] Musikhochschule Mannheim. Gehörbildung online. Online, 2021-02-09.

[mus09] musictheory.net. musictheory. Online, 2021-02-09.

[NK03] Michael Peter Norton and Denis G Karczub. *Fundamentals of noise and vibration analysis for engineers*. Cambridge university press, 2003.

[Rie] Walter Riedinger. Musikgrad. Online.

[Tec09a] 2020 Unity Technologies. Unity3d dokumentation audio overview. Online, 2021-02-09.

[Tec09b] 2020 Unity Technologies. Unity3d dokumentation microphone. Online, 2021-02-09.

[Vesa] Morten Vestergaard. Earbeater. Online.

[Vesb] Morten Vestergaard. Tonedear. Online.

[YMFA05] Yuuki Yazama, Yasue Mitsukura, Minoru Fukumi, and Norio Akamatsu. A simple algorithm of pitch detection by using fast direct transform. In *2005 International Symposium on Computational Intelligence in Robotics and Automation*, pages 205–209. IEEE, 2005.

[Zie09] Wieland Ziegenrücker. *ABC Musik Allgemeine Musiklehre*. Breitkopf & Härtel, 7 edition, 2009.