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

1 Einleitung

Im allgemeinen unterscheidet man bei den Gehörarten zwischen dem sogenannten absoluten Gehör und dem relativen Gehör. Während eine Person mit einem absoluten Gehör eine Note ohne eine Referenz und nur mit ihrem Gehör bestimmen kann, kann eine Person, welche nur ein ausgebildetest relatives Gehör hat, eine Note nur mit einem Vergleichston bestimmen. Wichtig ist dabei, dass das relative Gehör die meisten Menschen trainieren können, wohingegen man das absolute Gehör nicht trainieren kann [GVG06]. Will man das relative Gehör trainieren, so hat man die Wahl aus verschiedenen Übungsansätzen und Methoden. Unter diese Methoden fallen beispielsweise das Erhören und Vervollständigen von Intervallen und Akkorden. Weiterhin werden häufig Diktate verwendet um das relative Gehör weiter zu schulen, wie etwa ein Melodiediktat, wo die lernende Person die richtigen Noten der gehörten Melodie aufschreiben muss. Ein großer Aspekt ist allerdings auch, dass die Lernenden nicht nur Töne, Intervalle, Akkorde und Melodien erkennen können, sondern auch reproduzieren können müssen. Das Reproduzieren kann dabei heißen, dass ein Intervall gesungen werden muss oder ein Akkord vervollständigt werden soll. Die Fähigkeit diese all das reproduzieren zu können ist vorallem für Musiker und Musikerinnen wichtig, welche mit anderen zusammen musizieren. Dirigenten und Orchesterleiter sind ebenfalls stark angewiesen auf die Fähigkeit Intervalle, Akkorde oder Melodien vorsingen zu können um so den Musikern die gewollten musikalischen Ideen vermitteln zu können.

1.1 Motivation

„Study of music and its literature presupposes the ability to hear, read, and write the language. Lacking this ability one is comparatively helpless and dependent. This perfectly obvious truth is universally accepted without question in the study of every language excepting music. Teachers, capable otherwise, allow and encourage the serious study of music by students who do not comprehend what they hear. Ability to hear is the essence of music, the core of education in music is ear training ” [Spe47]

Herbert S. Spencer beschreibt in einem Artikel von 1947 die Notwendigkeit der Gehörbildung um Musik in vollen Zügen wahrnehmen und ausüben zu können. Er setzt dabei das Hören von Musik mit dem Lernen von Sprachen gleich, wobei er feststellt, dass einzig in der Lehre der Musik kaum Wert auf das Hören dieser gelegt wird. Er sieht darin einen großen Verlust und beschreibt einen Musiker ohne ausgebildetes Gehör als vergleichsweise Hilflos und Abhängig.

Obwohl Gehörbildung bereits seit langer Zeit von Musikern als wichtig für das Musizieren wahrgenommenen wird, findet die Förderung dieser heutzutage einzig im Studium der Musik statt. Weiterhin sitzt zum Üben der Gehörbildung häufig noch ein Lehrer an einem Klavier, welcher etwas vorspielt und die Schüler ihm zuhören und das Gehörte aufschreiben müssen. Der Unterricht der Gehörbildung ist dementsprechend nicht allgemein zugänglich für die breite Masse der Musiker und stark lokalisiert in die Umgebung des Seminarsaals. Es existieren bereits einige Programme, welche bei dem Üben der Gehörbildung der lernenden Person helfen, jedoch fehlen diesen häufig die Komponente gestellte Aufgaben mithilfe von Tonerkennung automatisch zu überprüfen. Ziel dieser Arbeit soll sein, die existierenden Trainingsprogramme zur Gehörbildung auf ihre wichtigsten Aspekte zu untersuchen und basierend auf den Ergebnissen der Analyse ein prototypisches Trainingsprogramm mithilfe von Unity3D zu entwickeln, welches die wesentlichen Aspekte der vorhandenen Software aufgreifen und verbessern soll. Das entwickelte Programm soll bestenfalls sowohl auszubildende Musiker im Studium unterstützen, als auch Amateurmusiker in der Fortbildung des Gehörs fördern. Unity bietet keinerlei Funktionen um von Haus aus eine echtzeit Tonverarbeitung zu implementieren, weshalb die Tonerkennung eine besondere Herausforderung darstellen wird. Weiterhin muss ein ansprechendes User Interface für sowohl Amateurmusiker, wie auch (auszubildende) professionelle Musiker entwickelt werden.

1.2 Aufbau der Arbeit

Im nächsten Kapitel werden zunächst einige wichtige Definitionen zu den Oberthemen der Signalverarbeitung, Musiktheorie und zu Serious Games gegeben, welche für das restliche Verständnis der Arbeit wichtig sind. Daraufhin werden im dritten Kapitel zunächst die aktuell wichtigsten bereits vorhandenen Programme zur Gehörbildung vorgestellt und zu vorher definierten Gesichtspunkten untersucht und analysiert. Im zweiten Teil des Kapitels werden außerdem bekannte Methoden zur Tonerkennung und zur echtzeit Eingabe von Ton in Unity ebenfalls untersucht und analysiert. Basierend auf den Erkenntnissen des dritten Kapitels, wird im darauffolgenden Kapitel ein Trainingsprogramm zur Gehörbildung mit Unity konzeptioniert. Die Konzeption ist dabei aufgeteilt in den Entwurf des Spiels aus der Sicht des Nutzer (user-centered design), des Entwurfs der Systemkomponenten, sowie dem Evaluationskonzept. Das fünfte Kapitel beschäftigt sich anschließend mit der Implementierung des Trainingsprogramms in Unity und geht im detail auf die implementierten Funktionen ein. Das darauffolgende Kapitel beleuchtet die Evaluationsschritte näher, wobei auf die Vorbereitung und die Durchführung der Evaluation eingegangen wird, sowie die Ergebnisse der Evaluation analysiert und ausgewertet werden. Im letzten Kapitel wird schließlich eine Zusammenfassung des Themas gezogen, sowie ein Ausblick auf mögliche erweiternde Funktionen, sowie weitere Anwendungsbereiche gegeben.

2 Grundlagen

2.1 Signalverarbeitung

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.

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 Musiktheorie

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]

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] [Sen01]

Tonsysteme und Stimmung

Wie bereits in 2.2 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.

$$i = i_1 + i_2 = 1017$$

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

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

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) eines Tons. Bei der reinen Stimmung 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]

2.3 Serious Games

Serious Games sind digitale Spiele, welche nicht nur unterhalten sollen, sondern auch ein weiteres Ziel verfolgen. Dies wird gemacht, um eine Brücke zwischen Bildung und der Vermittlung von neuem Wissen und Unterhaltung zu schlagen. Dieses Ziel wird auch das Characterizing Goal genannt. Dabei ist es wichtig, dass Serious Games dieses zusätzliche Ziel umsetzen wollen, ohne dabei den Spielspaß oder die generelle Spielerfahrung einzuschränken. Die Spiele sollen weiterhin das Hauptziel der Unterhaltung verfolgen und lediglich während dem spielen automatisch das Characterizing Goal vermitteln. Der Namenszusatz „serious ” soll also nicht bedeuten, dass das Spiel etwa in dem Sinne seriös oder langweilig sind, sondern verweist viel mehr auf die „seriösen ” Branchen in denen Serious Games eingesetzt werden können. Unter diese Branchen fallen unter anderem die Bildung, Sport, Krisenmanagement, Städteplanung, Religion oder Politik [**?**] . Einige prominente Beispiele sind Spiele, wie etwa „Coding Pirates” [Meg01], welches neues Wissen vermitteln möchte, oder ein Spiel welches zur Bewegung anregen möchte, wie etwa „Pokemon Go”[AWH16] oder die „Moto Tiles” [LLW18]. Um Serious Games einheitlich klassifizieren zu können, wurde das sogenannte „Serious Games Metadata Format” entwickelt. Dieses Format ist aufgebaut aus mehreren Stufen. Die erste Ebene (Core), enthält allgemeine Informationen über das Spiel, welche Interessenten einen Überblick über die Qualität, Innovationen und Anwendungen des Spiel geben sollen. Die zweite Ebene (Detailed) dient Kritikern und Experten als Grundlage der Bewertung eines Serious Games. Sie beinhaltet daher Punkte wie etwa Informationen zur User Experience und einem Fragebogen zur generellen Bewertung. Auf der dritten Ebene (Extension) wird schließlich eingegangen auf spezifische Aspekte des Spiels eingegangen. So werden hier beispielsweise bei Exergames Trainingsprogramme und genaue Sportübungen bewertet [GGS11]. Eine Plattform um nach dem Metadaten Format Spiele zu filtern ist das Serious Games Information Center (SG-IC) [e.V10].
Im Gegensatz zu einem Serious Game steht das Konzept der Gamification. Bei der Gamification handelt es sich um den Ansatz spielerische Komponenten in eine Umgebung einzubinden, welche kein Spiel ist. Populäre Beispiele für Anwendungen, welche Gamification umsetzen, sind „Duolingo” [duo]. Spielerische Elemente können ein Highscore System mit online Leaderboard sein oder die Möglichkeit Trophäen zu sammeln. Die Gamification Elemente sollen dem Nutzer meistens ein Gefühl von Fortschritt geben.

3 Analyse

Im folgenden Kapitel werde ich zunächst auf den aktuellen Stand der Trainingsprogramme für Gehörbildung eingehen und diese auf ihre wichtigsten Aspekte analysieren und wie man diese noch verbessern könnte. Daraufhin werde ich nach Methoden und Konzepten zur Echtzeiterkennung von Audio mit Unity recherchieren und analysieren, um diese in einen Prototypen eines Trainingsprogramms einzubinden.

3.1 Analyse von Trainingsprogrammen zur Gehörbildung

Ich stelle zunächst die interessantesten Anwendungen und Websites zum Training der Gehörbildung vor.

3.1.1 Methodik

Ich werde bei der Begutachtung der State of the Art Anwendungen vorallem auf die Aspekte der Übungsvielfalt, sowie dem Aufbau der Übungen achten. Unter die Übungsvielfalt fällt unter anderem eine Einschätzung der Wiederspielbarkeit und des abgefragten Wissens, sowie die Breite der verfügbaren Übungen zu verschiedenen musikalischen Bereichen innerhalb der Gehörbildung. Zu diesen Bereichen zählen beispielsweise Intervalle, Akkorde, oder Melodiediktate. Außerdem möchte ich auf die Weiterbildungsmöglichkeiten und die generelle Wissensvermittlung achten. Unter Wissensvermittlung verstehe ich hier, ob das Wissen beispielsweise innerhalb des Spiels vermittelt wird, oder nur zum Nachlesen bereitsgestellt wird. Weiterhin interessiert mich wie der Nutzer mit der Anwendung interargieren kann, dabei sind die Punkte der Navigierbarkeit, die Eingabemöglichkeiten, sowie der generelle Aufbau des UIs wichtig.

3.1.2 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. [Ves]

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. [dav]

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]

3.1.3 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 Anwendung 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 live überprüfen kann. Hat der Nutzer die Note lang genug gehalten, so wird Feedback gegeben mithilfe einer Linie durch das Notensystem, welche den Verlauf des gesungenen Tons angibt. 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 genann-

ten 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, was so umfangreich ist mit so vielen verschiedenen Modulen zum Lernen und einer sehr großen Anpassungsmöglichkeit an die einzelnen Bedürfnisse der Nutzer, dass es sämtliche anderen Programme übetrifft und somit den aktuellen Stand der Trainingsprogramme am besten widerspiegelt.

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

Ich gehe zunächst auf den Rechercheprozess zu den Ansätzen der Echtzeiterkennung von gesungenen Tönen ein und will daraufhin die interessantesten Ansätze analysieren.

3.2.1 Recherche

Ich bin bei der Recherche schrittweise vorgegangen und habe zunächst generell nach der Tonverarbeitung und Aufnahme in Unity recherchiert. Die erste Quelle für solche Informationen ist in der Regel die Dokumentation, dementsprechend habe ich mir die Unity Dokumentation angesehen und nach allem gesucht was Audio, Sound, Frequency oder Microphone erwähnt. Unter dem Schlagwort Microphone bin ich auf die Dokumentation der Microphone Klasse gestoßen, wo mir schließlich auffiel, dass es keinen direkten Weg gibt, um in Echtzeit auf die Microphone Daten zuzugreifen. [Tec09d] Da ich zunächst herausfinden wollte, wie und ob es möglich ist das Mikrofon in Echtzeit auszulesen mit Unity recherchierte ich weiter. Nach einer Suchanfrage „Unity Microphone in realtime" bin ich auf einen Artikel aus den Unity Foren gestoßen, welcher genau das Problem behandelte. [Meg01] Jedoch ergaben sich nach ersten Tests Probleme mit der Latenz, da diese noch nicht auf Echtzeit-Niveau war. Meine Recherche führte mich nun wieder zurück in die Unity Dokumentation wo ich nach weiteren Suchanfragen zu Audio auf den AudioManager stieß. In den Projekteinstellungen lässt sich die DSP Buffer Size der Anwendung anpassen zwischen bester Latenz und bester Qualität. [Tec09a] Veränderte ich diese Einstellung auf „good Latency" funktionierte der kleine Test zum Einlesen der Daten in Echtzeit. Unter den Ergebnissen der oben genannten Schlagworten in der Unity Dokumentation fand ich als nächstes eine Übersicht zu Audio in Unity. [Tec09b] Diese Übersicht 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 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 das Spektrum der Audiodaten des dazugehörigen AudioClips zuzugreifen bzw. es zu berechnen. [Tec09c] Die Funktion ermöglicht es direkt das Spektrum des AudioClips einer AudioSource mithilfe einer schnellen Fouriertransformation zu berechnen. 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 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) 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. Ich fand unter den Suchbegriffen „Pitch detection algorithm" bei Google Scholar und Google einige PDAs. Dabei achtete ich schon hier darauf, dass diese einerseits eine gute Erkennungsrate haben würde, sodass ich auf Basis dessen ein Trainingsprogramm aufbauen kann, jedoch nicht zu viel Rechenarbeit benötigen würden. 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 der Zero-Crossing rate [AVF08].

3.2.2 Audioinput und Verarbeitung mit Unity

Bevor wir uns mit der eigentlichen Erkennung von Tönen beschäftigen können, müssen wir etablieren, wie wir Töne vom Nutzer mithilfe von Unity einlesen und speichern. Glücklicherweise bietet Unity einige Hilfen um Audio einzulesen. Es ist uns möglich auf Audiorohdaten vom Nutzer zuzugreifen und somit mit diesen zu arbeiten. Weiterhin ist es auch möglich das Frequenzspektrum dieser Audiodaten, mithilfe einer von Unity bereitgestellten Methode, zu berechnen. Möchte man jedoch diese Methode verwenden, so ist man auch an die Datenstrukturen von Unity gebunden. Die Möglichkeiten umfassen jetzt also, entweder eine eigene Datenstruktur entwickeln zu können oder das Frequenzspektrum mithilfe der internen Methoden zu berechnen. Wie wir in der nächsten Sektion erkennen werden, bietet eine eigene Datenstruktur eine größere Auswahl an Pitch Detection Algorithmen, da durch die eigene Datenstruktur mehr potentielle Informationen zur Verfügung stehen. Es ist so möglich mithilfe von eigens implementierten schnellen Fouriertransformation selbst das Frequenzsprektrum der Daten zu berechnen. Auf der anderen Seite, ist es wahrscheinlich, dass die von Unity zur Verfügung gestellten Funktionen im Kontext einer Unityanwendung performanter sind. Weiterhin wissen wir bereits, dass der Grundton eines Klangs immer die tiefste dominante Schwingung im Frequenzspektrum des Klangs ist. Aus diesem Vorwissen, können wir schließen, dass eine Fouriertransformation auf den Audiodaten des Nutzers sehr wichtig sein wird. Mit dem Hintergrund eine echtzeit Tonerkennung entwickeln zu wollen, fällt dementsprechend die Auswahl Methode um Audioinput vom Nutzer zu lesen und weiterzuverarbeiten auf die Datenstruktur von Unity.

3.2.3 Ansätze der Pitch Detection Algorithmen

Infrage kommende Pitch Detection Algorithmen lassen sich in zwei Klassen einteilen. Es gibt zum einen Algorithmen, welche versuchen den Grundton zu berechnen, indem Operationen auf dem zeitabhängigen Signal ausgeführt werden. So kann beispielsweise der Grundton berechnet werden, indem man sich widerholende Freqeunzmuster sucht und aus diesen auf den Grundton schließt. [dlCMS01] Die andere Klasse versucht hingegen den Grundton aus dem Freqeunzspektrum zu berechnen. Wir wisse bereits, dass der Grundton innerhalb eines Spektrums die tiefste schwingende Freqeunz in diesem ist. Es gibt innerhalb dessen verschiedene Ansätze diese tiefste Frequenz zu berechnen, wie es etwa das sogenannte Cepstrum macht, welches zunächst das Spektrum berechnet, innerhalb diesem dann alle Werte logarithmiert und schließlich wieder die Umkehrfunktion der Spektrumsfunktion anwendet. Auf die Vorteile und Nachteile dieser Ansätze und auf weitere Unterschiede innerhalb dieser möchte ich nun eingehen.
Ich fange zunächst mit den Algorithmen an, welche nicht innerhalb des Frequenzspektrum arbeiten. Unter diesen Ansatz fallen Algorithmen und Ansätze wie etwa die „zero-crossing rate", „average magnitude difference function (AMDF)" oder die „average squared mean difference function (ASMDF)". Ich möchte einen kurzen Überblick über die Funktionsweisen dieser Algorithmen geben.

Zero-crossing Rate (ZCR)

Die ZCR ist einer der einfachsten Techniken um die Grundfrequenz innerhalb eines Klangs zu berechnen. Es wird, wie der Name schon verrät, gezählt wie oft das Signal vom Positiven auf Null und weiter ins Negative fällt oder anders herum vom Negativen Raum auf Null und in den positiven Raum ansteigt. Man kann aus der Häufigkeit dieser crossings die Grundfrequenz nun wie folgt berechnen.

$$F_{Signal} = \frac{zcr * f_{Sample}}{2 * |f_{Sample}|}$$

Wobei f_{Sample} die Samplerate der Aufnahme ist, $|f_{Sample}|$ die Anzahl der untersuchten Samples und zcr die erkannten zero-crossings. Jedoch ist dieser Ansatz in den meisten Fällen sehr ungenau und funktioniert nur gut bei Sinustönen. Die Laufzeit des Algorithmus hingegen sehr gut aufgrund seiner sehr simplen Natur. [AVF08]

Average magnitude difference function (AMDF)

Der AMDF versucht den Unterschied zwischen dem Originalsignal und einer verzögerten Version des gleichen Signals zu minimieren. Es wird dabei so vorgegangen, dass das verzögerte Signal von dem Original subtrahiert wird und anschließend die Größen der Unterschiede summiert werden. Minimiert man den berechneten AMDF Wert, so erhält man eine ungefähre Näherung an die Grundfrequenz. [RSC+74] Die mathematische Definition lautet wie folgt.

$$D_r = \frac{1}{L} \sum_{j=1}^{L} |S_j - S_{j-r}|$$

r beschreibt hierbei die Verzögerung des Signals. Der Algorithmus kann je nach Größe des Verzögerungsfensters eine große Rechenzeit beanspruchen. Der Unterschied zwischen dem AMDF und dem average squared mean difference function (ASMDF) liegt lediglich darin, dass bei letzterem die Differenzen zwischen den Signalen zusätzlich quadriert werden. Diese Anpassung führt zu einer höheren Robustheit

Es exisiteren weiterhin Algorithmen, welche nicht auf dem ursprünglichen Zeitabhängigen Signal arbeiten, sondern auf dem frequenzabhängigen Signal, dem Spektrum. Zu diesen Algorithmen zählen die so genannen „Cepstrum Analyse", sowie das „harmonic product spectrum" und ein „Maximum Likelihood" Ansatz. All diese Methoden setzen vorraus, dass minestens eine Fouriertransformation auf dem Signal durchgeführt wird.

Cepstrum

Das Cepstrum beschreibt ein Signal, welches zunächst in sein Spektrum überführt wird, mithilfe einer Fouriertransformation. Daraufhin wird innerhalb des Spektrums logarithmiert und schließlich wieder die Umkehrfunktion der Fouriertransofmration angewandt. Welche Bestandteil im zweiten Schritt genau logarithmiert wird, hängt von der Art des Cepstrums ab. Wendet man das „Power Cepstrum" an, so wird das „Power Spectrum" logarithmiert. Um das „Complex Cepstrum" zu berechnen, wird das gesamte Frequenzspektrum logarithmiert und möchte man das „Real Cepstrum" berechnen, so muss man die Amplitudenwerte logarithmieren. [OS04] Zur Pitch Detection ist jedoch nur das Complex Cepstrum geeignet. Möchte man aus diesem nun die erkannte Grundfrequenz berechnen, so muss man lediglich das absolute Maximum wählen.

Harmonic product spectrum

Ein weiterer Ansatz die Grundfrequenz zu berechnen besteht darin, den größte gemeinsamen Teiler der Obertöne zu berechnen. Wie wir bereits wissen, sind die Obertöne nichts anderes als Vielfache der Grundfrequenz. So sind beispielsweise für eine Grundfrequenz von 220Hz einige Obertöne 440Hz, 660Hz und 880Hz. Es ist trivial, dass der ggT von diesen Werten 220Hz ist, was wieder der Grundfrequenz entspricht. Das Harmonic product spectrum nutzt diesen Sachverhalt aus, indem es das Spektrum um immer steigende Faktoren $1, 2, 3, ..., N$ downsampled bzw. anschaulich erklärt staucht. Legt man die gestauchten Spektren übereinander, so muss die Grundfrequenz auf dem gleichen Wert liegen, wie der jeweils $1, 2, 3, ..., N$ Oberton. Mulitpliziert man diese Spektren dementsprechend miteinander, so ist der größte Ausschlag bei der Grundfrequenz zu finden. Bei diesem Ansatz kommt es häufig zu Oktavenfehlern, sodass der Ton eine Oktave zu hoch erkannt wird. [dlCMS01]

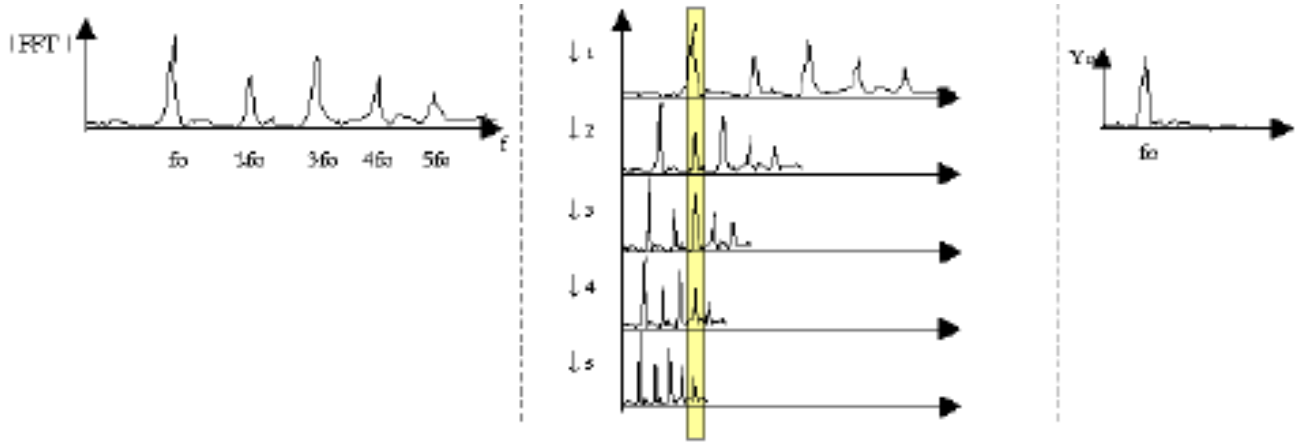

Abbildung 3.1: Überblick über den HPS Algorithmus [dlCMS01]

Maximum Likelihood

Der Maximum Likelihood Algorithmus versucht das vorhandene Spektrum mit einer Menge an idealen Spektren zu vergleichen und wählt anhand dieser den am wahrscheinlichsten dazugehörigen Ton aus. Dabei versucht der Algorithmus den Fehler zwischen dem gegebenen Spektrum und den idealen Spektren zu minimieren. Da dieser Algorithmus fest defnierte Referenzspektren benötigt, funktioniert er am besten, wenn der Ton von einem Instrument erzeugt wird, welches eine feste Stimmung hat, wie beispielsweise ein Klavier. Andernfalls kann es zu stark springenden Ergebnissen kommen. [dlCMS01]

3.2.4 Analyse der algorithmischen Ansätze zur Pitch Detection

In der folgenden Analyse will ich sowohl die Komplexität der Algorithmen berücksichtigen, als auch die erwartete Robustheit. Ziel der Analyse soll sein, die Ansätze der Algorithmen aufzubereiten, um daraufhin in der Konzeption einen Algorithmus zu konzipieren, welcher für eine Umsetzung in Unity geeignet ist.
Das größte Unterscheidungsmerkmal der oben vorgestellten Algorithmen ist, wie bereits erwähnt, die Notwendigkeit einer Fouriertransformation. Das Berechnen der Fouriertransformation fügt einen weiteren rechenintensiven Schritt zu dem Algorithmus hinzu. Bei der Entwicklung eines Algorithmus, welcher Ergebnisse in Echtzeit liefern muss, ist das offensichtlich ein negativer Punkt, welcher zu berücksichtigen ist. Naive betrachtet sollten so die Algorithmen, welche im Freqeunzbereich arbeiten auch rechenintensiver sein. So benötigt das Cepstrum eine Fouriertransformation und zusätzlich die Umkehrfunktion

dieser. Das HPS und der Maximum Likelihood Algorithmus jedoch nur eine anfängliche Fouriertransformation, woraufhin der Rest der Berechnung auf dem entstandenen Spektrum durchgeführt werden können. Innerhalb der Algorithmen, welche im Zeitabhängigen Raum arbeiten, ist die Zero-crossing rate offensichtlich weniger rechenintensiv, als die AMDF. Insofern lässt sich schließen, dass die zero-crossing rate der Laufzeiteffizienteste Algorithmus unter den vorgstellten Algorithmen ist. Betrachtet man jedoch die Robustheit der Algorithmen gegenüber Störgeräuschen und die durschnittliche Genauigkeit, so zeigt sich, dass die zero-cross rate nur gute Aussagen unter Idealbedingungen treffen kann [AVF08]. Generell lässt sich sagen, dass die Algorithmen, welche in der Frequenzdomäne arbeiten, unter dem Aspekt der Robustheit bessere Aussaugen treffen. Sowohl ZCR als auch AMDF sind in ihrer reinen Implementaion stark anfällig für Hintergrundgeräusche [dlCMS01]. Diese Beobachtung lässt sich unter anderem daher begründen, dass es einfacher ist in der Frequenzdomäne einen Noisefilter anzuwenden, welcher beispielsweise Frequenzen unter 50Hz aus dem Spektrum entfernt, jedoch aber auch sich durch die Eigenschaft der Frequenzdomäne im Allgemeinen genauere Aussagen über die Verhältnisse von Frequenzen zueinander treffen lassen.
Zusammenfassend sind Algorithmen in der Frequenzdomäne im Allgemeinen mächtiger und können genauere Aussagen über die Grundfrequenz treffen, weisen allerdings auch durch das nötige Berechnen einer Fouriertransformation eine wesentlich höhere Laufzeitkomplexität auf. Bei der Entwicklung eines Echtzeitsystems, muss man dementsprechend abwägen, wie hoch man die Rechenanforderungen setzen kann, ohne eine Vielzahl an Nutzern auszuschließen, aufgrund von fehlender leistungsfähiger Hardware.

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 auf Systemsicht mit den Eingaben umgegangen werden soll und schließlich das Erarbeiten eines Evaluationskonzept.

4.1 Konzeption des Spiels

Ich werde nun auf die allgemeine 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, sowie auf die Grundideen des Programms, werde ich auch eingehen. Der Nutzer soll mithilfe der Anwendung effektiv das Erkennen und Reproduzieren von Intervallen üben. Zum Lernen sollen dem Nutzer zwei Möglichkeiten gegeben werden, entweder kann Er eine geführte Lektionen basierte Einführung in das Thema absolvieren, wo Ihm auch Hilfestellungen gegeben werden um das Intervall effektiv zu erkennen, oder Er kann endlos Intervalle bestimmen zur Wiederholung und Festigung der Fähigkeiten. Die Aufgaben werden hierbei natürich zufallsgeneriert, sodass dem Nutzer niemals die Aufgaben ausgehen. Das Zufallsgenerieren von Aufgaben ist nicht selbstverständlich, wie aus der Analyse der State of the Art Anwendungen klar wird, denn einige der Anwendungen erlauben nur das Beantworten von vordefinierten Aufgaben. Dem Nutzer soll außerdem die Möglichkeit gegeben werden diese Aufgaben über das Mikrofon in Form von Gesang zu beantworten. Aus der Analyse des State of the Art wird klar, dass diese Option so gut wie keine Anwendung bietet, obwohl sie den Lerneffekt deutlich verbessern kann. Weiterhin soll es dem Nutzer möglich sein die Aufgabe durch einen Klick in das Notensystem zu beantworten, sollte dieser nicht singen wollen. Diese alternative Eingabemethode halte ich für besser als einen Button mit der richtigen Lösung zu drücke, wie es einige der State of the Art Anwendungen umsetzen. Im Gegensatz zu der simplen Auswahl einer Lösung durch einen Button kann durch das Anklicken der Note im Notensystem weiterhin ein Gefühl für das Erkennen von Noten vermittelt werden. Idealerweise kann die Anwendung alle relevanten Aspekte der Gehörbildung abfragen, wie etwa Akkorde erkennen oder ein Melodiediktat zu stellen, das würde jedoch den Rahmen dieser Arbeit sprengen. Ein einfaches „Authoringtool" zum Erstellen von Aufgaben, für den Lektionen basierten Teil ist hierbei schon ein erster Ansatz, um eventuell Lehrern die Möglichkeit zu geben eigene Aufgabensätze zu konzipieren und diese einfach einzubinden. Dabei soll es möglich sein das geforderte Intervall zu definieren und dazu eine Aufgabenstellung, sowie den Grundton zu definieren. Es ist in dem Zusammenhang der Tonerkennung auch wichtig, dass diese auf die Stimme jedes Nutzers angepasst werden kann, sodass auch hierfür ein Optionenmenü mit entsprechenden Anweisungen zur Anpassung der einzelnen Regler wichtig ist. Will man den Nutzer singen lassen, so muss auch sicher gestellt werden, dass dieser in seiner bevorzugten Stimmlage singen kann. Auch für diese Anpassung muss eine Option vorhanden sein. Weitere Anpassungsmöglichkeiten auf ür die Aufgaben selbst, sollen dem Nutzer weitere Tool an die Hand geben um sich so gezielt wie möglich fortzubilden. So etwa soll es dem Nutzer außerdem möglich sein den Grundton entweder zufällig zu generieren oder auch fest einzustellen oder die Richtung des Intervalls anpassen können, also ob dieses nach oben oder nach unten vervollständigt werden soll. Diese Vielzahl an Einstellungen sollen es dem Nutzer nicht nur ermöglichen ein bestimmtes Themenfeld zu lernen, sondern auch damit dieser eine gewissen Kontrolle über die Schwierigkeit hat und somit immer genau richtig beansprucht wird. Aus der Analyse kann man zudem erkennen, dass es besonders für unerfahrenere Nutzer, sehr von Vorteil sein kann, wenn es einige Erklärungen zu den wichtigsten musikalischen Grundlagen direkt auf der Plattform zum Nachlesen gibt.

4.1.1 Hauptmenü

Wird die Anwendung gestartet, gelangt man in das **Hauptmenü**. Von dem Hauptmenü aus kann man zu den Spielmodi „Endlosspiel" und „Modulen" navigieren, sowie zu den allgemeinen Einstellungen und zu dem Theoriebuch. Außerdem lässt sich die Anwendung von hier aus auch beenden. Es wird dem Nutzer zur Überprüfung außerdem angezeigt in welcher Tonlage die Aufgaben generiert werden, sollte er einen Spielmodus starten.

4.1.2 Einstellungen

In den **allgemeinen Einstellungen** wird dem Nutzer ein kurzer Text zu der Anpassung der Tonerkennung angezeigt, sowie ein Notensystem an welchem dieser die Tonerkennung ausprobieren kann und überprüfen kann, ob diese gut genug auf die eigene Stimme eingestellt ist. Das Anpassen der Tonerkennung soll mithilfe von zwei Slidern geschehen, welche jeweils die zwei Werte „Threshold" und „Filter" anpassen. Slider bieten die genauste Schrittweite bei der Anpassung und sind gleichzeitig einfach zu verstehen. Weiterhin gibt es in den allgemeinen Einstellungen Dropdown Menüs über welche man die Tonlage und das Eingabegerät ändern kann. Durch einen Button gelangt man zurück zum Hauptmenü.

4.1.3 Theoriebuch

Das **Theoriebuch** soll aus einer Navigierleiste links im Bildschirm und einem Contentfenster im restlichen Bereich bestehen. Die Navigierleiste beinhaltet mehrere Buttons mit den Themen als Label, welche den Content in dem restlichen Bereich des Fensters anpassen. Da diese Szene von drei verschiedenen Szenen aufgerufen wird, existiert außerdem ein Button, welcher die vorherige Szene wieder lädt. Die Infotexte zu den Themen werden aus einem Musiktheorie Buch übernommen und sollten somit die notwendige pädagogische Kompetenz aufweisen um die Inhalte gut genug zu vermitteln. [Zie09]

4.1.4 Endlosübung

Die **Endlosübung** zeigt auf der unteren Häfte des Bildschirms ein Notensystem an, auf welchem nicht nur der Grundton gerendert werden soll, sondern auch die Antwort eingegeben bzw. angezeigt werden soll. Entscheidet sich der Nutzer dazu die Note singen zu wollen, so soll die erklingende Note automatisch im rechten Drittel des Notensystem in echtzeit erscheinen. Möchte der Nutzer jedoch die Note per Maus setzen, so wird die unter dem Cursor liegende Note erscheinen, bewegt sich der Cursor weiter soll die Note nicht mehr gerendert werden. Klickt der Nutzer auf eine Note mit einem Linksklick, so soll diese Note eingeloggt werden und auch keine anderen Noten mehr gerendert werden können. Möchte der Nutzer einen Halbton eingeben, muss dieser einen Toggle aktivieren, welcher die ausgewählte Note um einen Halbton verschiebt. Um die Aufgabe zu lösen, muss die richtige Antwort für eine bestimmte Zeit gesungen werden, um falsche Erkennungen zu reduzieren. Wurde die Aufgabe richtig beantwortet erscheint ein Feedbacktext im oberen rechten Bereich der Szene, welche dem Nutzer mitteilt, dass die Antwort richtig ist. Erscheint dieser Text nicht, so ist die Aufgabe auch noch nicht gelöst. Sobald die Aufgabe gelöst ist, kann der Nutzer selbst entscheiden, wann er die nächste Aufgabe beginnen möchte mithilfe eines Buttons. Der extra Schritt den Nutzer auf weiter klicken zu lassen, soll verhindern, dass noch der Ton gesungen wird der letzten Aufgabe und schon der nächste Grundton gleichzeitig abgespielt wird. Zu Beginn jeder Aufgabe erklingt der Grundton des Intervalls über das Standardausgabegerät des Nutzers. Über einen Button soll es möglich sein, diesen Grundton so oft wie möglich sich erneut anhören zu können. Es ist außerdem möglich das Theoriebuch aufzurufen über einen Button. Weiterhin soll es möglich sein ein Anpassungsmenü aufzurufen, in welchem der Nutzer zurück in das Hauptmenü navigieren kann und die aktuelle Aufgabe überspringen kann, sollte er das wünschen. Es soll in diesem

Menü außerdem möglich sein die Eingabemethode zwischen Gesang und manueller Eingabe zu wechseln, sowie einen festen Grundton definieren und die Richtung des Intervalls bestimmen zu können. Das Anpassungsmenü soll sich durch einen Klick auf einen Button schließen lassen.

4.1.5 Modulübung

Zuletzt soll der **modulbasierte Spielmodus** ähnlich wie das Endlosspiel ein großes Notensystem implementieren in der unteren Hälfte des Bildschirms, während in der oberen Häfte alle relevanten Informationen angezeigt werden. Der Unterschied zum Endlosspiel ist dabei, dass statt zufällig generierten Aufgaben vorher definierte Aufgaben gestellt werden. Die Aufgaben können dabei den Grundton, das Intervall und den Text anpassen um so eine möglichst flexible Aufgabenstellung zu erlauben. Es soll außerdem möglich sein reine Textpassagen anzuzeigen, ohne eine Aufgabe zu stellen. Neben diesen Änderungen verglichen mit dem Endlosspiel, soll auch der modulbasierte Spielmodus ein Anpassungsmenü haben und auch eine Möglichkeit bieten die Theorie aufzurufen. Sind alle definierten Aufgaben abgeschlossen soll der Nutzer mit einem Klick auf „Nächste Übung” wieder in das Hauptmenü gelangen.

4.2 Konzeption der Systemkomponenten

Ich werde nun auf die Konzeption der Systemkomponenten eingehen und erläutern wie diese miteinander kommunizieren und interagieren sollen. Ich habe die Komponenten in vier Bereiche aufgeteilt, welche aus der Tonerkennung, dem Input und Output handling, der Aufgabenerzeugung und den Datenstrukturen bestehen.

4.2.1 Notensystem

Die Eingabe und Ausgabe der Noten arbeitet eng mit der Tonerkennung zusammen, da die erkannten Töne in Echtzeit auf dem Bildschirm in einem Notensystem angezeigt werden sollen. Das Notensystem wird dabei sowohl als Eingabe, als auch als Ausgabe verwendet. Es soll auf der linken Hälfte nur ermöglichen den Grundton anzuzeigen und auf der rechten Hälfte wahlweise entweder den vom Mikrofon eingelesenen Ton oder die Auswahl des Nutzers beim darüber hovern mit dem Cursor zu rendern. Dementsprechend muss es also möglich sein die rechte Hälfte entweder als Output für das Mikrofon zu verwenden (Outputmode) oder als Input für den User (Inputmode). Im Inputmode muss weiterhin die Möglichkeit existieren Halbtöne auswählen zu können, welche auf der gleichen Höhe liegen wie der dazugehörige Ganzton. Ist ein Halbtonschritt ausgewählt, so muss im Notensystem der relevante Ton mit dem passenden Vorzeichen angezeigt werden. Bei der Wahl des Vorzeichens kommt es hierbei auf die Richtung des Intervalls, also ob es nach Oben oder nach Unten vervollständigt wird. (2.2). Das Notensystem soll weiterhin zu jedem möglichen Ton auf dem Notensystem ein Objekt besitzen, welches einige Sachverhalte vereinfachen soll. So hat jede dieser Noten eine Hitbox, welche auf dem Notensystem so angeordnet liegt, dass sie den dortigen Ton repräsentiert. Die Notenobjekte werden von dem Notensystem initialisiert mit ihrem entsprechenden Notenwert, sowie dem Wert des nächst tiefer und höher liegenden Halbtons. Da jede Note selbst überprüft, ob diese aktuell aktiv ist, müssen sie auch mit dem Notensystem kommunizieren können, um ein Signal senden zu können, falls eine Note mittels eines Linksklicks ausgewählt wurde. Ist das der Fall, sendet diese Note dem Notensystem ein Signal, dass diese ausgewählt wurde, woraufhin das Notensystem alle anderen Noten deaktiviert, bis die ausgewählte Note durch einen weiteren Linksklick deaktiviert wurde. Wird die Note wieder abgewählt, sendet die Note ebenfalls ein Signal an das Notensystem, sodass dieses alle anderen Noten wieder freigeben kann und diese wieder wählbar sind. Das Notensystem weiß somit immer ob und welche Note aktiv ist und kann dadurch auf Anfrage den Wert der ausgewählten Note zur Verfügung stellen. Die Noten

müssen außerdem Zugriff auf das Notensystem haben, um abfragen zu können, ob der Nutzer aktuell einen Halbton setzen möchte, damit diese entsprechend ihren Sprite anpassen können. Je nach Input oder Outputmode müssen die Notenobjekte selbst entscheiden können, ob diese bei dem drüber fahren mit der Maus aktiv werden sollen oder nicht. Wird der Outputmode verwendet, so müssen die Noten über ihren Wert oder Namen angesprochen werden können. Es muss nun auch möglich sein mit dem Notensystem von außerhalb zu kommunizieren, damit der aktuell gesungene Ton ausgegeben werden kann. Die erkannten gesungenen Noten werden von der Spiellogik an die übergeordnete SceneMaster Klasse weitergegeben, welche schließlich dem Notensystem den anzuzeigenden Wert übergibt. Die Abstraktion zwischen Notensystem, Spielinterface (SceneMaster) und Spiellogik wird somit sichergestellt. Das Notensystem benötigt schließlich noch die Möglichkeit ähnlich wie den Spielerton setzen zu können auch den linken Aufgabenton bzw. Grundton setzen zu können.

4.2.2 Schnittstellen

Damit der Input aus dem Mikrofon und der Input aus dem Notensystem gebündelt an einer Schnittstelle zugreifbar sind, sollen die nötigen Daten gebündelt an einer Schnittstelle zur Verfügung gestellt werden. Es wird so eine Abgrenzung zwischen Spiellogik und Eingabe erreicht. Weiterhin soll ebenfalls eine Schnittstelle für Output Daten implementiert werden, welche die Daten gesammelt auf das UI schreibt.

4.2.3 Generierung der Töne

Da im Fokus der Anwendung die Verbesserung des relativen Gehörs steht, müssen vom Spiel Bezugstöne vorgegeben werden können. Es muss daher eine Komponente entwickelt werden, welche einfache Sinustöne berechnen kann. Ein Vorteil des Berechnen der Töne liegt darin, dass die Anwendung eine erhebeliche Menge an Speicherplatz spart. Desweiteren ist es so möglich sämtliche Frequenzen als Ton zu erzeugen. Dies bildet insofern einen Vorteil gegenüber eines festen Datensatzes mit Tonaufnahmen, dass es möglich ist die Grundstimmung der Aufgaben anzupassen, was bei einem festen Datensatz eine Verstimmung der Aufnahmen zur Folge hätte.

4.2.4 GUI Masterskripte

Um die Eingaben des Nutzers vom User Interface zu verarbeiten, werden generische öffentliche Methoden in sogenannten SceneMaster Klassen implementiert. Unter diese Eingaben fallen beispielsweise die Auswahl des Spielmodus im Hauptmnü oder das Öffnen der Optionen. Weiterhin sollen die SceneMaster Klassen mögliche Veränderungen am Interface setzen und globale Informationen an den SceneHandler übergeben, welcher globale Daten allen Klassen in allen Szenen zur Verfügung stellen soll. In den tatsächlichen Spielszenen übernehmen die SceneMaster außerdem die Kommunikation zwischen der Spiellogik und dem oben beschriebenen Notensystem.

4.2.5 Gamelogic

Die Spiellogik wird allgemein durch einen Elternklasse implementiert, sodass das Endlosspiel und der modulbasierte Übungsmodus von dieser Klasse erben können. Die grundsätzliche Kommunikation mit anderen Klassen bleibt bei beiden Klassen gleich, jedoch ist die Generierung dieser unterschiedlich. So wird bei dem Endlosspiel auf einem wahlweise zufälligem Grundton ein zufälliges Intervall nach oben oder unten gebildet. Bei dem modulbasierten Übungsmodus jedoch können die Aufgaben über eine .json Datei selbst definiert werden. Die Elternklasse der Spiellogik sollte die Schnittstellen zu anderen

Klassen definieren, darunter fallen der InputHandler, der Tongenerator und die GameUI / SceneMaster Klassen. Die Spiellogik soll alle Anpassungen vornehmen, welche den Spielverlauf verändern. Unter diese Entscheidungen fallen das Setzen eines festen Grundtons vom Nutzer, die Richtung der Intervalle und bei verändertem Input die entsprechenden Daten aus dem InputHandler zu verwenden. Außerdem soll die Spiellogik für die Generierung und Überprüfung der Aufgaben zuständig sein.

4.2.6 Datenstruktur und Utility Klassen

Schließlich müssen noch einige Hilfsklassen und eine Datenstruktur für Noten und Intervalle implementiert werden. Die Datenstruktur soll vorallem Noten und Intervalle von ihren Namen auf eine Frequenz bzw. ein Frequenzverhältnis in Cent in der Gleichstufigen Stimmung definieren. Sie definiert außerdem das Format der .json Datei zur Speicherung der Aufgaben, sowie die Grenzen der Stimmlagen.
Es wird weiterhin eine Hilfsklasse implementiert, welche häufig verwendete Berechnung mit Frequenzen und Tönen übernehmen soll. Es sollen Funktionen zur Berechnung eines Intervalls als Frequenzunterschied und zur Berechnung des daraus resultierenden zweiten Tons. Weiterhin werden Funktionen zur randomisierten Auswahl eines Grundtons und eines Intervalls aus der Datenstruktur zur Verfügung gestellt, sowie Hilfsfunktionen um den nächsten definierten Ton einer beliebigen Frequenz zu erhalten.

4.3 Konzeption eines Pitch Detection Algorithmus

Die Tonerkennung soll aus zwei Klassen zusammengesetzt sein. Dem FrequencyReader und dem FrequencyHandler. Der FrequencyReader soll das Mikrofon initialisieren, die Daten aus dem Mikrofon auslesen und schließlich der zweiten Klasse dem FrequencyHandler zur Verfügung stellen. Das Initialisieren soll dabei nicht nur das Mikrofon starten, sondern auch die erwähnten Maßnahmen (3.2.4) umsetzen, damit das Auslesen in Echtzeit geschehen kann. FrequencyReader stellt die Mikrofondaten und die zum Aufnehmen verwendete Samplerate bereit. FrequencyHandler greift auf die bereitsgestellten Daten von FrequencyReader zu um den folgenden Pitch Detection Algorithmus (1) auf die eingelesenen Daten des Mikrofons anzuwenden. Auf die Samplerate muss FrequencyHandler ebenfalls zugreifen um F_c berechnen zu können (3.2.4). Der Pitch Detection Algorithmus muss jeden Frame neu ausgeführt werden, um die neuen Mikrofondaten zu analysieren. FrequencyHandler stellt die Frequenz und den Namen der dazugehörigen Note anderen Klassen bereit.
Im Folgenden wird ein Konzept eines Pitch Detection Algorithmus vorgestellt, welcher auf den Ergebnisse der Analyse der Pitch Detection Algorithmen beruht. 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 aufbringen zu müssen.
Der erste Schritt ist zunächst, die Daten des Mikrofons mithilfe der Funktion einer FFT in die Frequenzdomän umzuwandeln. Üblicherweise muss man bei dem Berechnen einer FFT eine Fensterfunktion und die Größe des Fensters festlegen. Es sollen somit mögliche Informationsverluste vermieden werden, welche durch eine falsch „abgeschnittene" Periode zustande kommen. Eine Datensatzlänge in der Größe einer Potenz von 2 soll dieses Problem verhindern (2.1). Infrage kommen die Fensterfunktionen: Rectangular, Triangle, Hamming, Hanning, Blackman und BlackmanHarris. Diese unterscheiden sich in der Komplexität und dadurch auch in der Genauigkeit. Welche Fensterfunktion am besten angewendet wird, muss später während der tatsächlichen Implementation entschieden werden. 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 [YMFA05], 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. Dieser Ansatz ist möglich, da die menschliche Stimme auf verschiedenen Tonhöhen größtenteils die gleichen Anteile an Obertönen erzeugt, 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, zu berechnen gilt jedoch die Frequenz. Um die Frequenz aus dem Index zu berechnen, wird ein Wert benötigt, 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). 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.

4.4 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 aussagekräftigeres 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 wie gewünscht funktioniert hat lassen sich so hoffentlich Zusammenhänge zu nicht optimaler Hardware herstellen.

5 Prototypische Realisierung in Unity3D

5.1 Implementierung des UI

Hauptmenü

Einstellungen

Theoriebuch

Endlosübung

Modulübung

5.2 Implementierung der Systemkomponenten

Im Folgendenen wird die Entwicklung der protypischen Trainingssoftware zur Gehörbildung näher beleuchtet und der Prozess der Umsetzung der Konzeption gezeigt. Ich werde die konzipierten Systemkomponenten alle ansprechen und wie diese umgesetzt wurden.

In Unity gibt es die sogenannte MonoBehaviour Superklasse, welche einige Funktionen implementiert welche die Unity - Engine zu bestimmten Zeiten aufruft, wenn ein Skript, welches von MonoBehaviour erbt, in einer Szene eingebunden ist. Zu den wichtigsten Funktionen zählen die Funktion Start() und Update(). Start wird automatisch einmal während der Initialisierung des Skripts aufgerufen, wohingegen Update jeden Frame genau einmal aufgerufen wird. Es existieren noch weitere dieser Funktionen, wie etwa die Funktion FixedUpdate(), welche unabhängig von der Framerate in einem fixen Zeitintervall aufgerufen wird.

Notensystem

Das Herzstück der Ein- und Ausgabe bildet das Notensystem. Da dieses nicht nur als Anzeige des Grundtons der Aufgabe verwendet werden sollte, sondern zusätzlich entweder als Anzeige des gesungenen Tons oder als Eingabe des Tons mit der Maus. Umgesetzt wurde dieses komplexe Modul im wesentlichen in zwei Klassen. Diese sind zum einen die Klasse OnehotNote und zum anderen der NoteSystemHandler. Wie der Name der OnehotNote bereits erahnen lässt implementiert diese das oben beschriebene Verhalten für eine einzelne Note, wobei immer nur eine Note aktiv sein darf. Wie man in der Abbildung erkennen kann, hat jede der Noten eine „Hitbox ” in welcher sie auf dem Notensystem liegt. Diese Hitbox wird mithilfe eines Box Colliders umgesetzt, welcher innerhalb der OnehotNote Klasse es erlaubt abzufrgen, ob die Maus über dem Boxcollider hovert. In der folgenden Abbildung kann man die Boxcollider sehr gut erkennen an den grünen Boxen innerhalb des Notensystems. Weiterhin kann man die Struktur des Notensystem Gameobjects erkennen. Es wurden zwei Blöcke an Hitboxen zur Erkennung der Noten implementiert, wobei alle Notenobjekte den gleichen Skriptcode verwenden können, da der NotesystemHandler die jeweiligen OnehotNotes so initialisiert, dass diese das richtige Verhalten aufweisen. Jede

Note hat Zugriff auf den assoziierten SpriteRenderer um den Notenkopf bei verändertem Vorzeichen anzupassen, welcher vom NotesystemHandler gesetzt bzw. bezogen werden kann. Weiterhin hat die Note Zugriff auf den Boxcollider, um die Methoden OnMouseEnter, OnMouseExit, sowie OnMouseDown implementieren zu können. Um sicherzustellen, dass immer nur die Note ihren Sprite setzt, welche gerade tatsächlich gewollt ist, werden die boolschen Parameter StayOn, StayOff und EnableClickIn gesetzt.

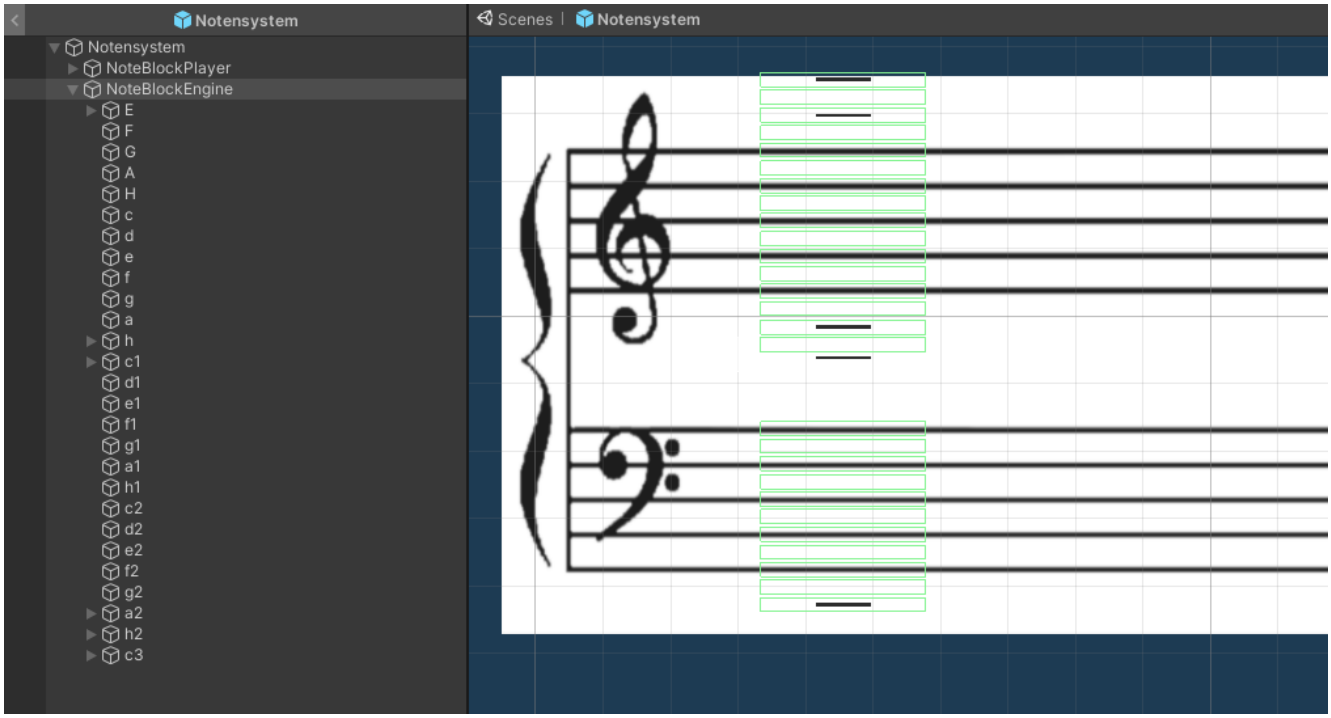

Abbildung 5.1: Aufbau des Notensystems in Unity

Während der Initialisierung des Notensystems, setzt dieses bei allen Noten zunächst den Parameter EnableClickIn. Dieser sorgt dafür, dass der linke Notenblock nicht auf die Maus reagiert, der rechte Notenblock jedoch schon. Klickt der Nutzer auf eine Note, so wird dabei der Paramter StayOn dieser negiert, sodass eine nicht angewählte Note angewählt bleibt oder eine angewählte Note wieder freigegeben wird und nicht mehr angezeigt wird. Bei dieser Aktion wird gleichzeitig dem Notensystem der gleiche Wert übergeben, woraufhin der NotensystemHandler bei allen anderen OnehotNotes den Parameter StayOff setzt. Will man bei einer Note durch klicken den Wert setzen im Notensystem, so müssen alle drei dieser Parameter negativ sein, damit die Note gesetzt werden kann.
Ist die Option zum Singen aktiviert, so kann der NotesystemHandler auch alleine extern alle Noten so ansteuern, dass der gesungene Ton angezeigt wird. Möglich gemacht wird dies durch die öffentlichen Methoden RenderNote, AddFlat, AddSharp, sowie ResetSprite. Über diese Methoden ist es möglich jegliche Veränderungen an dem Sprite durchzuführen und die Note somit extern zu setzen. Das Notensystem erfüllt somit sowohl die Aufgabe der Eingabe, als auch die Ausgabe. Der NotesystemHandler bezieht jedoch nicht selbst die Mikrofondaten, sondern stellt lediglich der Gamelogic eine Methode zur Verfügung um den gewünschten Ton setzen zu können. Der Ton muss in dem normierten internen Format als Integer übergeben werden. Er wird dann zunächst daraufhin überprüft ob er einen Halbton darstellt. Ist das der Fall, wird bei der Wahl des Notenobjekts darauf geachtet, ob die Aufgabe Intervall nach Oben oder Unten vervollständugt erwartet, um dann schließlich die richtige Note auszuwählen mit passendem Vorzeichen. Es wird außerdem die zuletzt vom Spieler gesungene Note gespeichert, um einige Rechenschritte zu sparen, sollte Sie mehrmals hintereinander gesetzt werden.

Schnittstellen

Es wurden zwei Schnittstellen implementiert, welche jeweils Inputdaten und Outputdaten sammeln und setzen. Der InputHandler hat Zugriff auf den FrequencyHandler, das Notensystem und den UIHandler, um von diesen alle relevanten Daten beziehen zu können. Es werden jeden Frame die gesungene Frequenz, die erkannte Note, die mit der Maus ausgewählten Note, sowie die Optionen des UI gespeichert. Diese Werte werden anderen Klassen öffentlich zum Lesen zur Verfügung gestellt.
Die Outputschnittstelle, übernimmt während des Spiels die Aufgabe die Daten richtig in der Szene zu platzieren. Die Spiellgik muss dafür nur die notwendigen Daten dem UIHandler übergeben. Um eine Aufgabe vollständig anzuzeigen, sind lediglich der geforderte Ton, der dazugehörige Grundton und das Intervall zwischen diesen Tönen zu übergeben. Der UIHandler setzt anschließend auf dem Notensystem den Grundton, sowie den gesungenen Spielerton und passt den Text der Aufgabe an, sodass das geforderte Intervall abgefragt wird. Es ist außerdem möglich von der Spiellogik einen angepassten Aufgabentext zu übergeben, welcher für das modulbasierte Lernen notwendig ist. Bei dem modulbasierten Modus liest die Spiellogik einen anpassbaren Aufgabentext aus einer .json. Dieser Text kann dann über diese alternative Methode gesetzt werden.

Generierung der Töne

Die Generierung der Töne erfolgt komplett isoliert innerhalb einer Klasse, welche lediglich eine Methode zur Verfügung stellt. Die Spiellogik kann die Methode PlayFreq mit einer Frequenz aufrufen, woraufhin ein Sinuston mit einer Samplerte von 44100 Hz berechnet wird. Um einen Sinuston zu berechnen, muss lediglich über die gesamte Länge der Samples eine Sinuswelle gelegt werden. Es wird dementsprechend ein Array mit 44100 Felder gefüllt

$$\text{ton}[i] = \sin(\frac{\pi * 2 * i * F}{F_s})$$

wobei i der Index des Arrays, F die Frequenz des zu berechnenden Tons und F_s die Samplerate, also 44100, ist.
Dieser Ton wird nach der Berechnung direkt einer AudioSource als Clip hinzugefügt und die AudioSource im Anschluss ausgeführt. Diese Umsetzung ermöglicht das mehrfache abspielen des Tons, mit einer einmaligen Berechnung, da der Ton immer in der AudioSource gespeichert wird.

GUI Masterskripte

Die sogenannten SceneMaster existieren in jeder Szene genau einmal und übernehmen allgemeine Aufgaben des UI und der Szenenkontrolle, wie etwa ein Szenenwechsel, sowie die Kommunikation mit dem SceneHandler, welcher globale Daten speichert. Im Fall der Spielszenen sind die SceneMaster die bereits angesprochenen UIHandler. Im Fall des SceneMasters des Hauptmenüs, implementiert dieser vorallem Methoden, welche das Wechseln von Szenen erlauben, sodass diese von einem Button aufgerufen werden können. Es wird zum Wechseln einer Szene lediglich der Aufruf der statischen Klasse SceneHandler mit dem Szenennamen als Übergabeparameter benötigt. Außerdem setzt er einige globale Werte, welche notwendig sind um das Spiel zu starten. Im Fall des Theoriebuchs werden öffentliche Methoden implementiert, welche das Wechseln zwischen den einzelnen „Buchseiten ” ermöglicht. Der SceneMaster der Optionenszene setzt ebenfalls vorallem Methoden um, welche das setzen von globalen Werten ermöglichen, wie etwa die angepassten Parameter der Tonerkennung.

Gamelogic

Da es zwei verschiedene Spielmodi gibt, mussten auch zwei unterschiedliche Gamelogicklassen geschrieben werden, welche jedoch viele Gemeinsamkeit teilen und daher von einer Elternklasse erben können. Die Elternklasse „BaseGameLogic ” implementiert die notwendigen Schnittstellen zu anderen Klassen, sowie die äußerst wichtige Methode „RefreshInputVals ”. Gamelogic hat Zugriff auf die Utility Klassen und auf die Datenstrukturen, sowie auf den Tongenerator und den Inputhandler. Außerdem kann die Gamelogic Klasse auf einen JsonLoader zugreifen und auf den UIHandler. Die Methode „RefreshInputVals ” ist wichtig, da diese nach jeder UI - Eingabe die Gamestate relevanten Parameter neu setzt, um so die vom Nutzer eingegebene Werte aus dem UI spätestens im nächsten Frame schon zu übernehmen. Es kann bei den Parametern zu einer Veränderung des Grundtons der Aufgabe kommen, falls der Nutzer diesen selbst wählen möchte, statt ihn zufällig generieren zu lassen. Tritt dieser Fall ein, so muss eine neue Aufgabe generiert werden, welche den fest gewählten Grundton verwendet. Die Methode überprüft daher bei jedem Aufruf, welcher Grundton verwendet werden muss und passt diesen dementsprechend an. Es werden die zwei virtuellen Methoden „NextExer ” und „NextExerButton ” vorgegeben, welche in den Unterklassen zu definieren sind. Es handelt sich bei diesen Methoden um die Anbindung zum Button um die nächste Übung zu beginnen (NextExerButton) und der Methode, welche alle Parameter setzt und Funktionsaufrufe tätigt um die nächste Übung zu generieren und anzuzeigen (NextExer). Es werden auch nicht-virtuelle Methoden implementiert, welche bestimmte Sachverhalte umsetzen, die für alle Spielmodi gleich sind. Unter diese fallen neben RefreshInputVals auch Methoden zum beziehen des Inputs aus der richtigen Quelle und zur Überprüfung der Aufgabe. Es wird außerdem eine zeitlich abhängige Überprüfung der Aufgaben implementiert, welche das richtig Beantworten der Aufgabe nur zulässt, wenn der Nutzer die richtige Note lang genung gesungen hat. Diese Überprüfung wird mithilfe der Unity internen Zeit umgesetzt, indem eine Flag gesetzt wird, wenn das erste Mal der Ton gesungen wird und in den folgenden Frames überprüft wird, ob die Zeit zwischen der ersten Eingabe und der neusten Eingabe bereits die Zeit der Aufgabe erreicht. Ist das der Fall, wird die Flag gesetzt, dass die Aufgabe gelöst wurde. Die Unterklassen der verschiedenen Spielmodi müssen lediglich die zwei vorgestellten virtuellen Methoden überschreiben.
Die Spiellogik für das Endlosspiel ruft jeden Frame die Methode zur Überprüfung der gehaltenen Noten in der Elternklasse mit dem Input auf. Wenn die Aufgabe gelöst ist, wird dem Nutzer Feedback über eine Ausgabe gegeben und dazu aufgefordert den Button zur nächsten Aufgabe zu drücken. Der Button führt dann die erste der überschriebenen Methoden aus (NextExerButton), welche überprüft, ob die Aufgabe tatsächlich abgeschlossen wurde, ist das der Fall wird NextExer aufgerufen. NextExer generiert eine neue Aufgabe basierend auf den Präferenzen des Nutzers und startet die nächste Aufgabe mit dem Abspielen des Grundtons.
Ähnlich wie das Endlosspiel ruft auch die Spiellogik der Module jeden Frame die Überprüfung der gehaltenen Noten mit dem Input auf. Der Ablauf der Überprüfung ist identisch bis zu dem Punkt wo der Nutzer auf den Button zur nächsten Aufgabe klickt, woraufhin auch hier die Methode NextExerButton aufgerufen wird. Der Unterschied dieser Implementierung der Methode, zu der des Endlosspiels, ist ein weiterer Fall der Überprüfung ob die Aufgabe abgeschlossen wurde oder nicht. Es wird hier zusätzlich überprüft, ob das letzte Modul vollendet wurde. Wurde das letzte Modul beantwortet, ruft die Methode den SceneHandler auf um die Szene zum Hauptmenü zu wechseln. Ist das jedoch nicht der Fall, so wird auch hier die Methode NextExer aufgerufen um die nächste Aufgabe zu setzen. Der Unterschied der Methode NextExer zu ihrem Pendant ist, dass hier die Bestandteile der Aufgabe aus der eingelesenen .json Datei gesetzt werden müssen.

Datenstruktur und Utility Klasse

5.3 Tonerkennung

Die Tonerkennung wurde realisiert mithilfe von zwei Klassen, welche von MonoBehaviour erben und jeweils ein eigenes GameObject in der Spielszene darstellen, sowie einer AudioSource, welche ebenfalls als alleinstehendes GameObject in der Szene existiert. Wie bereits konzipiert ist eine der Klassen (FrequencyReader) zuständig dafür das Mikrofon zu initialisieren, wohingegen die andere Klasse (FrequencyHandler) die tatsächliche Tonerkennung durchführt.
FrequencyReader implementiert ein öffentliches Attribut „playback", welches die AudioSource mit den geladenen Mikrofondaten repräsentiert und eine Konstante „SampleRate", welche die Samplerate von 44100Hz festlegt und freigibt. Die Klasse implementiert nur die Funktion Start(). Es werden in Start() das gewählte Eingabegerät aus dem SceneHandler abgerufen und daraufhin der AudioClip von playback direkt gesetzt. Es muss dann noch der Parameter loop der AudioSource playback auf true gesetzt werden. Dieser Parameter ermöglicht es dass die AudioSource den festgelegten Clip unendlich oft wiederholt und nicht nach einem Durchlauf abbricht. Es muss abschließend noch die Position der Daten des Mikrofons solange gepolled werden bis Daten vorliegen und die AudioSource abgespielt werden. Der aufgenommene Clip muss über eine AudioSource abgespielt werden, sodass die AudioDaten aus dem Mikrofon übernommen werden können. Ab diesem Punkt führt die Klasse FrequencyHandler den bereits vorgestellten Algorithmus (1) auf den AudioDaten des Mikrofons aus. Es wird die berechnete Frequenz, sowie die korrespondierende Note anderen Klassen zur Verfügung gestellt.

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(absMaxIndex)
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 1 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
```

Berechnung der Laufzeitkomplexität des vorgestellten Algorithmus

Es gilt zu zeigen, dass der vorgestellte Pitch Detection Algorithmus effizient ist. Ich werde die Funktion „HandleOvertones" zunächst untersuchen, da diese von der Funktion „ComputeFrequency" aufgerufen wird. Man kann sehen, dass die Zeilen 25 und 26, sowie die Zeile 39 nur einmal pro Ablauf aufgerufen werden und daher vernachlässigbar sind. Sei n die Länge des Spektrums. Die Schleife wird im worst case Szenario n mal und die Zeile 27 $n+1$ durchlaufen.

$$3 * \mathcal{O}(1) + 7 * \mathcal{O}(1) * \mathcal{O}(n) + \mathcal{O}(1) = \mathcal{O}(n)$$

Die Laufzeitkomplexität der Funktion beläuft sich demnach auf $\mathcal{O}(n)$. Die Komplexität der Funktion „ComputeFrequency" werde ich zunächst betrachten ohne auf die Funktionsaufrufe einzugehen. Die Schleife in dieser Funktion wird immer n mal und die Zeile 7 $n+1$ ausgeführt. Alle anderen Zeilen werden nur einmal ausgeführt, sodass auch hier eine Schranke von $\mathcal{O}(n)$ gesetzt werden kann. Die aufgerufenen Funktionen sind jedoch nicht so Laufzeiteffizient. Berücksichtigt man noch die Funktionsaufrufe, so stellt man schnell fest, dass die Laufzeitkomplexität höher ausfällt. Geht man davon aus, dass GetSpectrumData eine gewöhnliche rekursive FFT berechnet, so beläuft sich die Laufzeitkomplexität dieser auf $\mathcal{O}(n \log n)$. Eine Newton Interpolation hat sogar eine maximale Laufzeitkomplexität von $\mathcal{O}(n^2)$, diese Laufzeitkomplexität begründet auch, warum nur drei Punkte interpoliert werden. Da die Anzahl der zu interpolierenden Punkte fest ist, werde ich die Laufzeitkomplexität der Interpolation als konstant annehmen. Die Komplexität der Max, Avg und Copy Funktionen sind ebenfalls linear.

$$\mathcal{O}(n \log n) + 4 * \mathcal{O}(n) + 5 * \mathcal{O}(1) + 5 * \mathcal{O}(1) * \mathcal{O}(n) = \mathcal{O}(n \log n)$$

Den größten Einfluss auf die Zeitkomplexität nimmt die von Unity zur verfügung gestellte Funktion, sodass der Algorithmus nicht weiter optimiert werden kann, da diese Funktion essenziell ist.

6 Validierung der erarbeiteten Methoden und Konzepte

Im folgenden Kapitel werde ich zunächst auf die unternommenen Maßnahmen zur Vorbereitung der Evaluatione eigehen. Im nächsten Schritt wird die Durchführung der Evaluation näher erläutert und die Entscheidungsfindung der Durchführungsart beleuchtet. Schließlich werden die Ergebnisse der Evaluation ausgewertet und hinsichtlich der zu untersuchenden Punkte gedeutet. Das Ziel der Evaluation soll sein, die implementierte Tonerkennung in Unity auf eine möglichst großen Menge an Systemen testen zu können und somit die echtzeitkomponente der Anforderung überprüfen zu können.

6.1 Vorbereitung

Da die Evaluation nicht fest an einem Standort unter Idealbedingungen durchgeführt wird, sondern bei jedem Nutzer lokal auf den eigenen Geräten, ist es wichtig die Anwendung zunächst für eine möglichst große Bandbreite an Endnutzergeräten zur Verfügunge zu stellen. Die Anwendung wurde dementsprechend für die Betriebssysteme Windows 10 (32 Bit) und Windows 10 (64 Bit) exportiert, sowie für OSX. Zusätzlich zu der Anwendung erhält jeder Anwender ein Dokument, in welchem beschrieben wird, wie die Anwendung auszuführen ist, sowie eine Bedienungsanleitung zu der Anwendung. Die Bedienungsanleitung geht auf das Einstellen der Tonerkennung auf die eigene Stimme ein, sowie mögliche Lösungsanätze, falls diese die Stimme bzw. den Ton nicht genau erkennen kann. Weiterhin werden die zwei Übungsmodi beschrieben und wie diese zu handhaben sind.

6.2 Durchführung

6.3 Auswertung

7 Zusammenfassung

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.

[AWH16] Tim Althoff, Ryen W White, and Eric Horvitz. Influence of pokémon go on physical activity: study and implications. *Journal of medical Internet research*, 18(12):e315, 2016.

[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.

[dav] dave@tonesavvy.com. Tonedear. Online.

[dlCMS01] Patricio de la Cuadra, Aaron Master, and Craig Sapp. Efficient pitch detection techniques for interactive music. 01 2001.

[duo] Duolingo.

[e.V10] Hessisches Telemedia Technologie Kompetenz-Center e.V. Serious games information center. Online, 2019-04-10.

[GGS11] Stefan Göbel, Michael Gutjahr, and Ralf Steinmetz. What makes a good serious game—conceptual approach towards a metadata format for the description and evaluation of serious games. In *5th European Conference on Games Based Learning*, pages 202–210, 2011.

[GVG06] Manuela Gußmack, Oliver Vitouch, and Bartosz Gula. *Latentes absolutes Gehör–eine omnipräsente Fähigkeit?* na, 2006.

[Jak] Hans Lavdal Jakobsen. Earmaster. Online.

[LLW18] Yan-Xin Liu, Henrik Hautop Lund, and Li-Li Wu. Playful cognitive training with physical interactive tiles for elderly. In *2018 international conference on information and communication technology robotics (ICT-ROBOT)*, pages 1–4. IEEE, 2018.

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

[Meg01] Megan. How do i get unity to playback a microphone input in real time? Online, 2017-01-01.

[Meg01] Megan. Coding pirates. Online, 2017-09-01.

[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.

[OS04] Alan V Oppenheim and Ronald W Schafer. From frequency to quefrency: A history of the cepstrum. *IEEE signal processing Magazine*, 21(5):95–106, 2004.

[Rie] Walter Riedinger. Musikgrad. Online.

[RSC+74] Myron Ross, Harry Shaffer, Andrew Cohen, Richard Freudberg, and Harold Manley. Average magnitude difference function pitch extractor. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 22(5):353–362, 1974.

[Sen01] UdK Berlin Sengpiel. Harmonische, partialtöne, teiltöne und obertöne. 2002-11-01.

[Spe47] Herbert S Spencer. Ear training in music education. *Music Educators Journal*, 33(4):44–69, 1947.

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

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

[Tec09c] 2020 Unity Technologies. Unity3d dokumentation getspectrumdata. Online, 2021-02-09.

[Tec09d] 2020 Unity Technologies. Unity3d dokumentation microphone. Online, 2021-02-09.

[Ves] Morten Vestergaard. Earbeater. 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.